

**UCLA**

**UCLA Electronic Theses and Dissertations**

**Title**

Data Completion and Robust Principal Component Analysis under Low-rank Restrictions

**Permalink**

<https://escholarship.org/uc/item/9168c9jb>

**Author**

Chao, Zehan

**Publication Date**

2022

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA  
Los Angeles

Data Completion and Robust Principal Component Analysis under Low-rank  
Restrictions

A dissertation submitted in partial satisfaction  
of the requirements for the degree  
Doctor of Philosophy in Mathematics

by

Zehan Chao

2022

© Copyright by

Zehan Chao

2022

# ABSTRACT OF THE DISSERTATION

Data Completion and Robust Principal Component Analysis under Low-rank  
Restrictions

by

Zehan Chao

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 2022

Professor Deanna M. Hunter, Chair

In the real world, many kinds of high-dimensional data such as images [1, 2, 3], documents [4, 5], user-rating data [6, 7], and health-related data [8] have internal low-dimensional structures. Mathematicians conceptualize the idea of ‘low-dimension’ as low-matrix-rank and developed various dimensionality reduction methods such as principal component analysis (PCA) [9] and non-negative matrix factorization (NMF) [10] under the low-matrix-rank assumption. This thesis contains four projects during my Ph.D. study. The target data sets of the first three projects are under the assumption of low-matrix-rank or low-tensor-rank. The first part focuses on a matrix completion task, where we propose a data completion method with convex regularizers to address the fragmented data issue. We then combine the data completion method with a temporally hierarchical attention network (THAN) to predict human stress levels with recovered sensor data. In the second work, we propose a simple but efficient weighted higher-order singular value decomposition (HOSVD) algorithm for recovering the tensor data from noisy observations. We also combine the weighted HOSVD and the total variation minimization method to fill in the missing data for images and videos efficiently. In the third work, we propose a fast non-convex algorithm, Robust Tensor

CUR (RTCUR), for large-scale tensor robust principal component analysis (TRPCA) problems. The main advantage of RTCUR over other TRPCA methods is the computational efficiency; we demonstrate the efficiency and effectiveness of RTCUR on both synthetic and real-world datasets. In all these three works, we explore the connection between the rank in mathematical definition and the real-world data by developing algorithms with the low-rank assumption that solves real-world tasks.

The last work studies a quantitative framework to infer the political bias and source quality of media outlets from text. We collect the tweets that each media outlet posted during a specific time range and use a bidirectional long short-term memory (LSTM) neural network to infer the bias and quality values for each tweet.

The dissertation of Zehan Chao is approved.

Christopher R. Anderson

Mason A. Porter

Andrea Bertozzi

Deanna M. Hunter, Committee Chair

University of California, Los Angeles

2022

*To my father in heaven  
I wish you can see this*

# TABLE OF CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Matrix Completion Under Low-rank Assumption</b>	<b>4</b>
2.1	Motivation	4
2.1.1	Stress Level Prediction	5
2.2	Problem Formulation and Proposed Approach	6
2.2.1	Data Completion with Diurnal Regularizers	7
2.2.2	THAN for Stress Level Prediction	11
2.3	Experiments	15
2.3.1	Sensor Data Completion	17
2.3.2	Stress Level Prediction	18
2.4	Analysis and Discussions	18
2.5	Conclusion	22
<b>3</b>	<b>Tensor Completion Under Low-rank Assumption</b>	<b>24</b>
3.1	Motivation and Tensor Background	25
3.1.1	Tensor Preliminaries and Notations	26
3.2	Problem Formulation and Related Work	29
3.3	TV Minimization Algorithm	32
3.3.1	Matrix Denoising Algorithm	32
3.3.2	Tensor Completion with TV	33
3.4	Theoretical Error Bound	34
3.5	Experiments	35
3.5.1	Simulations for Weighted HOSVD	35



3.5.2	Simulations for TV with Initialization from Weighted HOSVD . . . . .	38
3.6	Numerical Results and Discussion . . . . .	39
<b>4</b>	<b>Tensor Robust PCA and CUR implementation . . . . .</b>	<b>44</b>
4.1	Robust Principle Component Analysis (RPCA) Background . . . . .	45
4.2	Tensor Robust Principal Component Analysis (TRPCA) . . . . .	47
4.3	Tensor CUR Decompostions . . . . .	48
4.3.1	Theoretical Guarantee for Sparsity . . . . .	50
4.4	Proposed Approach . . . . .	51
4.4.1	Step (I): Update Sparse Component $\mathcal{S}$ . . . . .	53
4.4.2	Step (II): Update Low-Tucker-rank Component $\mathcal{L}$ . . . . .	54
4.4.3	Computational Complexity . . . . .	54
4.4.4	Four Variants of RTCUR . . . . .	56
4.5	Numerical Experiments . . . . .	58
4.5.1	Synthetic Examples . . . . .	59
4.5.2	Color Video Background Subtraction . . . . .	62
4.5.3	Robust Face Modeling . . . . .	64
4.5.4	Network Clustering . . . . .	65
4.6	Conclusion and Future Work . . . . .	68
<b>5</b>	<b>Inference of Media Bias and Content Quality . . . . .</b>	<b>71</b>
5.1	Introduction . . . . .	72
5.1.1	Our Contributions . . . . .	76
5.1.2	Organization of this Chapter . . . . .	77
5.2	Background and Related Work on NLP Methods . . . . .	77
5.2.1	Naive-Bayes Method . . . . .	78

5.2.2	Support-Vector Machines (SVMs)	79
5.2.3	Decision Trees and Random Forests	80
5.2.4	Artificial Neural Networks (ANNs)	81
5.2.5	Long Short-Term Memory (LSTM) Neural Networks	82
5.3	Data Sets	83
5.4	Generation of a Media-Bias Chart	84
5.4.1	Text Preprocessing	85
5.4.2	Bidirectional LSTM Neural Network	85
5.4.3	Training and Results	87
5.5	Evaluation of the LSTM Network's Performance	88
5.5.1	Computational Experiments	89
5.5.2	Results	90
5.6	Conclusions and Discussion	92
<b>A List of Media Outlets and their Number of Tweets</b>		<b>94</b>
<b>References</b>		<b>97</b>

## LIST OF FIGURES

2.1	The overall schema of THAN. . . . .	12
2.2	The F1 scores of eight methods in stress level prediction . . . . .	20
2.3	The RMSE scores of DCDR in sensor data completion . . . . .	21
2.4	The F1 scores of THAN in stress level prediction . . . . .	21
3.1	Relative error for uniform sampling . . . . .	37
3.2	Relative error for non-uniform sampling . . . . .	37
3.3	The first frame of tested videos . . . . .	38
3.4	Comparison between TVTC and wHOSVD-TV . . . . .	40
3.5	Nuclear norm comparison for different recovery patterns . . . . .	42
3.6	Comparison of singular values between TV-recovery and original image . .	43
4.1	Illustration of the Fiber CUR Decomposition . . . . .	49
4.2	Illustration of the Chidori CUR Decomposition . . . . .	50
4.3	Empirical phase transition for all RTCUR variants . . . . .	58
4.4	Runtime vs. dimension comparison among RTCUR-F, RTCUR-R, GD, AAP, and IRCUR . . . . .	60
4.5	Runtime vs. relative error comparison among RTCUR-F, RTCUR-R, AAP, and IRCUR . . . . .	61
4.6	Visual results for color video background subtraction . . . . .	63
4.7	Visual results for robust face modeling . . . . .	69
4.8	Three communities detected in the “High-Dimensional Data Analysis” co- authorship network . . . . .	70
5.1	The Ad Fontes Media-Bias Chart (version 5.1) . . . . .	75

5.2	Flowcharts for inferring ideological biases and content qualities of the media outlets . . . . .	76
5.3	A schematic illustration of a simple decision tree . . . . .	80
5.4	A schematic illustration of a memory cell in an LSTM neural network . . .	82
5.5	Number of tweets from the 65 examined media outlets . . . . .	85
5.6	The structure of a bidirectional LSTM network . . . . .	87
5.7	Comparison of the (bias, quality) scores from the AFMBC to the (bias, quality) scores from the LSTM network . . . . .	88

## LIST OF TABLES

2.1	The sensor data completion performance of five methods for the activities Sleep and Walk . . . . .	18
2.2	The stress level prediction performance of eight methods with binary-class stress levels . . . . .	19
2.3	The stress level prediction performance of eight methods with 5-class stress levels . . . . .	19
2.4	The stress level prediction performance of THAN with different depths of temporally hierarchical structures. . . . .	20
2.5	The stress level prediction performance of THAN with different features . . . . .	21
3.1	Key notation in Chapter 3 and 4 . . . . .	26
3.2	The relative square error and runtime for different algorithms on video data . . . . .	39
4.1	Computational complexity for each step from Line 2 . . . . .	56
4.2	Video information and runtime comparison for color video background subtraction task. . . . .	62
4.3	Runtime comparison for face modeling task . . . . .	65
4.4	Runtime comparison of TRPCA algorithms: RGD and RTCUR . . . . .	68
5.1	Key notation in Chapter 5 . . . . .	78
5.2	The Pearson correlations from media-outlet split . . . . .	91
5.3	The Pearson correlations from tweet-level split . . . . .	91
A.1	Bias and quality scores in the Ad Fontes Media-Bias Chart . . . . .	96

## ACKNOWLEDGMENTS

Over the past six years, many people have helped me on my way to this Ph.D. thesis. Please accept my sincere proclamation of gratitude to everyone who has helped me and taught me right from wrong. Here, I would like to order my appreciation chronologically. My first acknowledgment goes to the teacher of the first class I took at UCLA, professor Luminita Vese, for warmly encouraging me after two failures in my qualification exams. It would be so hard to continue my study without your encouragement. Next, I would express my gratitude to my committee member, professor Andrea Bertozzi, for guiding me on the first project after completing my qualification exams. At that time, I only knew a little about research, but you offered me plenty of opportunities and taught me how to conduct research as a mature scientist.

My utmost gratitude goes to my advisor, professor Deanna Needell, for helping me find the right direction in my research and guiding me through multiple projects. None of my Ph.D. research work would have been possible without your continuous support and encouragement in the past years. Especially in the last year, when I was suffering from losing a family member, your support and comfort were significant and helped me move on with my life. I learned not only mathematics but also research attitudes from you, which will surely benefit the rest of my life.

I want to express my appreciation to the next committee member, Professor Mason A. Porter, for providing many insightful and thoughtful suggestions for my work. Addressing your comments in my paper was initially painful but eventually beneficial. Last but not least, I want to show thanks to Professor Chris Anderson for all the friendly help and insightful communication about my dissertation project.

I also want to thank Dr. Longxiu Huang and Dr. Hanqin Cai for guiding me through the details of the Tensor CUR project, which account for a significant contribution to this thesis. After the remote years during COVID and returning to campus, you are the most frequent people I meet in person, and I will cherish those meetings in the future.

In the end, I am grateful to my mother and my wife for accompanying me through the struggles and tough times of my life.

# CHAPTER 1

## Introduction

Many real-world data science tasks, such as image recovery [1, 2], text classification [4, 5], and recommender system [6, 7], are handled with an assumption that the given data set has an internal low-dimensional structure. The concept of ‘low-dimensional’ is mathematically ambiguous; hence the idea of low-dimension is commonly converted into low-matrix-rank, which is well-defined and could be solved with proper algorithms. In order to solve the data-related tasks (i.e. data completion, data denoising, etc.) under the low-rank assumption, researchers developed various algorithms such as singular value thresholding (SVT) [11], alternating projection [12], and convex optimization approaches (which convert the original task into a convex problem and apply convex optimization algorithms) [13, 14]. This paper will mainly focus on two tasks: data completion and robust principal component analysis (RPCA). More specifically, we will work on data completion tasks in Chapter 2 and Chapter 3 and discuss tensor RPCA in Chapter 4.

We briefly introduce the steps through the history of matrix completion tasks under the low-rank assumption, and we will leave the relevant parts about tensor completion to Section 3.1.

Suppose we have a partially observed matrix  $M$  under the low-rank assumption and give a deterministic sampling pattern  $\Omega$ . The most intuitive optimization problem raised here is:

$$\begin{aligned} \min_X \quad & \text{rank}(X), \\ \text{s.t.} \quad & P_\Omega(X) = P_\Omega(M). \end{aligned}$$



Here,  $P_\Omega(\cdot)$  denotes the linear mapping  $\mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}$  which only keep the entries in  $\Omega$  unchanged and set all other entries to be 0. We also use  $X_\Omega$  as a shorter notation for  $P_\Omega(X)$ .

However, due to the computational complexity (NP-hard) of this minimization problem [13], researchers have developed workarounds by defining new optimization problems which could be done in polynomial time. Two prominent substitutions are the nuclear norm minimization (NNM) [13] and low-rank matrix factorization (LRMF) [11]. The NNM could be expressed as the following optimization problem:

$$\begin{aligned} \min_X \quad & \|X\|_* \\ \text{s.t.} \quad & X_\Omega = M_\Omega, \end{aligned}$$

where  $\|\cdot\|_*$  stands for the sum of singular values. The LRMF could be expressed as the following optimization problem:

$$\begin{aligned} \min_{A,B} \quad & \|(X - AB)_\Omega\|_F \\ \text{s.t.} \quad & A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{r \times n}, \end{aligned}$$

where  $A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{r \times n}$  are restricted to be the low-rank components (with some fixed  $r$ ). These two workarounds, NNM and LRMF, both induce a series of variant methods to cope with the diversity of real-world data.

Since the operator  $\|\cdot\|_*$  is convex, NNM could be efficiently solved with gradient descent algorithms [13]. Additionally, one could add convex regularizers according to the data structure (such as total variation regularizer for image data [15]) and keep this problem convex. In Chapter 2, we will demonstrate that the performance of matrix completion on a particular real-world data set can be improved by adding specially designed convex regularizers. The content of Chapter 2 is from my previous paper *Learning to Predict Human Stress Level with Incomplete Sensor Data from Wearable Devices*.

Meanwhile, LRMF could be efficiently solved with singular value thresholding [11]. One of the most famous variants of LRMF is the non-negative matrix factorization (NMF), where the entries of both  $A$  and  $B$  are restricted to be non-negative [10].

Both NNM and LRMF could be raised to tensor settings with properly defined tensor ranks. In Chapter 3, we will introduce the corresponding version of NNM and LRMF in a tensor setting (tensors are higher-dimensional matrices; see Chapter 3 for precise definition). We will also propose a simple but efficient weighted higher-order singular value decomposition (HOSVD) algorithm for recovering the tensor data. The content of Chapter 3 is from my previous paper The content of this chapter is from my previous paper *Tensor Completion through Total Variation with Initialization from Weighted HOSVD*.

In Chapter 4, we will introduce the robust principal component analysis in the tensor setting. The principal component analysis (PCA) is a widely used method for dimension reduction tasks. We will then propose a fast non-convex algorithm for tensor robust principal component analysis (TRPCA) problems.

In Chapter 5, we shift the topic from low-rank tensors to more general machine learning research: we study tweets posted by media outlets and provide a quantitative framework to infer the political bias and source quality of media outlets from text. We demonstrate several machine-learning methods, such as a naive-Bayes method, a support-vector machine (SVM), and an LSTM network, to infer the bias and quality values for each tweet.

## CHAPTER 2

### Matrix Completion Under Low-rank Assumption

The content of this chapter is from my previous paper *Learning to Predict Human Stress Level with Incomplete Sensor Data from Wearable Devices*. This paper has been accepted by the 28th ACM International Conference on Information and Knowledge Management (CIKM 2019). The Co-authors are Jyun-Yu Jiang, Andrea Bertozzi, Wei Wang, Sean Young, and Deanna Needell. I conducted the experiment and wrote the manuscript related to data completion. Jyun-Yu conducted the experiment and wrote the manuscript related to the temporally hierarchical attention network (THAN). Andrea and Sean formulated and developed the project and edited the manuscript. Sean also provided the data source. Wei and Deanna advised on the project.

#### 2.1 Motivation

Since the tensor is a high-dimensional extension of a matrix, we start with tasks on matrix data in the first section. We will first introduce the matrix completion task under the low-rank assumption and then demonstrate a real-world application on Fitbit data based on the matrix completion task. This project incorporates the matrix completion method and one deep network called the temporally hierarchical attention network (THAN) to predict stress levels with Fitbit data.

For many regression and classification tasks, when data appears as a matrix with missing values, researchers would require the completed matrix instead of dropping all rows with missing data [16]. Besides the most basic strategy, data padding and data interpolation, recent studies proposed various matrix completion methods for different

objectives. For example, maximum likelihood estimation (MLE) is a useful tool to recover the matrix with sparse missing values or errors [17]. Matrix Factorization (MF) [18], along with its non-negative version, NMF [19], is another popular method that generates a low-rank matrix from sparse observations. The drawback of MF and NMF is that the algorithm will not fix the original data but only minimize the difference between the original matrix and the completed matrix. Alternatively, nuclear norm minimization (NNM), inspired by the same assumption that the matrix has a low-rank structure, will keep the existing data unchanged. Instead of forcing the rank of the completed matrix to be some fixed number  $r$ , NNM minimizes a convex relaxation of matrix rank. In [20], researchers proved that given the underlying matrix is low-rank and the observed entries are sampled uniformly at random, one can achieve exact recovery with high probability under mild additional assumptions by using NNM.

Based on NNM, the performance of matrix completion could be further improved by adding a specific regularizer. This regularizer is usually from the knowledge of some known pattern of the given matrix. For example, if we are recovering an image with a piece-wise smooth pattern, adding the total variation regularizer can improve the performance of the task [21]. If one assumes entries with smaller absolute values are more likely to be missing, one can then add a regularization term for unobserved entries that penalize the size of the missing values [15].

### **2.1.1 Stress Level Prediction**

Traditional methods for monitoring stress levels are mainly from physiological signals such as heart rate, blood pressure, and body temperature [22]. By applying machine learning techniques, researchers can predict human stress and emotion by measuring their physiological signals. A study has shown that, with the specially designed sensor recording body movement, speech volume, and pitch, a person-specific model gives 93% accuracy on binary stress prediction [23]. Personal-specific models significantly outperform general models [24, 25, 26] regardless of the machine learning model. Al-

ternatively, the lack of personal history information motivates us to develop general models that treat all input instances in the same way.

Physiological signals also have limitations: the necessity of continuous signal measurement and the data noise restrict its broad application. Meanwhile, researchers have analyzed stress and mood predictors from broadly accessible data such as smartphone usage [27] or general wearable device data such as geographic location [24] and activities. There are many studies about predicting health-related indexes with large-scale accessible datasets, integrating both public health information and data science. For example, an existing study shows a significant association between negative sentiment tweets and stress level [28], inferring the possibility of monitoring people’s mood and stress using social media data. Several studies [25, 26] showed the potential to use wearable sensors and other integrated data sources to monitor people’s stress levels through a machine learning model.

In recent years, different machine learning models have been applied to predict stress and other similar health indexes. Multi-linear regression (MLR) and support vector machine (SVM) perform well when the data appears to be linearly separable; these methods have been applied to phone call/email usage and mobility metrics to predict mood and depression [27, 24]. Random forest (RF) and adaptive boosting tree (AB)[29, 30], classifiers with a decision tree structure, are good at detecting impactful signals and have outperformed several other baselines when predicting binary stress labels from mobile phone usage and weather[26]. The  $k$ -Nearest Neighbors (KNN) graph is another non-neural network classifier used in several diagnostic applications [31, 32, 23]. In the case of Artificial Neural Networks (ANN), adding a task-specific layer for each person could significantly improve the prediction performance [25].

## 2.2 Problem Formulation and Proposed Approach

We first formally define the objectives of predicting human stress levels with sparse and incomplete sensor data. Let  $\mathbf{U}$  and  $\mathbf{S}$  be the set of  $m$  users and the set of activities

that can be recorded by wearable devices. For each activity  $s \in \mathbf{S}$ , let  $M^s \in \mathbb{R}^{m \times n}$  denotes the sensor data that should be collected by wearable devices for  $n$  hours. Each entry  $M_{ij}^s \in M^s$  denotes the percentage of time (0–100) doing the activity  $s$  in the  $j$ -th hour for the  $i$ -th user  $u_i \in \mathbf{U}$ . Note that entries in  $M^s$  can be missing and unavailable because users may not always wear the devices. Here the set of recorded hours  $\Omega$  for all users can be defined as:

$$\Omega = \{(i, j) \mid M_{ij}^s \text{ exists}\}. \quad (2.1)$$

The recorded data can then be described as a linear mapping  $P_\Omega(M^s) : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}$ , where

$$[P_\Omega(M^s)]_{ij} = \begin{cases} M_{ij}^s, & (i, j) \in \Omega \\ 0, & (i, j) \notin \Omega. \end{cases} \quad (2.2)$$

Each user  $u_i \in \mathbf{U}$  also has a set of features  $\mathbf{r}_{u_i}$  to indicate personal characteristics. The two goals of this project are listed as follows:

1. **Sensor Data Completion:** Given the recorded data  $P_\Omega(M^s)$ , the first goal is to recover the accurate sensor data  $M^s$ . More precisely, for each activity  $s$ , we aim to infer a matrix  $X^s$  so that  $X^s$  can be as similar to  $M^s$  as possible.
2. **Stress Level Prediction:** For each user  $u_i \in \mathbf{U}$ , the next goal is to predict the stress level  $c_{u_i} \in \mathbf{C}$  of the user with the recovered sensor data  $X^s$ , where  $\mathbf{C}$  is a predefined sets of stress levels. Specifically, a predicted level  $\hat{c}_{u_i} \in \mathbf{C}$  is inferred for the user  $u_i$  so that  $\hat{c}_{u_i}$  can be likely to be  $c_{u_i}$ . In this project, we consider both binary-class and 5-class stress level prediction due to the nature of collected data (described in Section 2.3).

### 2.2.1 Data Completion with Diurnal Regularizers

To recover the sensor data, we follow the low-rank assumption [18] to define a minimization problem that can be approximately solved by the alternating direction method of

multipliers (ADMM) [33, 21].

**Nuclear Norm Minimization.** Based on the low-rank assumption, the task of recovering  $M^s$  with a low rank can be formally defined as the following optimization problem:

$$\begin{aligned} X^s &= \underset{X}{\operatorname{argmin}} \operatorname{rank}(X) \\ \text{s.t. } & P_\Omega(X) = P_\Omega(M^s). \end{aligned} \tag{2.3}$$

Unfortunately, the problem has been proven to be NP-hard [34]. Therefore, we follow [35] to apply the nuclear norm  $\|X\|_*$  as a substitution and convert the original problem into a convex optimization problem of nuclear norm minimization (NNM):

$$\begin{aligned} X^s &= \underset{X}{\operatorname{argmin}} \|X\|_* \\ \text{s.t. } & P_\Omega(X) = P_\Omega(M^s). \end{aligned} \tag{2.4}$$

Here we denote  $\|X\|_* = \sum_{k=1}^n \sigma_k(X)$  as the nuclear norm of a matrix  $X$ , where  $\sigma_k(X)$  is the  $k$ -th singular value of  $X$ . The nuclear norm is connected with matrix rank since matrix rank can be interpreted as the number of non-zero singular values.

**Daily and Weekly Regularizers.** In addition to solving NNM, previous studies [21, 15] have demonstrated that the performance of matrix completion can be improved by specific regularizations according to the data structure. Since the sensor data are generated as a temporal sequence, we regularize the differences of each activity in the same hour across different days and different weeks. Note that the regularizers should be convex so that the optimization problem can have a proper unique solution for further NNM optimization.

We first assume that people should have relatively regular diurnal activity patterns (measured in hours). Hence, a diurnal penalizer can regularize diurnal patterns as:

$$D_{i,j}^d(X) = \begin{cases} X_{i,j+24} - X_{i,j}, & 1 \leq j \leq n - 24, \\ 0, & j > n - 24. \end{cases} \tag{2.5}$$

Similar, weekly patterns can also imply a weekly penalizer as:

$$D_{i,j}^w(X) = \begin{cases} X_{i,j+168} - X_{i,j}, & 1 \leq j \leq n - 168, \\ 0, & j > n - 168. \end{cases} \quad (2.6)$$

The regularizer based on two patterns can then be computed as:

$$\|X\|_{Reg} = \gamma_1 \sum_{i,j} (D_{i,j}^d(X))^2 + \gamma_2 \sum_{i,j} (D_{i,j}^w(X))^2, \quad (2.7)$$

where  $\gamma_1$  and  $\gamma_2$  control the importance of two penalizers. Finally, the minimization problem becomes

$$\begin{aligned} X^s &= \underset{X}{\operatorname{argmin}} (\|X\|_* + \|X\|_{Reg}) \\ \text{s.t. } & P_\Omega(X) = P_\Omega(M^s). \end{aligned} \quad (2.8)$$

**Optimization with ADMM.** ADMM [33, 21] is a general algorithm to solve convex optimization problems with the following form:

$$\begin{aligned} &\underset{x_1, x_2}{\operatorname{argmin}} f_1(x_1) + f_2(x_2) \\ \text{s.t. } & A_1 x_1 + A_2 x_2 = b, \end{aligned} \quad (2.9)$$

where  $f_1$  and  $f_2$  are convex functions of  $x_1$  and  $x_2$ , respectively, and  $A_1$  and  $A_2$  are linear constraints. The idea is to break down the expression into smaller convex pieces and update  $x_1$  and  $x_2$  separately so that each step becomes easier to solve. To solve Eq. (2.8) through ADMM, we first follow [21] to convert the regularizer shown in Eq. (2.7) into the following form:

$$\|X\|_{Reg} = \gamma_1 \|X - X\phi_1\|_F^2 + \gamma_2 \|X - X\phi_2\|_F^2, \quad (2.10)$$



where  $\phi_1$  and  $\phi_2$  are two auxiliary matrices defined as.

$$\phi_1 = \begin{bmatrix} O_{24,(n-24)} & O_{n-24,24} \\ I_{n-24} & I_{24} \end{bmatrix}, \quad \phi_2 = \begin{bmatrix} O_{168,(n-168)} & O_{n-168,168} \\ I_{n-168} & I_{168} \end{bmatrix},$$

where  $O_{i,j}$  is an  $i \times j$  zero sub-matrix, and  $I_k$  is an identity sub-matrix of size  $k$ . Therefore, our minimization problem (2.8) can be reformulated as:

$$\begin{aligned} X^s &= \underset{X}{\operatorname{argmin}} (\|X\|_* + \gamma_1 \|X - X\phi_1\|_F^2 + \gamma_2 \|X - X\phi_2\|_F^2) \\ \text{s.t. } &P_\Omega(X) = P_\Omega(M^s). \end{aligned}$$

To fit our optimization problem into ADMM, we introduce an auxiliary variable  $Y$  such that  $P_{\Omega^C}(Y) = P_{\Omega^C}(X)$ , where  $\Omega^C$  is the complement set of the observation set  $\Omega$ . To optimize Eq. (2.2.1), two auxiliary matrices can then be computed as:

$$\begin{aligned} X_D^s &= \underset{X,Y}{\operatorname{argmin}} \|X\|_* + \gamma_1 \|Y - Y\phi_1\|_F^2 \\ \text{s.t. } &P_\Omega(X) = P_\Omega(M) \text{ and } P_{\Omega^C}(X) = P_{\Omega^C}(Y) \end{aligned} \tag{2.11}$$

$$\begin{aligned} X_W^s &= \underset{X,Y}{\operatorname{argmin}} \|X\|_* + \gamma_2 \|Y - Y\phi_2\|_F^2 \\ \text{s.t. } &P_\Omega(X) = P_\Omega(M) \text{ and } P_{\Omega^C}(X) = P_{\Omega^C}(Y). \end{aligned} \tag{2.12}$$

Therefore, the optimization problem now has the general form of ADMM as shown in Eq (2.9). More precisely,  $f_1(X) = \|X\|_*$  is a convex functions on  $X$ ;  $f_2(Y) = \|Y - Y\phi_1\|_F^2$  is a convex functions on  $Y$ ;  $P_\Omega(X) = P_\Omega(M)$  and  $P_{\Omega^C}(X) = P_{\Omega^C}(Y)$  are the two linear constraints. Based on ADMM, the augmented Lagrangian of Eq. (2.11) can be derived as:

$$\begin{aligned} L_t(X_D, Y_D, Z_D) &= \|X_D\|_* + \gamma_1 \|Y_D - Y_D\phi_1\|_F^2 \\ &\quad + Z_D^T(X_D - Y_D) + \frac{t}{2} \|X_D - Y_D\|_F^2, \end{aligned}$$

where  $Z$  is the dual variable and  $t > 0$  is the parameter for quadratic penalty . At

iteration  $k$ , ADMM first minimizes  $L_t$  over  $x_1$  while fixing  $x_2$  and  $z$ .  $X_D^{s,k+1}$ ,  $Y_D^{s,k+1}$ , and  $Z_D^{s,k+1}$  can then be respectively updated with three sub-problems as follows:

$$X_D^{s,k+1} = \underset{X}{\operatorname{argmin}} L_t(X, Y_D^{s,k}, Z_D^{s,k}), \quad (2.13)$$

$$Y_D^{s,k+1} = \underset{Y}{\operatorname{argmin}} L_t(X_D^{s,k+1}, Y, Z_D^{s,k}), \quad (2.14)$$

$$Z_D^{s,k+1} = Z_D^{s,k} + t(Y_D^{s,k+1} - X_D^{s,k+1}). \quad (2.15)$$

The sub-problem (2.13) can be solved by singular value shrinkage [11] as:

$$X_D^{s,k+1} = \mathcal{D}_{(1-\gamma_1)/t}(Y_D^k + \frac{1}{t}Z_D^k), \quad (2.16)$$

where the soft-thresholding operator  $\mathcal{D}$  for singular value shrinkage is defined as:

$$\mathcal{D}_\tau(X) = U \mathcal{D}_\tau(\Sigma) V^*, \quad \mathcal{D}_\tau(\Sigma) = \operatorname{diag}(\{\sigma_i - \tau\}_+). \quad (2.17)$$

Since  $L_t$  is convex, the sub-problem (2.14) can be solved by computing  $\partial L_t / \partial Y$  and setting it to be zero:

$$Y_D^{k+1} = (\lambda X_D^{s,k+1} - Z_D^{s,k}) [2\gamma_1(I - \phi_1 - \phi_1^T + \phi_1^T \phi_1) + tI]^{-1}. \quad (2.18)$$

The sub-problem 2.15 can directly be updated with the solutions of Eq. 2.13 and 2.14. The optimization process iteratively solves sub-problems and stops when  $X_D^{s,k}$  converges to  $X_D^s$ . Similarly, Eq. (2.12) for obtaining  $X_W^s$  can also be optimized with the same approach. Finally, we complete the sensor data matrix by  $X^s = (X_D^s + X_W^s)/2$ .

### 2.2.2 THAN for Stress Level Prediction

To infer human stress level, we propose THAN to exploit temporally hierarchical structures to model behaviors. To ease the discussion, we focus on the two-level temporal structure, including daily- and hourly-level, while the depth of the hierarchical structure

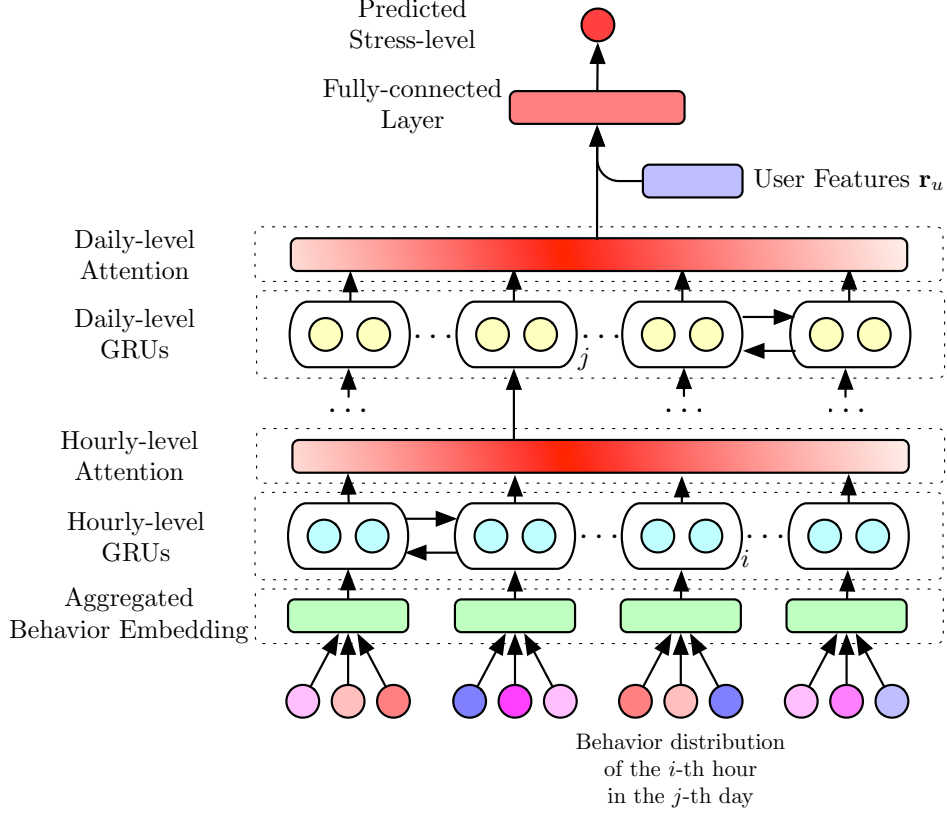


Figure 2.1: The overall schema of THAN.

can be simply expanded to monthly-level or replaced with other temporally hierarchical structures. Note that although the hierarchical structures have been applied in natural language processing [36, 37], this thesis is the first work to apply the idea in stress level prediction.

Figure 2.1 shows the overall schema of the proposed THAN for stress level prediction. Each activity  $s$  has a trainable embedding vector so that user behaviors in each hour can be represented by an aggregated behavior embedding vector with the recovered sensor data  $M^s$ . The hourly-level recurrent neural network (RNN) models user behaviors in each day with hourly-level attention while the daily-level RNN derives an overall sensor representation with daily-level attention. Finally, the sensor representation and user features can be applied to predict the stress level with a fully-connected hidden layer. More details are explained as follows.

**Aggregated Behavior Embedding.** Suppose we are inferring the stress level of the

user  $u \in \mathbf{U}$ . Since wearable devices can only provide aggregated activity information for sensor data, we propose aggregated behavior embedding to represent user behaviors in a latent space. For each activity  $s \in \mathbf{S}$ , we learn a  $d_s$ -dimensional behavior embedding  $\mathbf{e}_s$ . For the  $j$ -th hour of the  $i$ -th day, the aggregated behavior embedding  $\mathbf{a}_{i,j}$  is defined as:

$$\mathbf{a}_{i,j} = \sum_{s \in \mathbf{S}} X_{u,i,j}^s \cdot \mathbf{e}_s, \quad (2.19)$$

where  $X_{u,i,j}^s$  is the recovered sensor data about the activity  $s$  in the  $j$ -th hour of the  $i$ -th day for the user  $u$ .

**Hourly-level RNN.** THAN follows a bottom-up approach for computations with bidirectional recurrent neural networks (Bi-RNNs) with attention [38]. To encode the behaviors in the  $i$ -th day, a Bi-RNN scans the sequences of corresponding aggregated behavior embeddings during both forward and backward passes. In the forward pass, the Bi-RNN generates a sequence of hidden states  $[\overrightarrow{\mathbf{h}}_{i,1}, \overrightarrow{\mathbf{h}}_{i,2}, \dots, \overrightarrow{\mathbf{h}}_{i,24}]$ , where  $\overrightarrow{\mathbf{h}}_{i,j} = \text{RNN}(\overrightarrow{\mathbf{h}}_{i,j-1}, \mathbf{a}_{i,j})$  is a  $d_g$ -dimensional hidden states generated by a dynamic function such as long short-term memory (LSTM) [39] or gated recurrent units (GRU) [40]. Here we use GRU instead of LSTM because it requires fewer parameters [41]. The backward pass then processes the sequence reversely and derives the backward hidden states  $[\overleftarrow{\mathbf{h}}_{i,1}, \overleftarrow{\mathbf{h}}_{i,2}, \dots, \overleftarrow{\mathbf{h}}_{i,24}]$ , where  $\overleftarrow{\mathbf{h}}_{i,j} = \text{RNN}(\overleftarrow{\mathbf{h}}_{i,j+1}, \mathbf{a}_{i,j})$ . The forward and backward hidden states are then concatenated as hidden representations for hours in the day as follows:  $[\mathbf{h}_{i,1}, \mathbf{h}_{i,2}, \dots, \mathbf{h}_{i,24}]$ , where  $\mathbf{h}_{i,j} = [\overrightarrow{\mathbf{h}}_{i,j}; \overleftarrow{\mathbf{h}}_{i,j}]$ .

To estimate the importance of each hour in the day, the attention mechanism [38] is applied to extract and aggregate important hidden representations. More precisely, the importance  $\alpha_{i,j}$  of  $\mathbf{h}_{i,j}$  can be estimated as:

$$\alpha_{i,j} = \frac{\exp(\mathbf{z}_{i,j} \cdot \mathbf{z}^H)}{\sum_{j'} \exp(\mathbf{z}_{i,j'} \cdot \mathbf{z}^H)}, \quad (2.20)$$

where  $\mathbf{z}_{i,j} = \tanh(\mathcal{F}^H(\mathbf{h}_{i,j}))$ ;  $\mathcal{F}^H(\cdot)$  is a fully-connected layer [42];  $\tanh$  is the activation for computing similarity; and  $\mathbf{z}^H$  is the context vector to measure the importance

of each hour. Finally, the representation of the  $i$ -th day can be represented as the weighted sum of the hidden representations as follows:

$$\mathbf{a}_i = \sum_j \alpha_{i,j} \cdot \mathbf{h}_{i,j}. \quad (2.21)$$

**Daily-level RNN.** With the daily representations, the Bi-RNN is applied again to derive the overall representation for stress level prediction. The sequences of forward and backward hidden states can be generated as  $[\vec{\mathbf{h}}_1, \vec{\mathbf{h}}_2, \dots, \vec{\mathbf{h}}_L]$  and  $[\overleftarrow{\mathbf{h}}_1, \overleftarrow{\mathbf{h}}_2, \dots, \overleftarrow{\mathbf{h}}_L]$ , where  $L$  is the number of days for sensor data.  $\vec{\mathbf{h}}_i = \text{RNN}(\vec{\mathbf{h}}_{i-1}, \mathbf{a}_i)$  and  $\overleftarrow{\mathbf{h}}_i = \text{RNN}(\overleftarrow{\mathbf{h}}_{i+1}, \mathbf{a}_i)$ . The hidden representations for all days can then be represented as  $[\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_L]$ , where  $\mathbf{h}_i = [\vec{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]$ .

The importance of each day  $\alpha_i$  in sensor day can also be estimated by the attention mechanism as follows:

$$\alpha_i = \frac{\exp(\mathbf{z}_i \cdot \mathbf{z}^D)}{\sum_{i'} \exp(\mathbf{z}_{i'} \cdot \mathbf{z}^D)}, \quad (2.22)$$

where  $\mathbf{z}_{i,j} = \tanh(\mathcal{F}(\mathbf{h}_{i,j}))$ ;  $\mathcal{F}(\cdot)$  is a fully-connected layer

To estimate the importance of each hour in the day, the attention mechanism [38] is applied to extract and aggregate important hidden representations. More precisely, the importance  $\alpha_{i,j}$  of  $\mathbf{h}_{i,j}$  can be estimated as:

$$\alpha_{i,j} = \frac{\exp(\mathbf{z}_{i,j} \cdot \mathbf{z}^H)}{\sum_{j'} \exp(\mathbf{z}_{i,j'} \cdot \mathbf{z}^H)}, \quad (2.23)$$

where  $\mathbf{z}_{i,j} = \tanh(\mathcal{F}^D(\mathbf{h}_{i,j}))$ ;  $\mathcal{F}^D(\cdot)$  is a fully-connected layer;  $\mathbf{z}^D$  is the context vector to measure the importance of each day. Finally, the overall representation of sensor data can be derived as:

$$\mathbf{a} = \sum_i \alpha_i \cdot \mathbf{h}_i. \quad (2.24)$$

**Stress Level Prediction.** The representation of sensor data  $\mathbf{a}$  presents the user behaviors while the user features  $\mathbf{r}_u$  provide the aspects of user characteristics. To precisely infer the stress level of the user, we apply a fully-connected hidden layer to

combine the knowledge of two resources:

$$\hat{c}_u = \underset{c}{\operatorname{argmax}} \mathcal{F}^c (\mathcal{F}^P([\mathbf{a}; \mathbf{r}_u])), \quad (2.25)$$

where  $\mathcal{F}^P(\cdot)$  is a fully-connected layer with  $d_p$  hidden neurons, and  $\mathcal{F}^c(\cdot)$  computes the logit of the stress level  $c \in \mathcal{C}$  for classification.

**Learning and Optimization.** The task of stress level prediction can be modeled as a classification problem. Here we adopt the categorical cross-entropy [43] as the loss function for optimization. More precisely, the loss function for each training example can be written as

$$- \sum_c \mathbf{1}(c = c_u) \log (\mathcal{F}^c (\mathcal{F}^P([\mathbf{a}; \mathbf{r}_u]))), \quad (2.26)$$

where  $\mathbf{1}(\cdot)$  is an indicator function;  $c_u$  is the ground truth of the stress level.

**User Features.** Any user-related information can be represented in the user feature vectors to potentially boost the prediction performance. In this thesis, we adopt historical stress levels in the previous 11 weeks as user features. The mean and variance values of historical stress levels are also considered. Finally, there are 13 user features in the experiments. Note that we also evaluate the performance without using user features to demonstrate the effectiveness of both the proposed model and user features.

## 2.3 Experiments

We conduct extensive experiments and in-depth analysis to evaluate the proposed data completion with diurnal regularizer (DCDR) and THAN.

**Data Collection.** To obtain the datasets, we filtered data from a longitudinal study among college students, designed to collect Twitter, survey, and wearable device data for 12 weeks from September 28, 2015, to December 20, 2015.

The students were asked to fill in an online survey and report their stress levels. There are five options: 1 (Not at all), 2 (Low), 3 (Average), 4 (High), and 5 (Extremely

high). For the purpose of this paper, we view the survey data as “ground truth”. Each of the students was invited/requested to wear a smart wristband to record hourly aggregated information of activities, including *sleep*, *walk*, *run*, *bike*, and *moderate activities*, as the collected sensor data.

Incomplete data should at least have basic structural information, so we filtered out students with fewer or equal to two surveys taken and fewer or equal to 2 hours’ sleep records. This leaves 75 students and only 52% entries of the sensor data observed. Finally, we have 855 records of weekly reported stress levels with incomplete sensor data. Note that the number is not  $75 \times 12$  because some students did not turn in the survey every week, so some ground truth data is missing.

**Training Data for Algorithms.** For the task of sensor data completion, we uniformly randomly sample 90% of observed entries for training and evaluate DCDR with the remaining observed entries. For the task of stress level inference, we treat each week for each student as an instance with the corresponding sensor data and the stress level indicated in the survey. We then randomly sample 80% of instances as training data and evaluate THAN with the remaining instances. Finally, we have 7,0786 and 7,865 training and testing entries for sensor data completion, while 684 and 171 records are applied for training and testing in stress level inference. For the set of stress levels  $\mathcal{C}$ , we attempt two different partitions, including binary-class and 5-class stress levels. In the binary-class setting, we consider the students who reported stress levels greater than 3 as stressed students.

**Implementation Details.** We implement DCDR in Matlab while THAN is implemented in Tensorflow [44]. The Adam optimizer [45] is applied to optimize THAN with a  $10^{-4}$  initial learning rate. After parameter tuning,  $\gamma_1$  and  $\gamma_2$  in DCDR are set as 0.2154 and 0.0774. The number of dimensions for behavior embeddings  $d_s$  in THAN is 4 while the numbers of dimensions for GRU hidden states and context vectors of attention mechanism are 64. The number of hidden neurons in the fully-connected layer is 128.

**Evaluation Measurements.** For sensor data completion, we treat the problem as a regression task and consider Root Mean Squared Error (RMSE) and Mean Average Error (MAE) as the evaluation measurements. In addition, we also use a threshold 0.5 to convert the task into a classification problem while accuracy, precision, recall, and F1 score can be applied for evaluation:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}, \quad (2.27)$$

$$\text{Precision} = \frac{TP}{TP + FP}, \quad (2.28)$$

$$\text{Recall} = \frac{TP}{TP + FN}, \quad (2.29)$$

$$\text{F1 score} = \frac{2 * (\text{Recall} * \text{Precision})}{\text{Recall} + \text{Precision}}. \quad (2.30)$$

Here,  $TP, FP, TN$ , and  $FN$  stand for true positive, false positive, true negative, and false negative, respectively. Stress level prediction is a classification task, so accuracy, precision, recall, and F1 score are considered evaluation measurements.

### 2.3.1 Sensor Data Completion

To evaluate the performance of DCDR, we compare with four conventional approaches for data completion, including interpolation (ITP) [46], matrix factorization (MF) [47], non-negative matrix (NMF) [19], and NNM [33, 21]. Note that NNM is a special case of DCDR at  $\gamma_1 = \gamma_2 = 0$ . Table 2.1 shows the performance of five methods in the task of sensor data completion for two activities. For the *Sleep* activity, baseline methods have similar and good performance in both regression and classification tasks because the low-rank assumption and data redundancy are leveraged. Our proposed DCDR outperforms all of the baseline methods. This is because the regularizers capture the patterns of user behaviors instead of only modeling sensor data as a low-rank matrix. For the *Walk* activity, it is interesting that all of the methods perform worse because people usually have diurnal sleeping patterns while the walking behaviors are more



Table 2.1: The sensor data completion performance of five methods for the activities *Sleep* and *Walk*. All improvements of DCDR against baseline methods are significant at 99% level in a paired *t*-test.

Method	<i>Sleep</i>						<i>Walk</i>					
	RMSE	MAE	Accuracy	Precision	Recall	F1	RMSE	MAE	Accuracy	Precision	Recall	F1
ITP [46]	0.2957	0.1975	0.8488	0.7249	0.7481	0.7363	0.1027	0.0613	0.6544	0.7395	0.4072	0.5252
MF [47]	0.3011	0.2010	0.8568	0.6964	0.7873	0.7390	0.1111	<b>0.0504</b>	0.7439	0.0191	0.6618	0.0371
NMF [19]	0.3053	0.2055	0.8514	0.6796	0.7815	0.7270	0.1113	<b>0.0504</b>	<b>0.7996</b>	<b>0.7432</b>	0.0127	0.0250
ADMM [33]	0.3096	0.2089	0.8433	0.5880	<b>0.8472</b>	0.6942	<b>0.1025</b>	0.0552	0.7202	0.5841	0.4301	0.4954
DCDR	<b>0.2861</b>	<b>0.1754</b>	<b>0.8671</b>	<b>0.7464</b>	0.7864	<b>0.7659</b>	0.1039	0.0573	0.7200	0.5999	<b>0.4676</b>	<b>0.5256</b>

random and irregular. This is also the reason that THAN does not improve ADMM by adding regularizers.

### 2.3.2 Stress Level Prediction

To evaluate the performance of THAN, we compare with seven baseline methods introduced in section 2.1.1, including decision tree (DT) [48], random forest (RF) [26], Adaboost (AB) [29, 30], *k*-nearest neighbor (KNN) [31], support vector machine (SVM) [26], multi-linear regression (MLR) [27], and artificial neural network (ANN) [49]. Table 2.2 and 2.3 indicate the performance of eight methods in stress level prediction with binary-class and 5-class stress levels. For the baseline methods, RF performs the best because of its robustness with limited training data. The complicated methods, such as SVM and ANN, perform worse because more training data are needed without considering temporally structural information. Our proposed THAN outperforms all of the baseline methods in both binary-class and 5-class stress level prediction. This is because the temporally hierarchical structures can precisely capture the structural patterns of user behaviors while the attention mechanism is capable of effectively extracting essential information for predicting human stress levels.

## 2.4 Analysis and Discussions

We conduct ablation studies and parameter sensitivity analyses to demonstrate the robustness and effectiveness of our proposed approach.

Table 2.2: The stress level prediction performance of eight methods with binary-class stress levels. All improvements of THAN against baseline methods are significant at 99% level in a paired  $t$ -test.

Method	Accuracy	Precision	Recall	F1
DT [48]	0.6433	0.6111	0.5714	0.5906
RF [26]	0.7076	<b>0.7288</b>	0.5584	0.6324
AB [29, 30]	0.6784	0.6774	0.5455	0.6043
KNN [31]	0.5789	0.5352	0.4935	0.5135
SVM [26]	0.6491	0.6735	0.4286	0.5238
MLR [27]	0.5965	0.5541	0.5325	0.5430
ANN [49]	0.6608	0.6727	0.4805	0.5606
THAN	<b>0.7661</b>	0.6796	<b>0.9091</b>	<b>0.7778</b>

Table 2.3: The stress level prediction performance of eight methods with 5-class stress levels. All improvements of THAN against baseline methods are significant at 99% level in a paired  $t$ -test.

Method	Accuracy	Micro-F1	Macro-F1
DT [48]	0.2982	0.2982	0.2747
RF [26]	0.4561	0.4561	0.4631
AB [29, 30]	0.4737	0.4737	0.4389
KNN [31]	0.2982	0.2982	0.2568
SVM [26]	0.3216	0.3216	0.1861
MLR [27]	0.4737	0.4737	0.4389
ANN [49]	0.3509	0.3509	0.3013
THAN	<b>0.4757</b>	<b>0.4757</b>	<b>0.4706</b>

**Effectiveness of Sensor Data Completion.** We first verify the effectiveness of recovered sensor data for predicting stress levels. Figure 2.2 illustrates the F1 scores of eight methods in stress level prediction with and without sensor data completion. Almost all of the methods could be improved with sensor data completion. The results show that although about 50% of sensor data are unobserved, DCDR can still appropriately recover missing data, thereby benefiting the downstream applications.

**Depth of Temporally Hierarchical Structures.** One of the contributions in THAN is to consider temporally hierarchical structures, so here we verify the effectiveness of this idea. Table 2.4 shows the performance of THAN in stress level prediction with different depths of temporally hierarchical structures. Note that depth-1 THAN degenerates to a sequential RNN with attention without considering any hierarchical

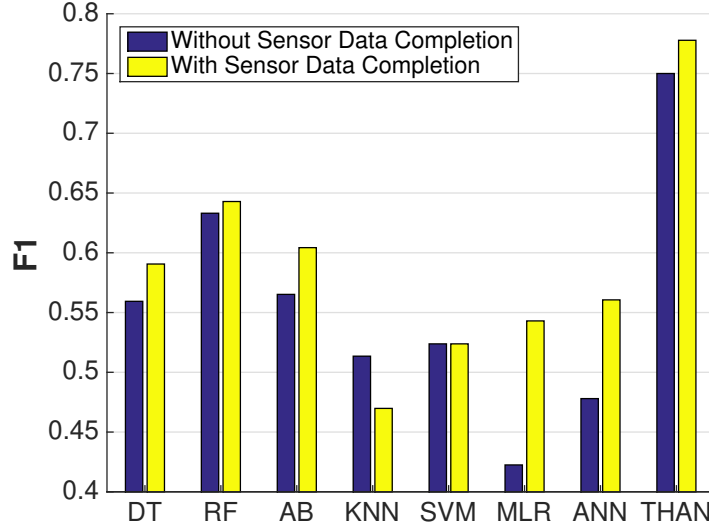


Figure 2.2: The F1 scores of eight methods in stress level prediction with and without sensor data completion.

Table 2.4: The stress level prediction performance of THAN with different depths of temporally hierarchical structures.

Depth	Acc	Prec	Rec	F1
1 (Hourly RNNs)	0.7485	0.6771	0.8442	0.7514
2 (Hourly & Daily RNNs)	<b>0.7661</b>	<b>0.6796</b>	<b>0.9091</b>	<b>0.7778</b>

structure. More specifically, depth-1 THAN only utilizes hourly RNNs when depth-2 THAN exploits both hourly and daily RNNs based on hierarchical structures. The results show that the stress level prediction performance can be significantly boosted after considering temporally hierarchical structures. It further demonstrates that the temporally structural information in sensor data is much beneficial for understanding human stress.

**Effectiveness of User Features.** We propose a set of personal user features for THAN to capture individual characteristics for predicting stress levels more precisely. Here we conduct an ablation study to validate the effectiveness of our proposed user features. Table 2.5 shows the performance of THAN in stress level prediction with different features. Note that S and U in Table 2.5 represent the usage of sensor data and user features. The results demonstrate that the proposed user features are effective because the historical and statistical features can directly reflect the stress level

Table 2.5: The stress level prediction performance of THAN with different features. S denotes the sensor data while U represents the user features.

Features	Accuracy	Precision	Recall	F1
S Only	0.6140	0.5902	0.4675	0.5217
U Only	0.7485	0.7361	0.6883	0.7114
S+U	<b>0.7661</b>	<b>0.6796</b>	<b>0.9091</b>	<b>0.7778</b>

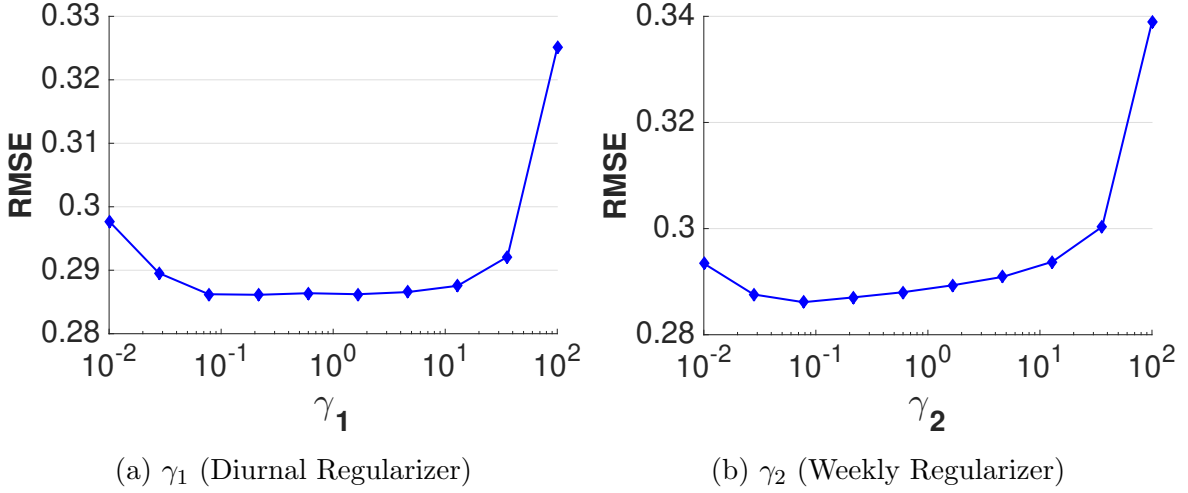


Figure 2.3: The RMSE scores of DCDR in sensor data completion with different weights of the parameters  $\gamma_1$  and  $\gamma_2$ .

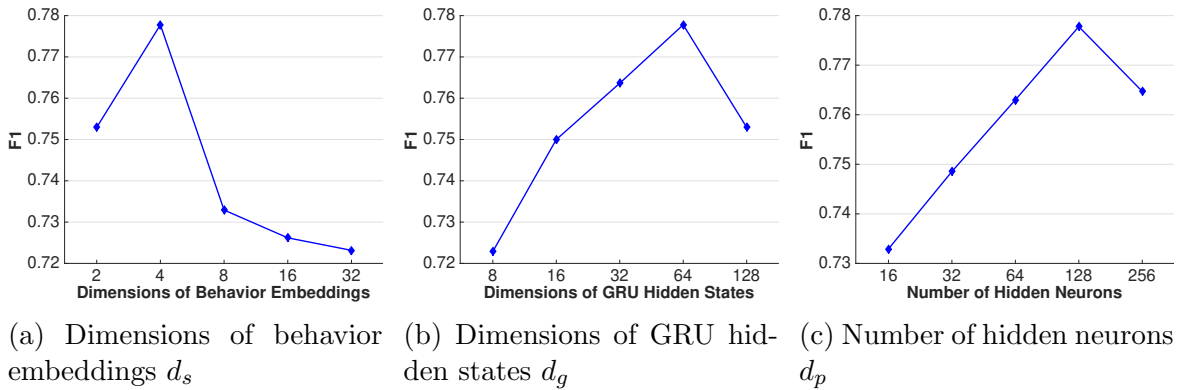


Figure 2.4: The F1 scores of THAN in stress level prediction across different parameters.

distribution for a specific user. Without personalized information, THAN with only sensor data can still achieve good performance. It indicates that sensor data and user features all are valuable for stress level prediction so that THAN can achieve the best prediction performance by simultaneously considering both features.

**Regularizers in DCDR.** As shown in Section 2.3.2, regularizers play an important role in sensor data completion, so we conduct a parameter sensitivity analysis for the regularization weights of two regularizers. Figure 2.3 illustrates the RMSE scores of DCDR in sensor data completion with different weights of  $\gamma_1$  and  $\gamma_2$  for diurnal and weekly regularizers. The results show that the regularization weights cannot be too small or too large for satisfactory data completion, so parameter tuning is essential. Moreover, the best parameters of  $\gamma_1$  and  $\gamma_2$  are 0.2154 and 0.0774. It indicates that the diurnal patterns are more effective than the weekly patterns for stress level prediction.

**Parameter Sensitivity of THAN.** We also conduct the parameter sensitivity analysis for three parameters of THAN, including the numbers of dimensions for behavior embeddings  $d_s$ , dimensions for GRU hidden states  $d_g$ , and hidden neurons in the last fully-connected layer  $d_p$ . Figure 2.4 shows the F1 scores of THAN in stress level prediction across different parameters. For behavior embeddings, it is interesting that THAN does not favor a large dimension number  $d_s$ . This can be because the number of available activities is limited, so it does not require a complicated latent space to represent each activity. For the number of dimensions for GRU hidden states  $d_g$  and the number of hidden neurons  $d_p$ , greater numbers usually lead to more satisfactory results. However, the model can start over-fitting if the numbers become too large. As a result, it demonstrates again that parameter tuning is required to achieve the best performance.

## 2.5 Conclusion

This chapter introduced a matrix completion method under the low-rank assumption and with specific regularizers that accommodate the property of the given real-world data set. We focus on predicting human stress levels with incomplete sensor data from wearable devices. To recover missing sensor data, our model, Data Completion with Diurnal Regularizers (DCDR), exploits user diurnal and weekly behavior patterns as two effective regularizers, thereby precisely inferring unobserved sensor data. To predict

human stress levels, we propose Temporally Hierarchical Attention Network (THAN) considers temporally hierarchical structures in sensor data with hierarchical RNNs with the attention mechanism. The extensive experiments demonstrate that our approach outperforms competitive baseline methods in sensor data completion and stress level prediction. Moreover, the analysis results also show the robustness and effectiveness of our proposed models. We can conclude that the conventional low-rank assumption is not enough for accurate data completion, while the patterns of diurnal behaviors can be beneficial as regularizers.

## CHAPTER 3

### Tensor Completion Under Low-rank Assumption

The content of this chapter is from my previous paper *Tensor Completion through Total Variation with Initialization from Weighted HOSVD*. This paper has been accepted by the Information Theory and Applications Workshop (ITA 2020). The co-authors are Longxiu Huang and Deanna Needell. I performed computational experiments, analyzed and interpreted the output of these methods, and wrote and edited the manuscript. Longxiu formulated theoretical content (theoretical error bound) for the weighted higher-order singular value decomposition (HOSVD) and wrote and edited the manuscript. Deanna Needell advised on and provided valuable comments on the project.

In this work, we study the tensor completion problem with the deterministic sampling pattern (where the indexes of observed entries are known). We keep the low-rank assumption and introduce the tensor rank. The main contribution is a simple but efficient weighted HOSVD algorithm for recovery from noisy observations. We also use the weighted HOSVD result as an initialization for the total variation minimization algorithm. We demonstrate the accuracy of the weighted HOSVD algorithm from theoretical and numerical perspectives. In the numerical simulation parts, we show that by using the proposed initialization, the total variation minimization algorithm can efficiently fill in the missing data for images and videos.

### 3.1 Motivation and Tensor Background

Tensor, a high-dimensional array that is an extension of the matrix, plays an important role in a wide range of real-world applications [50, 51]. Due to the high-dimensional structure, the tensor could preserve more information compared to the matrix. For instance, a  $k$  frame,  $m \times n$  video stored as an  $m \times n \times k$  tensor will keep the connection between each frame. Splitting the frames or unfolding this tensor may lose some conjunctive information. Most of the real-world datasets are partially missing and incomplete data, which can lead to low performance of downstream applications. Relations between missing and existing data, such as linear dependency and repetitiveness, can be leveraged to recover unavailable data and improve the quality and scale of the incomplete dataset. The task of recovering missing elements from partially observed tensors is called tensor completion and has attracted widespread attention in many applications. e.g., image/video inpainting [52, 1], recommendation systems [53]. Matrix completion problem [54, 55, 56, 57, 58, 59, 60, 61, 62, 63], as a special case of tensor completion problem has been well-studied in the past few decades, which enlightened researchers on developing further tensor completion algorithms. Among different types of data matrices, image data is commonly studied and widely used for performance indicators [1]. One traditional way to target image denoising problems is to minimize the total variation norm [64]. This method is based on the assumption of the local smoothness pattern of the data. Yet in recent decades, thanks to the algorithm development of NMF and nuclear norm minimization (NNM), the low-rank structure assumption has become increasingly popular and extensively applied in related studies [33, 65, 66]. In both matrix completion and tensor completion studies [21, 67], researchers are trying to utilize and balance both assumptions in order to improve the performance of image recovery and video recovery tasks.



### 3.1.1 Tensor Preliminaries and Notations

Tensors, matrices, vectors, and scalars are denoted in different typefaces for clarity below. We summarize our primary notations in Table 3.1.

Typeface	Definition	Example
Calligraphic boldface capital letters	Tensors	$\mathcal{X}, \mathcal{T}, \dots$
Capital letters	Matrices	$X, M, \dots$
Lower boldface letters	Vectors	$\mathbf{x}, \mathbf{v}, \dots$
Regular letters	Scalars	$a, b, c, \dots$

Table 3.1: Key notation in Chapters 3 and 4.

The set of the first  $d$  natural numbers will be denoted by  $[d] := \{1, \dots, d\}$ . Let  $\mathcal{T} \in \mathbb{R}^{d_1 \times \dots \times d_n}$  and  $\alpha \in \mathbb{R}$ , then  $\mathcal{T}^{(\alpha)}$  represents the pointwise power operator i.e.,  $(\mathcal{T}^{(\alpha)})_{i_1 \dots i_n} = (\mathcal{T}_{i_1 \dots i_n})^\alpha$ . We use  $\mathcal{T} \succ 0$  to denote the tensor with  $\mathcal{T}_{i_1 \dots i_n} > 0$  for all  $i_1, \dots, i_n$ . We use  $\mathbf{1}_\Omega$  to denote the tensor with all entries equal to 1 on  $\Omega$  and 0 otherwise.

**Definition 1** (Tensor [68]). *A **tensor** is a multidimensional array. The dimension of a tensor is called the order (also called the mode). The space of a real tensor of order  $n$  and of size  $d_1 \times \dots \times d_n$  is denoted as  $\mathbb{R}^{d_1 \times \dots \times d_n}$ . The elements of a tensor  $\mathcal{T} \in \mathbb{R}^{d_1 \times \dots \times d_n}$  are denoted by  $\mathcal{T}_{i_1 \dots i_n}$ .*

For an order  $n$  tensor  $\mathcal{T}$  can be matricized in  $n$  ways by unfolding it along each of the  $n$  modes, next we will give the definition for the matricization of a given tensor.

**Definition 2** (Matricization of a tensor [68]). *The **mode- $k$  matricization of tensor**  $\mathcal{T} \in \mathbb{R}^{d_1 \times \dots \times d_n}$  is the matrix, which is denoted as  $\mathcal{T}_{(k)} \in \mathbb{R}^{d_k \times \prod_{j \neq k} d_j}$ , whose columns are composed of all the vectors obtained from  $\mathcal{T}$  by fixing all indices but  $i$ th.*

In order to illustrate the matricization of a tensor, let us consider the following example.

**Example 1.** Let  $\mathcal{T} \in \mathbb{R}^{3 \times 4 \times 2}$  with the following entries (column as the index of one

mode represents the set of all elements on that mode.):

$$T_{1,:,:} = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix} \quad T_{2,:,:} = \begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix},$$

then the three mode-n matricizations are

$$\mathcal{T}_{(1)} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix},$$

$$\mathcal{T}_{(2)} = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 4 & 5 & 6 & 16 & 17 & 18 \\ 7 & 8 & 9 & 19 & 20 & 21 \\ 10 & 11 & 12 & 22 & 23 & 24 \end{bmatrix},$$

$$\mathcal{T}_{(3)} = \begin{bmatrix} 1 & 2 & 3 & \dots & 10 & 11 & 12 \\ 13 & 14 & 15 & \dots & 22 & 23 & 24 \end{bmatrix}.$$

**Definition 3** (Folding Operator). *Suppose  $\mathcal{T}$  be a tensor. The **mode- $k$  folding operator** of a matrix  $M = \mathcal{T}_{(k)}$ , denoted as  $\text{fold}_k(M)$ , is the inverse operator of the unfolding operator.*

**Definition 4** ( $\infty$  norm). *Let  $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times \dots \times d_n}$ , the  $\|\mathcal{T}\|_\infty$  is defined as*

$$\|\mathcal{T}\|_\infty = \max_{i_1, i_2, \dots, i_n} |\mathcal{T}_{i_1 i_2 \dots i_n}|.$$

*The unit ball under  $\infty$  norm is denoted by  $\mathbf{B}_\infty$ .*

**Definition 5** (Frobenius norm). *The **Frobenius norm** for tensor  $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times \dots \times d_n}$  is defined as*

$$\|\mathcal{T}\|_F = \sqrt{\sum_{i_1, i_2, \dots, i_n} \mathcal{T}_{i_1 i_2 \dots i_n}^2}.$$

**Definition 6** (Product Operations).

- *Mode-k Product:* The **mode-k product** of tensor  $\mathcal{T} \in \mathbb{R}^{d_1 \times \dots \times d_n}$  and matrix  $A \in \mathbb{R}^{d \times d_k}$  is defined by

$$\mathcal{T} \times_k A = \text{fold}_k(A\mathcal{T}_{(k)}),$$

i.e.,

$$(\mathcal{T} \times_k A)_{i_1 \dots i_{k-1} j i_{k+1} \dots i_n} = \sum_{i_k=1}^{d_k} \mathcal{T}_{i_1 i_2 \dots i_n} A_{j i_k}.$$

- *Outer product:* Let  $\mathbf{a}_1 \in \mathbb{R}^{d_1}, \dots, \mathbf{a}_n \in \mathbb{R}^{d_n}$ . The **outer product** among these  $n$  vectors is a tensor  $\mathcal{T} \in \mathbb{R}^{d_1 \times \dots \times d_n}$  defined as:

$$\mathcal{T} = \mathbf{a}_1 \otimes \dots \otimes \mathbf{a}_n, \quad \mathcal{T}_{i_1, \dots, i_n} = \prod_{k=1}^n \mathbf{a}_k(i_k).$$

- *Kronecker product of matrices:* The **Kronecker product** of  $A \in \mathbb{R}^{I \times J}$  and  $B \in \mathbb{R}^{K \times L}$  is denoted by  $A \otimes B$ . The result is a matrix of size  $(KI) \times (JL)$  and defined by

$$A \otimes B = \begin{bmatrix} A_{11}B & A_{12}B & \dots & A_{1J}B \\ A_{21}B & A_{22}B & \dots & A_{2J}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{I1}B & A_{I2}B & \dots & A_{IJ}B \end{bmatrix}.$$

- *Khatri–Rao product:* Given matrices  $A \in \mathbb{R}^{d_1 \times r}$  and  $B \in \mathbb{R}^{d_2 \times r}$ , their **Khatri–Rao product** is denoted by  $A \odot B$ . The result is a matrix of size  $(d_1 d_2) \times r$  defined by

$$A \odot B = \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \dots & \mathbf{a}_r \otimes \mathbf{b}_r \end{bmatrix},$$

where  $\mathbf{a}_i$  and  $\mathbf{b}_i$  stands for the  $i$ -th column of  $A$  and  $B$  respectively.

- *Hadamard product:* Given two tensors  $\mathcal{X}$  and  $\mathcal{Y}$ , both of size  $d_1 \times \dots \times d_n$ , their **Hadamard product** is denoted by  $\mathcal{X} \square \mathcal{Y}$ . The result is also of the size  $d_1 \times d_2 \times \dots \times d_n$  and the elements of  $\mathcal{X} \square \mathcal{Y}$  are defined as the elementwise

tensor product:

$$(\mathcal{X} \boxtimes \mathcal{Y})_{i_1 i_2 \dots i_n} = \mathcal{X}_{i_1 i_2 \dots i_n} \mathcal{Y}_{i_1 i_2 \dots i_n}.$$

**Definition 7** (Rank-one Tensors). *An  $n$ -order tensor  $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times \dots \times d_n}$  is rank 1 if it can be written as the out product of  $n$  vectors, i.e.,*

$$\mathcal{T} = \mathbf{a}_1 \otimes \dots \otimes \mathbf{a}_n.$$

**Definition 8** (Tensor (CP) rank [69, 70]). *The rank of a tensor  $\mathcal{X}$ , denoted  $\text{rank}(\mathcal{X})$ , is defined as the smallest number of rank-one tensors that generate  $\mathcal{X}$  as their sum. We use  $K_r$  to denote the cone of rank- $r$  tensors.*

Different from the case of matrices, the rank of a tensor is presently not understood well. And the problem of computing the CP rank of a tensor is NP-hard [71]. Next, we will introduce another rank definition related to the tensor.

**Definition 9** (Tucker rank [70]). *Let  $\mathcal{X} \in \mathbb{R}^{d_1 \times \dots \times d_n}$ . The tuple  $(r_1, \dots, r_n) \in \mathbb{N}^n$ , where  $r_k = \text{rank}(\mathcal{X}_{(k)})$  is called tensor Tucker rank of  $\mathcal{X}$ . We use  $K_{\mathbf{r}}$  to denote the cone of Tucker rank  $\mathbf{r}$  tensors.*

## 3.2 Problem Formulation and Related Work

The total variation (TV) minimization method is a class of algorithms where the initialization could impact the algorithm's efficiency (see Section 3.3 for details about the TV minimization). In order to find a good initialization for the iterative TV minimization algorithm, we would like to solve the following questions.

**Question 1.** *Given a deterministic sampling pattern  $\Omega$  and corresponding (possibly noisy) observations from the tensor, what type of recovery error can we expect, in what metric, and how may we efficiently implement this recovery?*

**Question 2.** *Given a sampling pattern  $\Omega$ , and noisy observations  $\mathcal{T}_\Omega + \mathcal{Z}_\Omega$ , for what rank-one weight tensor  $\mathcal{H}$  can we efficiently find a tensor  $\widehat{\mathcal{T}}$  so that  $\|\mathcal{H} \boxtimes (\widehat{\mathcal{T}} - \mathcal{T})\|_F$*

is small compared to  $\|\mathcal{H}\|_F$ ? And how can we efficiently find such weight tensor  $\mathcal{H}$ , or certify that a fixed  $\mathcal{H}$  has this property?

In order to find weight tensor, we consider the optimization problem

$$\mathcal{W} := \underset{\mathcal{X} \succ 0, \text{rank}(\mathcal{X})=1}{\text{argmin}} \|\mathcal{X} - \mathbf{1}_\Omega\|_F.$$

$\mathcal{W}$  can be estimated by using the least square algorithm (with a constraint on CP rank) [72]. After we find  $\mathcal{W}$ , then we consider the following optimization problem to estimate  $\mathcal{T}$ :

$$\widehat{\mathcal{T}} = \mathcal{W}^{(-1/2)} \boxtimes \underset{\text{Tucker\_rank}(\mathcal{T})=\mathbf{r}}{\text{argmin}} \|\mathcal{T} - \mathcal{W}^{(-1/2)} \boxtimes \mathcal{Y}_\Omega\|_F, \quad (3.1)$$

where  $\mathcal{Y}_\Omega = \mathcal{T}_\Omega + \mathcal{Z}_\Omega$ . As we know, to solve problem (3.1) is NP-hard [71]. In order to solve (3.1) in polynomial time, we consider the HOSVD process [73]. Assume that  $\mathcal{T}$  has Tucker rank  $\mathbf{r} = [r_1, \dots, r_n]$ . Let

$$\widehat{A}_i = \underset{\text{rank}(A)=r_i}{\text{argmin}} \|A - (\mathcal{W}^{(-1/2)} \boxtimes \mathcal{Y}_\Omega)_{(i)}\|_2,$$

and set  $\widehat{U}_i$  to be the left singular vector matrix of  $\widehat{A}_i$ . Then the estimated tensor is of the form

$$\widehat{\mathcal{T}} = \mathcal{W}^{(-1/2)} \boxtimes ((\mathcal{W}^{(-1/2)} \boxtimes \mathcal{Y}_\Omega) \times_1 \widehat{U}_1 \widehat{U}_1^T \times_2 \cdots \times_n \widehat{U}_n \widehat{U}_n^T).$$

In the following content, we call our algorithm weighted HOSVD algorithm.

With the output from weighted HOSVD,  $\widehat{\mathcal{T}}$ , we can solve the following total variation problem with  $\widehat{\mathcal{T}}$  as initialization:

$$\begin{aligned} \min_{\mathcal{X}} \quad & \|\mathcal{X}\|_{TV} \\ \text{s.t.} \quad & \mathcal{X}_\Omega = \mathcal{Y}_\Omega. \end{aligned}$$

This total variation minimization problem can be solved by an iterative algorithm. We will discuss the details of the algorithm in Section 3.3.

While the task went up to tensor completion, the low-rank assumption became even harder to approach. Some of the recent studies performed matrix completion algorithms on the unfolded tensor and obtained considerable results. For example, [1] introduced nuclear norm to unfolded tensors and took the weighted average for loss function. They proposed several algorithms to solve the following minimization problem:

$$\begin{aligned} \min_{\mathcal{X}} \quad & \sum_{i=1}^n \alpha_i \|\mathcal{X}_{(i)}\|_* \\ \text{s.t.} \quad & \mathcal{X}_\Omega = \mathcal{T}_\Omega. \end{aligned}$$

Xu et al. [74] applied low-rank matrix factorization (LRMF) to all-mode unfolded tensors and defined the minimization problem as following:

$$\begin{aligned} \min_{\mathcal{X}, \mathbf{A}, \mathbf{B}} \quad & \sum_{i=1}^n \alpha_i \|\mathcal{X}_{(i)} - A_i B_i\|_F^2 \\ \text{s.t.} \quad & \mathcal{X}_\Omega = \mathcal{T}_\Omega, \end{aligned}$$

where  $\mathbf{A} = \{A_1, \dots, A_n\}$  and  $\mathbf{B} = \{B_1, \dots, B_n\}$  are the set of low-rank matrices with different size according to the unfolded tensor. This method is called TMac and can be solved using alternating minimization.

While researchers often test the performance of their tensor completion algorithms on image/video/MRI data, they started to combine NNM and LRMF with total variation norm minimization when dealing with relevant recovery tasks. For example, [67] introduced the TV regularization into the tensor completion problem:

$$\begin{aligned} \min_{\mathcal{X}, A, B} \quad & \sum_{i=1}^n \alpha_i \|\mathcal{X}_{(i)} - A_i B_i\|_F^2 + \mu \|B_3\|_{TV} \\ \text{s.t.} \quad & \mathcal{X}_\Omega = \mathcal{T}_\Omega. \end{aligned}$$

Note that the equation above only computes the TV-norm for the first 2 modes (with  $B_3$ ). For example, assume that  $\mathcal{X}$  is a video that can be treated as a 3-order tensor,

then the TV-norm of  $B_3$  only counts the variation within each frame and does not count the variation between frames.

To deal with RGB image data as tensors, Li et al. [75] unfolded the tensor in 2 ways (the horizontal and vertical dimensions) and minimized the TV and nuclear norms of each unfolded matrix. Their minimization problem could be written as follows:

$$\begin{aligned} \min_{\mathcal{X}} \quad & \sum_{i=1}^2 (\alpha_i \|\mathcal{X}_{(i)}\|_* + \mu \|\mathcal{X}_{(i)}\|_{TV}), \\ \text{s.t.} \quad & \mathcal{X}_\Omega = \mathcal{T}_\Omega. \end{aligned}$$

In our experiments, we noticed that for a small percentage of observations (for instance, less than 50% entries are observed), the (unfolded) output tensor from TV-minimization recovery will have long-tail, slow-decaying singular values, which distribute similarly to the singular values of the (unfolded) original tensor. However, NNM will force a large portion of smaller singular values to be zero, which cannot be ignored in the original tensor. Therefore one should be really careful with the choice of minimization problem when performing the completion tasks on a specific dataset. We will discuss the details in Section 3.5.

### 3.3 TV Minimization Algorithm

#### 3.3.1 Matrix Denoising Algorithm

TV-norm is often discretized by [76]:

$$\|u\|_{TV} \approx \sum_{i,j} \sqrt{(\nabla_x u)_{i,j}^2 + (\nabla_y u)_{i,j}^2}.$$

Hence the image-denoising problem is defined as:

$$\min_X \sum_{i,j} \sqrt{(\nabla_x M_{i,j})^2 + (\nabla_y M_{i,j})^2} + \lambda \|M - X\|_F.$$

This minimization problem can be solved by a gradient descent algorithm (see [64] Section 3.3 for algorithm details).

### 3.3.2 Tensor Completion with TV

Similar to the image denoising algorithm, we first compute the divergence of each entry and move each entry toward the divergence direction. To keep the existing entries unchanged, we project the observed entries to their original values at each step. We consider the following minimization problem:

$$\begin{aligned} \min_{\mathcal{X}} \quad & \|\mathcal{X}\|_{TV}, \\ \text{s.t.} \quad & \mathcal{X}_\Omega = \mathcal{T}_\Omega. \end{aligned}$$

The related algorithm is summarized in Algorithm 1.

---

**Algorithm 1:** Tensor Completion Through TV Minimization

---

**Input** : Incomplete tensor  $\mathcal{T} \in \mathbb{R}^{d_1 \times \dots \times d_n}$ ; Sampling pattern

$$\Omega \in \{0, 1\}^{d_1 \times \dots \times d_n}; \text{ stepsize } h_k, \text{ threshold } \lambda; \mathcal{X}^0 \in \mathbb{R}^{d_1 \times \dots \times d_n}.$$

Set  $\mathcal{X}^0 = \mathcal{X}^0 + (\mathcal{T}_\Omega - \mathcal{X}_\Omega^0)$ .

**for**  $k = 0 : K$  **do**

**for**  $i = 1 : n$  **do**

$$\quad \nabla_i(\mathcal{X}_{\alpha_1, \dots, \alpha_n}^k) = \mathcal{X}_{\alpha_1, \dots, \alpha_i+1, \dots, \alpha_n}^k - \mathcal{X}_{\alpha_1, \dots, \alpha_i, \dots, \alpha_n}^k, (\alpha_i = 1, 2, \dots, d_i - 1)$$

$$\quad (\nabla_i(\cdot) = 0 \text{ when } \alpha_i = d_i)$$

$$\quad \Delta_i(\mathcal{X}_{\alpha_1, \dots, \alpha_n}^k) = \mathcal{X}_{\alpha_1, \dots, \alpha_i-1, \dots, \alpha_n}^k + \mathcal{X}_{\alpha_1, \dots, \alpha_i+1, \dots, \alpha_n}^k - 2\mathcal{X}_{\alpha_1, \dots, \alpha_i, \dots, \alpha_n}^k, (\alpha_i = 2, 3, \dots, d_i - 1) (\Delta_i(\cdot) = 0 \text{ when } \alpha_i = 1 \text{ or } d_i)$$

$$\quad \Delta(\mathcal{X}_{\alpha_1, \dots, \alpha_n}^k) = \sum_i \Delta_i(\mathcal{X}_{\alpha_1, \dots, \alpha_n}^k)$$

$$\quad \mathcal{X}_{\alpha_1, \dots, \alpha_n}^{k+1} = \mathcal{X}_{\alpha_1, \dots, \alpha_n}^k + h_k \cdot \text{shrink}\left(\frac{\Delta(\mathcal{X}_{\alpha_1, \dots, \alpha_n}^k)}{\sqrt{\sum_i \nabla_i^2(\mathcal{X}_{\alpha_1, \dots, \alpha_n}^k)}}, \lambda\right)$$

$$\quad \mathcal{X}_\Omega^{k+1} = \mathcal{T}_\Omega$$

**Output:**  $\mathcal{X}^K$

---

In Algorithm 1, the Laplacian operator computes the divergence of the gradient for each point. The shrink operator moves the input towards 0 with distance  $\lambda$ , formally



defined as:

$$\text{shrink}(x, \lambda) = \mathbf{sign}(x) \cdot \max(|x| - \lambda, 0)$$

For  $\mathcal{X}^0$  initialization, simple tensor completion with total variation (TVTC) method would set  $\mathcal{X}^0$  to be a zero tensor, i.e.,  $\mathcal{X}^0 = \mathbf{0}^{d_1 \times \dots \times d_n}$ , but our proposed method will set  $\mathcal{X}^0$  to be the result from weighted HOSVD (w-HOSVD). We will show the theoretical and experimental advantages of w-HOSVD in Section 3.4.

### 3.4 Theoretical Error Bound

In order to show the efficiency  $\widehat{\mathcal{T}}$  as the initialization for the total variation algorithm, we only need to show that  $\widehat{\mathcal{T}}$  is close to  $\mathcal{T}$ . The following theorem gives the bound of  $\|\mathcal{W} \boxtimes (\mathcal{T} - \widehat{\mathcal{T}})\|_F$ .

**Theorem 2.** *Let  $\mathcal{W} = \mathbf{w}_1 \otimes \dots \otimes \mathbf{w}_n \in \mathbb{R}^{d_1 \times \dots \times d_n}$  have strictly positive entries, and fix  $\Omega \subseteq [d_1] \times \dots \times [d_n]$ . Suppose that  $\mathcal{T} \in \mathbb{R}^{d_1 \times \dots \times d_n}$  has Tucker rank  $\mathbf{r} = [r_1, \dots, r_n]$  for problem (3.1). Then with probability at least  $1 - 2^{-|\Omega|/2}$ ,*

$$\|\mathcal{W}^{(1/2)} \boxtimes (\mathcal{T} - \widehat{\mathcal{T}})\|_F \leq 4\sigma\mu\sqrt{|\Omega| \ln(2)} + 2\|\mathcal{T}\|_\infty \|\mathcal{W}^{(1/2)} - \mathcal{W}^{(-1/2)} \boxtimes \mathbf{1}_\Omega\|_F,$$

where  $\mu^2 = \max_{(i_1, \dots, i_n) \in \Omega} \frac{1}{\mathcal{W}_{i_1 \dots i_n}}$ .

Notice that the upper bound in Theorem 2 is for the optimal output  $\widehat{\mathcal{T}}$  for problems (3.1). But the upper bound in Theorem 2 contains no information about the rank of the underlying tensor  $\mathcal{T}$ . In order to introduce the rank information of the underlying tensor  $\mathcal{T}$ , we restrict our analysis for Problem (3.1) by considering the HOSVD process. We have the following result.

**Theorem 3** (General upper bound for tensor with Tucker rank  $\mathbf{r}$ ). *Let  $\mathcal{W} = \mathbf{w}_1 \otimes \dots \otimes \mathbf{w}_n \in \mathbb{R}^{d_1 \times \dots \times d_n}$  have strictly positive entries, and fix  $\Omega \subseteq [d_1] \times \dots \times [d_n]$ . Suppose that  $\mathcal{T} \in \mathbb{R}^{d_1 \times \dots \times d_n}$  has Tucker rank  $\mathbf{r} = [r_1 \ \dots \ r_n]$ . Suppose that  $\mathcal{Z}_{i_1 \dots i_n} \sim \mathcal{N}(0, \sigma^2)$*

and let

$$\widehat{\mathcal{T}} = \mathcal{W}^{(-1/2)} \boxtimes ((\mathcal{W}^{(-1/2)} \boxtimes \mathcal{Y}_\Omega) \times_1 \widehat{U}_1 \widehat{U}_1^T \times_2 \cdots \times_n \widehat{U}_n \widehat{U}_n^T)$$

where  $\widehat{U}_1, \dots, \widehat{U}_n$  are obtained by considering the HOSVD approximation process. Then with probability at least

$$1 - \sum_{i=1}^n \frac{1}{d_i + \prod_{j \neq i} d_j}$$

over the choice of  $\mathcal{Z}$ , we have

$$\begin{aligned} & \|\mathcal{W}^{(1/2)} \boxtimes (\mathcal{T} - \widehat{\mathcal{T}})\|_F \\ & \leq \left( \sum_{k=1}^n \sqrt{r_k \log \left( d_k + \prod_{j \neq k} d_j \right) \mu_k} \right) \sigma + \left( \sum_{k=1}^n r_k \|(\mathcal{W}^{(-\frac{1}{2})} \boxtimes 1_\Omega - \mathcal{W}^{(\frac{1}{2})})_{(k)}\|_2 \right) \|\mathcal{T}\|_\infty. \end{aligned} \tag{3.2}$$

where

$$\mu_k^2 = \max_{i_1, \dots, i_n} \left\{ \sum_{i_1, \dots, i_{k-1}, i_{k+1}, \dots, i_n} \frac{1_{(i_1, i_2, \dots, i_n) \in \Omega}}{\mathcal{W}_{i_1 i_2 \dots i_n}}, \sum_{i_k} \frac{1_{(i_1, i_2, \dots, i_n) \in \Omega}}{\mathcal{W}_{i_1 i_2 \dots i_n}} \right\}.$$

## 3.5 Experiments

In this section, we conducted numerical simulations to show the efficiency of the proposed weighted HOSVD algorithm first. Then, we will include the experiment results to show that using the weighted HOSVD algorithm results as initialization of the TV minimization algorithm can accelerate the convergence speed of the original TV minimization.

### 3.5.1 Simulations for Weighted HOSVD

We test our weighted HOSVD algorithm for 3-order tensor of the form  $\mathcal{T} = \mathcal{C} \times_1 U_1 \times_2 U_2 \times_3 U_3$  under uniform and nonuniform sampling patterns, where  $U_i \in \mathbb{R}^{d_i \times r_i}$  and  $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$  with  $r_i < d_i$ . First, we generate  $\mathcal{T}$  of the size  $100 \times 100 \times 100$  with

Tucker rank  $\mathbf{r} = [r, r, r]$ , and  $r$  varies from 2 to 10. Then we add Gaussian random noise with  $\sigma = 10^{-2}$  to  $\mathcal{T}$ . Next, we generate a sampling pattern  $\Omega$  containing 10% uniformly random entries of  $\mathcal{T}$ . We estimate  $\widehat{\mathcal{T}}_o$ ,  $\widehat{\mathcal{T}}_p$  and  $\widehat{\mathcal{T}}_w$  by considering

$$\begin{aligned}\widehat{\mathcal{T}}_o &= \underset{\text{Tucker\_rank}(\mathcal{X})=\mathbf{r}}{\operatorname{argmin}} \|\mathcal{X} - \mathcal{Y}_\Omega\|_F, \\ \widehat{\mathcal{T}}_p &= \underset{\text{Tucker\_rank}(\mathcal{X})=\mathbf{r}}{\operatorname{argmin}} \|\mathcal{X} - \frac{1}{p}\mathcal{Y}_\Omega\|_F, p = \frac{|\Omega|}{d_1 d_2 d_3}, \\ \widehat{\mathcal{T}}_w &= \mathcal{W}^{(-1/2)} \boxtimes \\ &\quad \underset{\text{Tucker\_rank}(\mathcal{X})=\mathbf{r}}{\operatorname{argmin}} \|\mathcal{X} - \mathcal{W}^{(-1/2)} \boxtimes \mathcal{Y}_\Omega\|_F,\end{aligned}$$

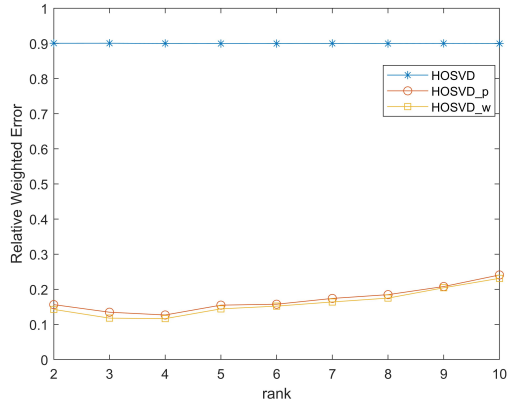
from the truncated HOSVD algorithm. We give names HOSVD associated with  $\widehat{\mathcal{T}}_o$ , HOSVD\_p associated with  $\widehat{\mathcal{T}}_p$ , and HOSVD\_w associated with  $\widehat{\mathcal{T}}_w$ . For an estimate  $\widehat{\mathcal{T}}$ , we consider both the weighted and unweighted relative errors:

$$\frac{\|\mathcal{W}^{(1/2)} \boxtimes (\widehat{\mathcal{T}} - \mathcal{T})\|_F}{\|\mathcal{W}^{(1/2)} \boxtimes \mathcal{T}\|_F} \text{ and } \frac{\|\widehat{\mathcal{T}} - \mathcal{T}\|_F}{\|\mathcal{T}\|_F}.$$

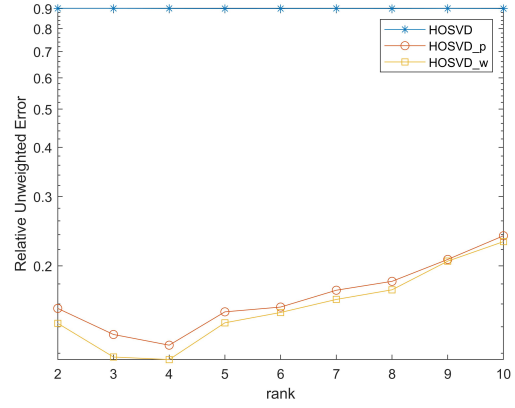
Both the weighted and unweighted relative errors are plotted in Figure 3.1. Each point represents the mean error of 20 experiments with  $\mathcal{T}$  generated from different random seeds.

When each entry is sampled randomly uniformly with probability  $p$ , then we have  $\mathbb{E}(\mathcal{Y}_\Omega) = p\mathcal{T}$  which implies that the estimate  $\widehat{\mathcal{T}}_p$  should perform better than  $\widehat{\mathcal{T}}_o$ . In Figure 3.1, we take the sampling pattern to be uniform at random. The estimates  $\widehat{\mathcal{T}}_p$  and  $\widehat{\mathcal{T}}_w$  perform significantly better than  $\widehat{\mathcal{T}}_o$  as expected.

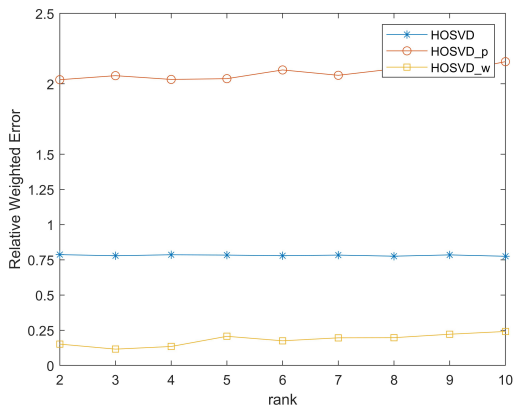
In Figure 3.2, the set-up is the same as the one in Figure 3.1 except that the random sampling is weighted (with  $\mathcal{W}$  as the weights). Then using  $\frac{1}{p}$  is not a good weight tensor where, as shown in Figure 3.2, the relative error from  $\widehat{\mathcal{W}}_p$  is greater than 1. But  $\widehat{\mathcal{T}}_w$  still works better than  $\widehat{\mathcal{T}}_o$ .



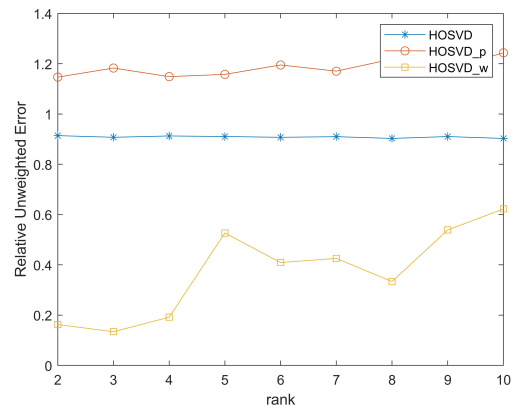
(a)



(b)

Figure 3.1: Relative error for uniformly sampled  $\Omega$ .

(a)



(b)

Figure 3.2: Relative error for non-uniformly sampled  $\Omega$ .

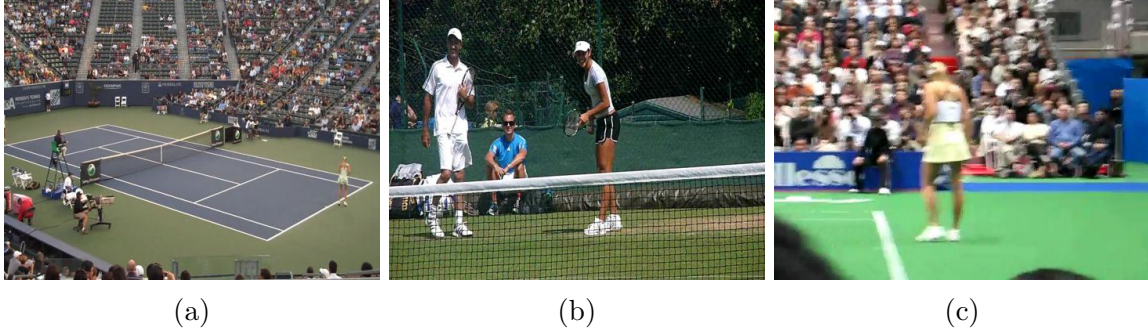


Figure 3.3: The first frame of tested videos

### 3.5.2 Simulations for TV with Initialization from Weighted HOSVD

To show the advantage of weighted HOSVD, we test our proposed algorithm and simple TV minimization method, along with a baseline algorithm on video data. We mask a specific ratio of entries and conduct each completion algorithm to obtain the completion results  $\hat{\mathcal{T}}$ . The tested sampling rates (SR) are 10%, 30%, 50%, and 80%. We then compute the relative root mean square error (RSE):

$$\text{RSE} = \frac{\|\hat{\mathcal{T}} - \mathcal{T}\|_F}{\|\mathcal{T}\|_F}$$

For each method to evaluate their performance. Meanwhile, we compare the mean running time until the algorithm converges to some preset threshold.




We have tested our algorithm on three video data from Olympic Sports Dataset<sup>1</sup>. The video data are tennis-serve data from an Olympic Sports Dataset. The three videos are color videos. In our simulation, we use the same setup as the one in [77]. We pick 30 equally spaced frames from each video. Each image frame is reshaped to a  $360 \times 480 \times 3$  tensor. Then each video is transformed into a 4-D tensor data of size  $360 \times 480 \times 3 \times 30$ . The first frame of each video after preprocessing is illustrated in Figure 3.3.

---

<sup>1</sup>Publicly available at <http://vision.stanford.edu/Datasets/OlympicSports>.

### 3.6 Numerical Results and Discussion

Table 3.2: The relative square error (RSE) and time spend for different algorithms on video data (wHOSVD-TV stands for TV algorithm with Initialization from Weighted HOSVD).

Video	Method	SR	RSE	Time(S)
	wHOSVD-TV	10%	0.2080	82.26
		30%	0.1418	50.40
		50%	0.1045	41.31
		80%	0.0566	33.21
	ST-HOSVD	10%	N/A	N/A
		30%	0.1941	521.94
		50%	0.1381	175.82
		80%	0.0667	128.68
	wHOSVD-TV	10%	0.2694	35.21
		30%	0.1888	21.08
		50%	0.1411	16.55
		80%	0.0767	12.88
	ST-HOSVD	10%	N/A	N/A
		30%	0.2249	1130.77
		50%	0.1480	1304.31
		80%	0.0749	976.65
	wHOSVD-TV	10%	0.2198	156.5
		30%	0.1394	87.89
		50%	0.0955	72.15
		80%	0.0470	18.44
	ST-HOSVD	10%	N/A	N/A
		30%	0.1734	560.97
		50%	0.1105	158.74
		80%	0.0594	52.09

The video simulation results are reported in Table 3.2 and Figure 3.4. There are existing studies that performed the same completion task on the same dataset (see [77].) In [77], the ST-HOSVD [78] had the best performance among several low-rank-based tensor completion algorithms.

We record each completion task’s error and running time and compare them with the previous low-rank-based algorithms. One can observe that the TV-based algorithm is more often compatible with video data.

We also have implemented total variations with zero filling initialization for the

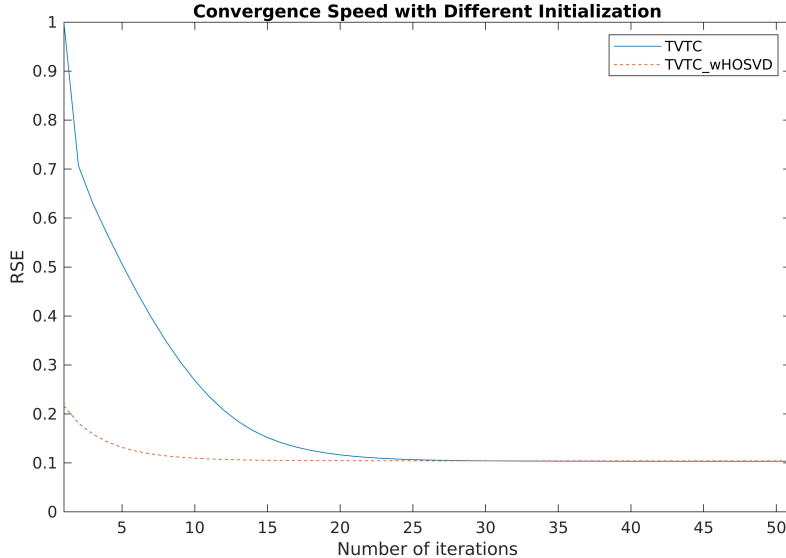


Figure 3.4: Comparison between TVTC and wHOSVD-TV on video 1 with SR = 50%.

entries that are not observed and with the tensor obtained from weighted HOSVD termed TVTC and wHOSVD-TV, respectively. The iterative results are shown in Figure 3.4, which shows that using the result from weighted HOSVD as initialization could notably reduce the iterations of TV-minimization for achieving the convergence threshold ( $\|\mathcal{X}^k - \mathcal{X}^{k-1}\|_F < 10^{-4}$ ).

The relation between the smoothness pattern (where the TV minimization is based) and the low-rank pattern (where the optimization under low-rank constraint is based) is mysterious. Many existing studies use TV minimization for the image completion/denoising task [76, 64], and many other studies use optimization under low-rank constraints to handle the same task [79, 80]. Also, recent studies take the two patterns simultaneously and convert the task to a mixed optimization problem [67, 81]. Through experiments, we find that, with uniform random missing entries, the singular values from total variation minimization on image-like data have a similar distribution pattern to those from the original image.

We uniformly randomly mask 70% entries for several grey-scale images and performed both nuclear norm minimization and total variation minimization on the images. Then the nuclear norms for the original image, masked image, TV estimates, and

NNM estimates are computed (see Figure 3.5). From this figure, we can see that the nuclear norm from the image recovered from TV minimization is already smaller than the original image, while NNM will produce a further smaller nuclear norm. By observing the singular values of the TV-recovered matrix and the NNM-recovered matrix, we can see that TV-minimization could better capture the smaller singular values and better preserve the overall structures of the original matrix. Unlike user-rating data and synthetic low-rank tensors, the image-like data tends to have a non-trivial tail of singular values. Figure 3.6 shows the similarity between the original image and the TV-recovered image, which hints at the performance comparison between TV-recovery and low-rank recovery.

Additionally, because both  $\|\cdot\|_*$  and  $\|\cdot\|_{TV}$  are convex functions, the mixed minimization problem with restricted observation entries

$$\hat{\mathcal{X}}_{\text{mix}} = \min_{\mathcal{X}} \sum_i \alpha_i \|\mathcal{X}_{(i)}\|_* + \lambda \|\mathcal{X}\|_{TV}$$

will produce a result whose nuclear norm is between  $\|\mathcal{X}_{NNM}\|_*$  and  $\|\mathcal{X}_{TV}\|_*$ , where  $\hat{\mathcal{X}}_{NNM}$  and  $\hat{\mathcal{X}}_{TV}$  are the results from each minimization problem (with the same constraint):

$$\hat{\mathcal{X}}_{NNM} = \min_{\mathcal{X}} \sum_i \alpha_i \|\mathcal{X}_{(i)}\|_*,$$

$$\hat{\mathcal{X}}_{TV} = \min_{\mathcal{X}} \|\mathcal{X}\|_{TV}.$$

To summarize, we thoroughly analyzed the behavior of the TV-minimization approach and the low-rank approach for image completion tasks. We propose a hybrid algorithm that uses the weighted HOSVD as the initialization and TV-minimization for iterative steps on the image and video completion task. The condition for choosing which approach will still depend on the mathematical property (e.g., distribution of singular values) of the original image (or video).



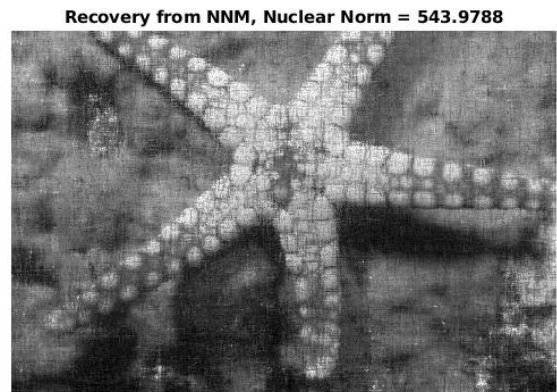
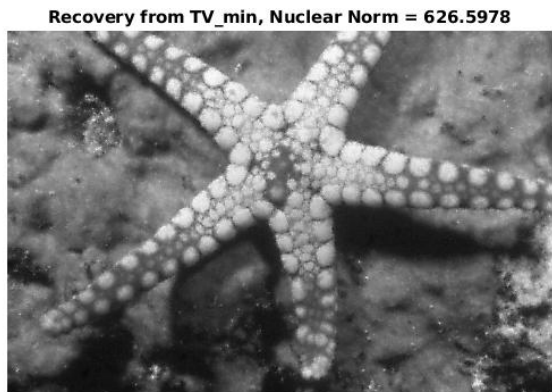
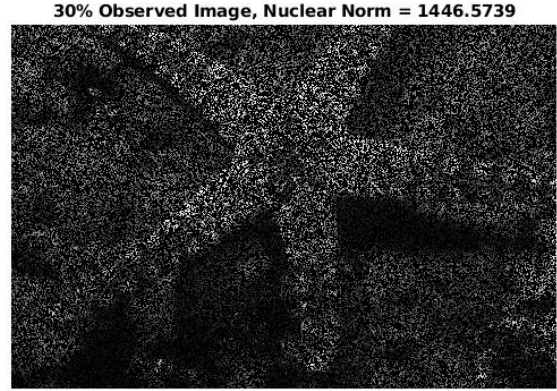
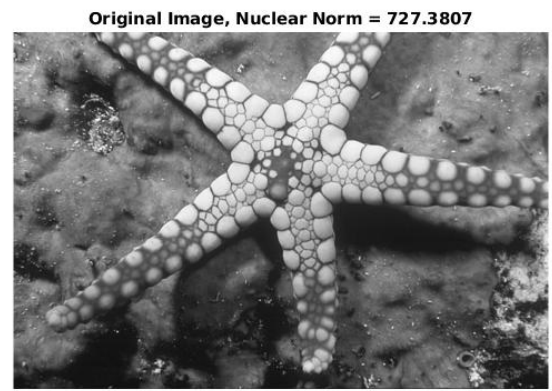
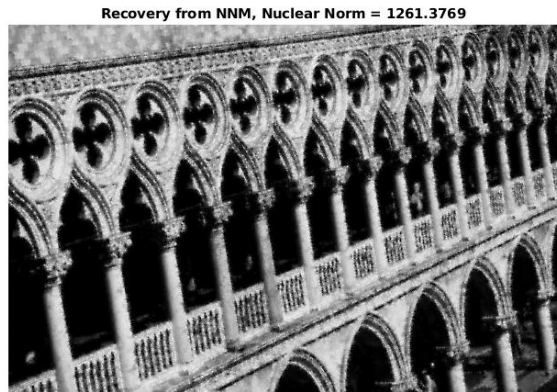
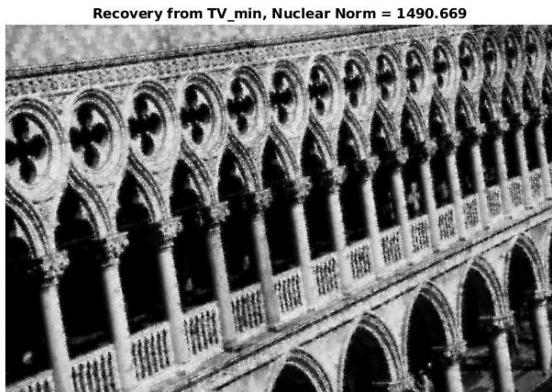
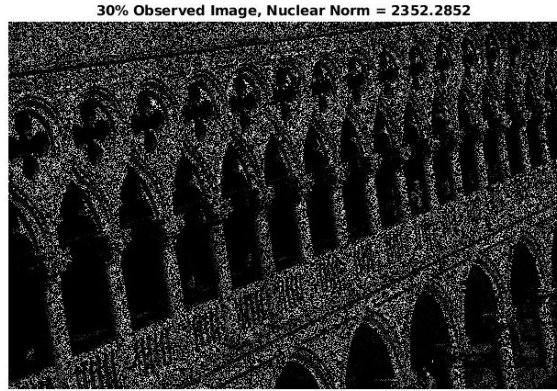
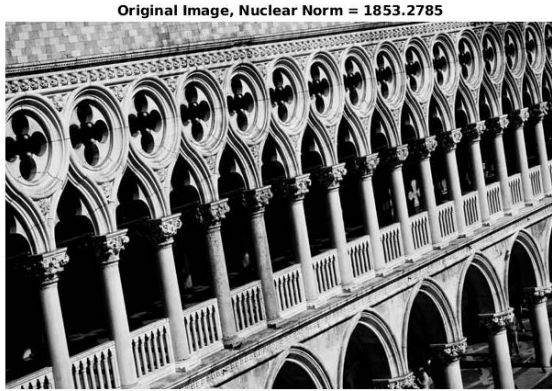


Figure 3.5: Nuclear norm comparison for different recovery patterns

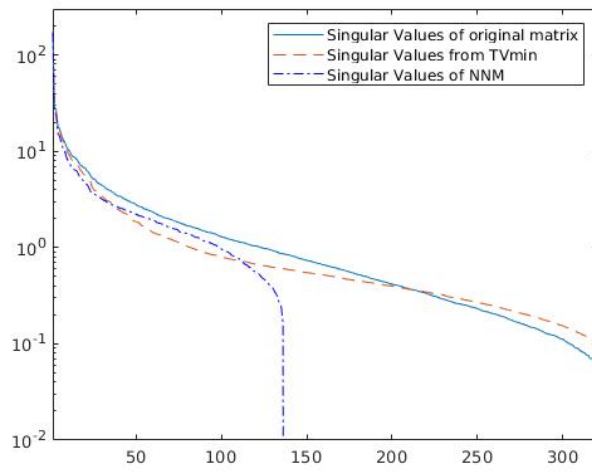


Figure 3.6: Comparison of singular values between TV-recovery and original image, original image is the same as in Figure 3.5

## CHAPTER 4

### Tensor Robust PCA and CUR implementation

The content of this chapter is from my previous paper *Robust Tensor CUR: Rapid Low-Tucker-Rank Tensor Recovery with Sparse Corruptions*. The Co-authors are Hanqin Cai, Longxiu Huang, and Deanna Needell. I performed computational experiments, formulated and implemented methods, analyzed and interpreted the output of these methods, and wrote and edited the manuscript. Hanqin formulated and developed the project, provided ideas and guidance on the alternating projection method, and wrote and edited the manuscript. Longxiu helped formulate and develop the project, provided ideas and guidance on the CUR decomposition method, and wrote and edited the manuscript. Deanna advised on the project and wrote and edited the manuscript. Special thanks to Jianfeng Cai and Jingyang Li for providing the network clustering data set.

We study the tensor robust principal component analysis (TRPCA) problem, a tensorial extension of matrix robust principal component analysis (RPCA), that aims to split the given tensor into an underlying low-rank component and a sparse outlier component. This work proposes a fast algorithm, called Robust Tensor CUR (RTCUR), for large-scale non-convex TRPCA problems under the Tucker rank setting. RTCUR is developed within a framework of alternating projections that projects between the set of low-rank tensors and the set of sparse tensors. We utilize the recently developed tensor CUR decomposition to reduce the computational complexity in each projection substantially. In addition, we develop four variants of RTCUR for different application settings. We demonstrate the effectiveness and computational advantages of RTCUR against state-of-the-art methods on both synthetic and real-world datasets.

## 4.1 Robust Principle Component Analysis (RPCA) Background

As in the matrix setting, but to an even greater extent, principal component analysis is one of the most widely used methods for such dimension reduction tasks. However, standard PCA is over-sensitive to extreme outliers [82]. To overcome this weakness, robust principal component analysis (RPCA) has been proposed to tolerate the sparse outliers in data analysis [83]. In particular, RPCA aims to reconstruct a low-rank matrix  $\mathbf{L}^*$  and a sparse outlier matrix  $\mathbf{S}^*$  from the corrupted observation

$$\mathbf{X} = \mathbf{L}^* + \mathbf{S}^*. \quad (4.1)$$

We seek to find  $\mathbf{L}^*$  and  $\mathbf{S}^*$  by solving the following non-convex problem [83], and we use  $\mathbf{L}$  and  $\mathbf{S}$  to denote the outcomes:

$$\begin{aligned} & \underset{\mathbf{L}, \mathbf{S}}{\text{minimize}} && \|\mathbf{X} - \mathbf{L} - \mathbf{S}\|_{\text{F}} \\ & \text{subject to} && \mathbf{L} \text{ is low-rank,} \\ & && \mathbf{S} \text{ is sparse.} \end{aligned} \quad (4.2)$$

The term *low-rank* refers to the constraint that the rank of  $\mathbf{L}$  is much smaller than its size, and the term *sparse* refers to the restriction on the number of non-zero entries in  $\mathbf{S}$  (for example, each column and row of  $\mathbf{S}$  could contain at most 10% non-zero entries). This RPCA model has been widely studied [84, 85, 12, 86, 87, 88, 89] and applied to many applications, e.g., face modeling [90], feature identification [91], and video background subtraction [92]. However, the original RPCA method can only handle 2-mode arrays (i.e., matrices), while real-world data are often more naturally represented by higher-dimensional arrays (i.e., tensors). For instance, in the application of video background subtraction, a color video is automatically a 4-mode tensor (height, width, frame, and color). To apply RPCA to tensor data, one has to unfold the original tensor into a matrix along some specific mode(s). It may not be clear how the rank of the unfolded matrix depends on the rank of the original tensor in some tensor rank settings.

In addition, we seek methods that utilize the structural information of tensor rather than ignoring the information. Therefore, it is important to generalize the standard RPCA to tensor settings. This task is called tensor robust principal component analysis (TRPCA) [93]. Moving from matrix PCA to the tensor setting could be challenging because some standard results known in the matrix case may not be generalized to tensors smoothly [94]. For example, the rank of a matrix is well and uniquely defined, but researchers have proposed several definitions of tensor rank, such as Tucker rank [95], CP rank [96], and Tubal rank [97].

In this work, we consider the TRPCA problem under the Tucker rank setting. Our main contributions are two-fold:

1. We propose a novel non-convex approach, called Robust Tensor CUR (RTCUR), for large-scale<sup>1</sup> TRPCA problems (see Section 4.4). RTCUR uses a framework of alternating projections and employs a novel mode-wise tensor decomposition [98] for fast low-rank tensor approximation. We demonstrate four variants of the proposed approach with different sampling strategies (see Section 4.4.4 for the details about sampling strategies). The computational complexity of RTCUR is as low as  $\mathcal{O}(n^2 dr^2 \log^2 d)$  or  $\mathcal{O}(ndr^n \log^n d)$ , depending on the sampling strategy, for an input tensor of size<sup>2</sup>  $\mathbb{R}^{d_1 \times \dots \times d_n}$  with Tucker rank  $(r_1, \dots, r_n)$ . Both computational complexities are substantially lower than the state-of-the-art TRPCA methods. For instance, two state-of-the-art methods [93, 79] based on tensor singular value decomposition have computational costs at least  $\mathcal{O}(nd^m r)$ .
2. We verify the empirical advantages of RTCUR with synthetic and real-world datasets (see Section 4.5). In particular, we show that RTCUR has the best speed performance compared to the state-of-the-art of both tensor and matrix RPCA. Moreover, we also show that tensor methods achieve better reconstruction

---

<sup>1</sup>In our context, ‘large-scale’ refers to large  $d$ .

<sup>2</sup>For notation simplicity, we assume the tensor has the same  $d$  and  $r$  along each mode when we discuss complexities. All log operators used in this paper stand for natural logarithms.

quality than matrix methods under certain outlier patterns.

## 4.2 Tensor Robust Principal Component Analysis (TRPCA)

There is a long list of studies on RPCA [99] and low-rank tensor approximation [100], so we refer readers to those two review articles for the aforementioned topics and focus on TRPCA works in this section. Consider a given tensor  $\mathcal{X}$  that can represent a hypergraph network or a multi-dimensional observation [101]; the general assumption of TRPCA is that  $\mathcal{X}$  can be decomposed as the sum of two tensors:

$$\mathcal{X} = \mathcal{L}^* + \mathcal{S}^*, \quad (4.3)$$

where  $\mathcal{L}^* \in \mathbb{R}^{d_1 \times \dots \times d_n}$  is the underlying low-rank tensor and  $\mathcal{S}^* \in \mathbb{R}^{d_1 \times \dots \times d_n}$  is the underlying sparse tensor. Compared to the exact low-rank tensor models, TRPCA model contains an additional sparse tensor  $\mathcal{S}$ , which accounts for potential model outliers and hence more stable with sparse noise. Different from the well-defined matrix rank, there exist various definitions of tensor decompositions that lead to various versions of tensor rank, and lead to different versions of robust tensor decompositions. For example, [102, 93] formulate TRPCA as a convex optimization model based on the tubal rank [97].

Based on the Tucker rank, we aim to solve the non-convex optimization problem in this work:

$$\begin{aligned} & \underset{\mathcal{L}, \mathcal{S}}{\text{minimize}} \quad \|\mathcal{X} - \mathcal{L} - \mathcal{S}\|_{\text{F}} \\ & \text{subject to} \quad \mathcal{L} \text{ is low-Tucker-rank,} \\ & \quad \quad \quad \mathcal{S} \text{ is sparse.} \end{aligned} \quad (4.4)$$

Researchers have developed different optimization methods to solve (4.4) [101, 103, 104]. For example, the work in [101] integrated the Riemannian gradient descent (RGD) and gradient pruning methods to develop a linearly convergent algorithm for

(4.4). This RGD algorithm will also serve as a guideline approach in our experiments. However, one of the major challenges in solving the Tucker rank based TRPCA problem is the high computational cost for computing the Tucker decomposition. If  $\mathcal{L}^*$  is rank- $(r_1, \dots, r_n)$ , the existing methods, e.g., [105, 106, 103, 104, 101], have computational complexity at least  $\mathcal{O}(nd^nr)$ —they are thus computationally challenging in large-scale problems. Thus, it is necessary to develop a highly efficient TRPCA algorithm for time-intensive applications.

### 4.3 Tensor CUR Decompositions

Researchers have been actively studying CUR decompositions for matrices in recent years [107, 108]. For a matrix  $\mathbf{X} \in \mathbb{R}^{d_1 \times d_2}$ , let  $\mathbf{C}$  be a submatrix consisting a subset of columns of  $\mathbf{X}$  with column indices  $J$ ,  $\mathbf{R}$  be a submatrix consisting a subset of rows of  $\mathbf{X}$  with row indices  $I$ , and  $\mathbf{U} = \mathbf{X}(I, J)$ . The theory of CUR decompositions states that  $\mathbf{X} = \mathbf{C}\mathbf{U}^\dagger\mathbf{R}$  if  $\text{rank}(\mathbf{U}) = \text{rank}(\mathbf{X})$ . The first extension of CUR decompositions to tensors involved a single-mode unfolding of 3-mode tensors [109]. Later, [110] proposed a different variant of tensor CUR that accounts for all modes. Recently, [98] dubbed these decompositions with more descriptive monikers, namely Fiber and Chidori CUR decompositions. In this paper, we will employ both Fiber CUR decomposition and Chidori CUR decomposition (see Figures 4.1 and 4.2 for illustration) to accelerate an essential step in the proposed algorithm. We state the Fiber CUR and Chidori CUR decomposition characterizations below for the reader’s convenience.

**Theorem 4** ([98, Theorem 3.3]). *Let  $\mathcal{A} \in \mathbb{R}^{d_1 \times \dots \times d_n}$  with Tucker rank  $(r_1, \dots, r_n)$ . Let  $I_i \subseteq [d_i]$  and  $J_i \subseteq [\prod_{j \neq i} d_j]$ . Set  $\mathcal{R} = \mathcal{A}(I_1, \dots, I_n)$ ,  $\mathbf{C}_i = \mathcal{A}_{(i)}(:, J_i)$  and  $\mathbf{U}_i = \mathbf{C}_i(I_i, :)$ . Then the following statements are equivalent:*

- (i)  $\text{rank}(\mathbf{U}_i) = r_i$ ,
- (ii)  $\mathcal{A} = \mathcal{R} \times_{i=1}^n (\mathbf{C}_i \mathbf{U}_i^\dagger)$ ,
- (iii)  $\text{rank}(\mathbf{C}_i) = r_i$  for all  $i$  and the Tucker rank of  $\mathcal{R}$  is  $(r_1, \dots, r_n)$ .

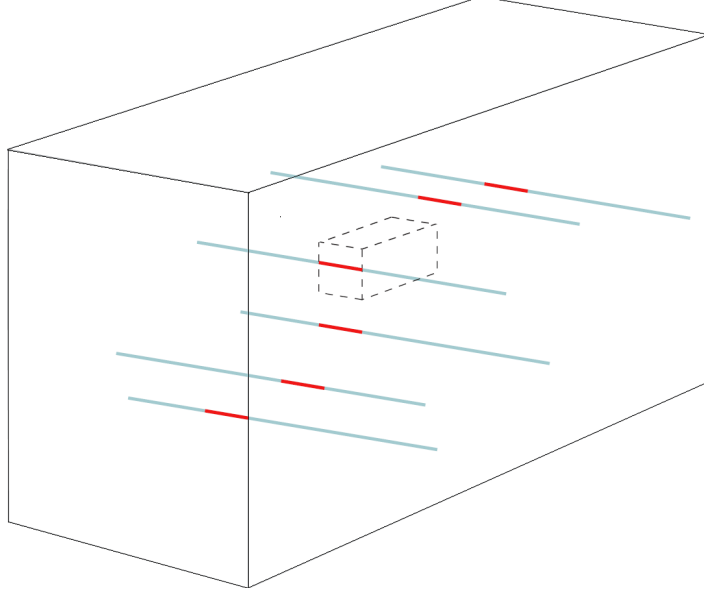


Figure 4.1: ([98, Figure 2]). Illustration of the Fiber CUR Decomposition of Theorem 4 in which  $J_i$  is not necessarily related to  $I_i$ . The lines correspond to rows of  $\mathbf{C}_2$ , and red indices correspond to rows of  $\mathbf{U}_2$ . Note that the lines may (but do not have to) pass through the core subtensor  $\mathcal{R}$  outlined by dotted lines. For the figure's clarity, we do not show fibers in  $\mathbf{C}_1$  and  $\mathbf{C}_3$ .

**Remark 5.** In particular, when  $J_i$  are sampled independently from  $I_i$ , (ii) is called Fiber CUR decomposition. When  $J_i = \otimes_{j \neq i} I_j$ , (ii) is called Chidori CUR decomposition.

In addition, according to [111, Corollary 5.2], if one uniformly samples indices  $I_i$  and  $J_i$  with size  $|I_i| = \mathcal{O}(r_i \log(d_i))$  and  $|J_i| = \mathcal{O}\left(r_i \log\left(\prod_{j \neq i} d_j\right)\right)$ , then  $\text{rank}(\mathbf{U}_i) = r_i$  holds for all  $i$  with high probability under some mild assumptions. Thus, the tensor CUR decomposition holds and its computational complexity is dominated by computing the pseudo-inverse of  $\mathbf{U}_i$ . Given the dimension of  $\mathbf{U}_i$ , the computational complexity of the pseudo-inverse of  $\mathbf{U}_i$  with Fiber sampling is  $\mathcal{O}((n-1)r^3 \log^2 d)$ , thus Fiber CUR decomposition costs  $\mathcal{O}(nr^3 \log^2 d)$ . The Chidori CUR decomposition has a slightly larger  $|J_i|$ , which is  $\prod_{j \neq i} r_i \log(d_i) = \mathcal{O}((r \log d)^{n-1})$ , thus the decomposition cost  $\mathcal{O}(r^{n+1} \log^n d)$ . By contrast, the computational complexity of HOSVD is at least  $\mathcal{O}(rd^n)$ .



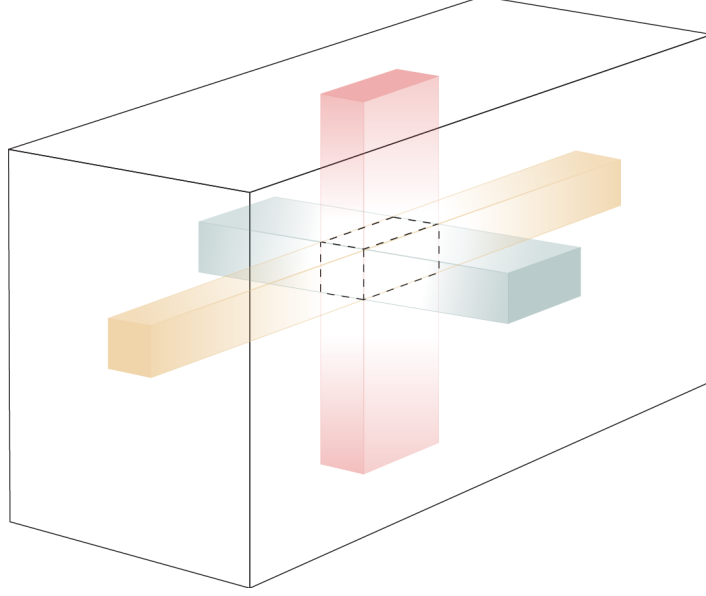


Figure 4.2: ([98, Figure 1]). Illustration of Chidori CUR decomposition of a 3-mode tensor in the case when the indices  $I_i$  are each an interval and  $J_i = \otimes_{j \neq i} I_j$  (see Theorem 4). The matrix  $\mathbf{C}_1$  is obtained by unfolding the red subtensor along mode 1,  $\mathbf{C}_2$  by unfolding the green subtensor along mode 2, and  $\mathbf{C}_3$  by unfolding the yellow subtensor along mode 3. The dotted line shows the boundaries of  $\mathcal{R}$ . In this case  $\mathbf{U}_i = \mathcal{R}_{(i)}$  for all  $i$ .

### 4.3.1 Theoretical Guarantee for Sparsity

**Definition 10** ( $\alpha$ -sparsity). A tensor  $\mathcal{S} \in \mathbb{R}^{d_1 \times \dots \times d_n}$  is  $\alpha$ -sparse if

$$\max_{k_1, \dots, k_{i-1}, k_{i+1}, \dots, k_n} \|\mathcal{S} \times_{j=1, j \neq i}^n e_{k_j}^{(j)}\|_0 \leq \alpha d_i.$$

Next, we provide one way to generate a sparse tensor that satisfies  $\alpha$ -sparsity.

**Theorem 6.** Suppose the entries of an  $n$ -order tensor  $\mathcal{S} \in \mathbb{R}^{d \times \dots \times d}$  are generated i.i.d. with  $\mathcal{S}_{i_1, \dots, i_d} = \mathcal{X}_{i_1, \dots, i_d} \mathcal{G}_{i_1, \dots, i_d}$ , where  $\mathcal{X}_{i_1, \dots, i_n}$  satisfies the Bernoulli distribution with expectation  $\frac{\alpha}{2}$  i.e.,  $\mathcal{X}_{i_1, \dots, i_n} \sim \text{Ber}(\frac{\alpha}{2})$  and  $\mathcal{G}_{i_1, \dots, i_n}$  follows a standard Gaussian distribution. Then  $\mathcal{S}$  is  $\alpha$ -sparse with probability at least  $1 - \frac{n}{d}$  provided that  $\alpha d > \frac{2n}{\log(4)-1} \log d$ .

**Proof of Theorem 6.** The support of the tensor  $\mathcal{S}$  can be labeled by the tensor

$\mathcal{X}$  by setting

$$\mathcal{X}_{i_1, \dots, i_n} = \begin{cases} 1, & \mathcal{S}_{i_1, \dots, i_n} \neq 0 \\ 0, & \text{otherwise.} \end{cases}$$

Since  $\mathcal{X}_{i_1, \dots, i_n} \sim \text{Ber}(\frac{\alpha}{2})$ , we have  $\mathbb{E}(\mathcal{X}_{i_1, \dots, i_n}) = \frac{\alpha}{2}$ . Let's first consider the sparsity of one fiber of  $\mathcal{S}$  with  $i_1, \dots, i_{j-1}, i_{j+1}, \dots, i_n \in [d]$ . According to the multiplicative Chernoff bound, we have

$$\mathbb{P}\left(\sum_{i_j=1}^d \mathcal{X}_{i_1, \dots, i_{j-1}, i_j, i_{j+1}, \dots, i_n} \geq \alpha d\right) < \left(\frac{e}{4}\right)^{\frac{\alpha d}{2}}. \quad (4.5)$$

Taking the sparsity of all fibers along all modes into account, then the probability that  $\mathcal{S}$  is  $\alpha$ -sparsity can be bounded by:

$$\begin{aligned} \mathbb{P}(\mathcal{S} \text{ is } \alpha\text{-sparse}) &\geq \left(1 - \left(\frac{e}{4}\right)^{\frac{\alpha d}{2}}\right)^{nd^{n-1}} \\ &\geq 1 - nd^{n-1} \left(\frac{e}{4}\right)^{\frac{\alpha d}{2}}. \end{aligned} \quad (4.6)$$

When  $\alpha d > \frac{2n}{\log(4)-1} \log d$ , we thus have:

$$\mathbb{P}(\mathcal{S} \text{ is } \alpha\text{-sparse}) \geq 1 - \frac{n}{d}. \quad (4.7)$$

## 4.4 Proposed Approach

In this section, we propose an efficient approach, called Robust Tensor CUR (RTCUR), for the non-convex TRPCA problem (4.4). RTCUR is developed in a framework of alternating projections: (I) First, we project  $\mathcal{X} - \mathcal{L}^{(k)}$  onto the space of sparse tensors to update the estimate of outliers (i.e.,  $\mathcal{S}^{(k+1)}$ ); (II) then we project the less corrupted data  $\mathcal{X} - \mathcal{S}^{(k+1)}$  onto the space of low-Tucker-rank tensors to update the estimate (i.e.,  $\mathcal{L}^{(k+1)}$ ). In our algorithm, the key to acceleration is using the tensor CUR decomposition for inexact low-Tucker-rank tensor approximation in Step (II), which is proved to be much more efficient than the standard HOSVD [98], in terms of computational complexity. Consequently, in Step (I), this inexact approximation allows us to estimate

only the outliers in the smaller subtensors and submatrices involved in the tensor CUR decomposition. RTCUR is summarized in Line 2. Notice that there are two variants of tensor CUR decompositions which will result in different  $J_i$  (see Remark 5), but the steps of Line 2 will remain the same. Therefore, we will not distinguish the two decomposition methods in Sections 4.4.1 and 4.4.2 when discussing the details of Step (I) and (II). We will then show the computational complexity for Line 2 with both Fiber and Chidori CUR decompositions in Section 4.4.3.

---

**Algorithm 2: Robust Tensor CUR (RTCUR)**

---

**Input** :  $\mathcal{X} = \mathcal{L}^* + \mathcal{S}^* \in \mathbb{R}^{d_1 \times \dots \times d_n}$ : observed tensor;  $(r_1, \dots, r_n)$ : underlying Tucker rank of  $\mathcal{L}^*$ ;  $\varepsilon$ : targeted precision;  $\zeta^{(0)}, \gamma$ : thresholding parameters;  $\{|I_i|\}_{i=1}^n, \{|J_i|\}_{i=1}^n$ : cardinalities for sample indices.

//  $J_i$  is defined differently for different sampling strategies. See

Section 4.4.4 for details about  $J_i$  and sampling strategies.

Set  $\mathcal{L}^{(0)} = \mathbf{0}, \mathcal{S}^{(0)} = \mathbf{0}, k = 0$ .

Uniformly sample the indices  $\{I_i\}_{i=1}^n, \{J_i\}_{i=1}^n$

**while**  $e^{(k)} > \varepsilon$  **do**

//  $e^{(k)}$  is defined in (4.14)

(Optional) Resample the indices  $\{I_i\}_{i=1}^n, \{J_i\}_{i=1}^n$  // Step (I): Updating  $\mathcal{S}$

$$\zeta^{(k+1)} = \gamma \cdot \zeta^{(k)}$$

$$\mathcal{S}^{(k+1)} = \text{HT}_{\zeta^{(k+1)}}(\mathcal{X} - \mathcal{L}^{(k)})$$

// Step (II): Updating  $\mathcal{L}$

$$\mathcal{R}^{(k+1)} = (\mathcal{X} - \mathcal{S}^{(k+1)})(I_1, \dots, I_n)$$

**for**  $i = 1, \dots, n$  **do**

$$\left[ \begin{array}{l} C_i^{(k+1)} = (\mathcal{R}^{(k+1)})_{(i)}(:, J_i) \\ U_i^{(k+1)} = \text{SVD}_{r_i}(C_i^{(k+1)}(I_i, :)) \end{array} \right.$$

$$\mathcal{L}^{(k+1)} = \mathcal{R}^{(k+1)} \times_{i=1}^n C_i^{(k+1)} \left( U_i^{(k+1)} \right)^\dagger$$

$k = k + 1$

**Output:**  $\mathcal{R}^{(k)}, C_i^{(k)}, U_i^{(k)}$  for  $i = 1, \dots, n$ : the estimates of the tensor CUR

decomposition of  $\mathcal{L}_*$

---

#### 4.4.1 Step (I): Update Sparse Component $\mathcal{S}$

We consider the simple yet effective hard thresholding operator  $\text{HT}_\zeta$  for outlier estimation. The operator is defined as:

$$(\text{HT}_\zeta(\mathcal{X}))_{i_1, \dots, i_n} = \begin{cases} \mathcal{X}_{i_1, \dots, i_n}, & |\mathcal{X}_{i_1, \dots, i_n}| > \zeta; \\ 0, & \text{otherwise.} \end{cases} \quad (4.8)$$

As shown in [12, 87, 84], with a properly chosen thresholding value,  $\text{HT}_\zeta$  is effectively a projection operator onto the support of  $\mathcal{S}^*$ . More specifically, we update

$$\mathcal{S}^{(k+1)} = \text{HT}_{\zeta^{(k+1)}}(\mathcal{X} - \mathcal{L}^{(k)}). \quad (4.9)$$

If  $\zeta^{(k+1)} = \|\mathcal{L}^* - \mathcal{L}^{(k)}\|_\infty$  is chosen, then we have  $\text{supp}(\mathcal{S}^{(k+1)}) \subseteq \text{supp}(\mathcal{S}^*)$  and  $\|\mathcal{S}^* - \mathcal{S}^{(k+1)}\|_\infty \leq 2\|\mathcal{L}^* - \mathcal{L}^{(k)}\|_\infty$ . Empirically, we find that iteratively decaying thresholding values

$$\zeta^{(k+1)} = \gamma \cdot \zeta^{(k)} \quad (4.10)$$

provide superb performance with carefully tuned  $\gamma$  and  $\zeta^{(0)}$ . Note that a favorable choice of  $\zeta^{(0)}$  is  $\|\mathcal{L}^*\|_\infty$ , which can be easily estimated in many applications. The decay factor  $\gamma \in (0, 1)$  should be tuned according to the level of difficulty of the TRPCA problem, e.g., those problems with higher rank, more dense outliers, or large condition numbers are considered to be harder. For successful reconstruction of  $\mathcal{L}^*$  and  $\mathcal{S}^*$ , the harder problems require larger  $\gamma$ . When applying RTCUR on both synthetic and real-world data, we observe that  $\gamma \in [0.6, 0.9]$  generally performs well. Since real-world data normally leads to more difficult problems, we fix  $\gamma = 0.7$  for the synthetic experiment and  $\gamma = 0.8$  for the real-world data studies in Section 4.5.

#### 4.4.2 Step (II): Update Low-Tucker-rank Component $\mathcal{L}$

SVD is the most popular method for low-rank approximation under matrix settings since SVD gives the best rank- $r$  approximation of given matrix  $\mathbf{X}$ , both with respect to the operator norm and to the Frobenius norm [112]. Similarly, HOSVD has been the standard method for low-Tucker-rank approximation under tensor settings in many works [113, 1, 112, 114]. However, the computational complexity of HOSVD is at least  $\mathcal{O}(rd^n)$ ; hence computing HOSVD is very expensive when the problem scale is large. Inspired by the recent development on tensor CUR decomposition [98], we employ tensor CUR decomposition for accelerated inexact low-Tucker-rank tensor approximations. Namely, we update the estimate of the low-Tucker-rank component  $\mathcal{L}$  by setting

$$\mathcal{L}^{(k+1)} = \mathcal{R}^{(k+1)} \times_{i=1}^n \mathbf{C}_i^{(k+1)} \left( \mathbf{U}_i^{(k+1)} \right)^\dagger, \quad (4.11)$$

where

$$\begin{aligned} \mathcal{R}^{(k+1)} &= (\mathcal{X} - \mathcal{S}^{(k+1)})(I_1, \dots, I_n), \\ \mathbf{C}_i^{(k+1)} &= (\mathcal{X} - \mathcal{S}^{(k+1)})_{(i)}(:, J_i), \\ \mathbf{U}_i^{(k+1)} &= \text{SVD}_{r_i}(\mathbf{C}_i^{(k+1)}(I_i, :)), \end{aligned} \quad (4.12)$$

#### 4.4.3 Computational Complexity

As mentioned in Section 4.3, the complexity for computing a tensor CUR decomposition is much lower than HOSVD, and the dominating steps in RTCUR are the hard thresholding operator and the tensor/matrix multiplications. For both Fiber and Chidori CUR decompositions, only the sampled subtensors and submatrices are required when computing (4.12). Thus, we merely need to estimate the outliers on these subtensors and submatrices, and (4.9) should not be fully executed. Instead, we only

compute

$$\begin{aligned}\mathcal{S}^{(k+1)}(I_1, \dots, I_n) &= \text{HT}_{\zeta^{(k+1)}}((\mathcal{X} - \mathcal{L}^{(k)})(I_1, \dots, I_n)), \\ \mathcal{S}_{(i)}^{(k+1)}(:, J_i) &= \text{HT}_{\zeta^{(k+1)}}((\mathcal{X} - \mathcal{L}^{(k)})_{(i)}(:, J_i))\end{aligned}\tag{4.13}$$

for all  $i$ . Not only can we save the computational complexity on hard thresholding but also, much smaller subtensors of  $\mathcal{L}^{(k)}$  need to be formed in (4.13). We can form the required subtensors from the saved tensor CUR components, which is much cheaper than forming and saving the whole  $\mathcal{L}^{(k)}$ .

In particular, for  $\mathcal{X} \in \mathbb{R}^{d \times \dots \times d}$ ,  $r_1 = \dots = r_n = r$  and  $|I_1| = \dots = |I_n| = \mathcal{O}(r \log d)$ , computing  $\mathcal{L}^{(k)}(I_1, \dots, I_n)$  requires  $n$  tensor-matrix product operations so the complexity for computing  $\mathcal{L}^{(k)}(I_1, \dots, I_n)$  is  $\mathcal{O}(n(r \log d)^{n+1})$  for both Fiber and Chidori CUR decompositions. The complexity for computing  $\mathcal{L}_{(i)}^{(k)}(:, J_i)$  with Fiber CUR is different from the complexity of computing  $\mathcal{L}_{(i)}^{(k)}(:, J_i)$  with Chidori CUR. With Fiber CUR, we compute each fiber in  $\mathcal{L}_{(i)}^{(k)}(:, J_i)$  independently and each fiber takes  $n$  tensor-matrix product operations. The first  $n - 1$  operations transform the  $n$ -mode core tensor  $\mathcal{L}^{(k)}(I_1, \dots, I_n)$  into a 1-mode tensor, which is a vector of length  $\mathcal{O}(r \log d)$ , and the last operation transforms this vector into another vector of length  $d$ . Since there are  $J_i = \mathcal{O}(nr \log d)$  fibers in total, the complexity for computing  $\mathcal{L}_{(i)}^{(k)}(:, J_i)$  with Fiber CUR decomposition is  $\mathcal{O}(nr \log d((r \log d)^n + dr \log d))$ . With Chidori CUR, we compute  $\mathcal{L}_{(i)}^{(k)}(:, J_i)$  as a complete unit using  $n$  tensor-matrix product operations. The first  $n - 1$  operations on the core tensor do not change its size, and the last operation changes the size of the  $i$ th mode to  $d$ . Therefore the complexity for computing  $\mathcal{L}_{(i)}^{(k)}(:, J_i)$  with Chidori CUR decomposition is  $\mathcal{O}(n(r \log d)^{n+1} + d(r \log d)^n)$ .

Moreover, for time-saving purposes, we may avoid computing the Frobenius norm of the full tensor when computing the relative error for the stopping criterion. In RTCUR, we adjust the relative error formula to be

$$e^{(k)} = \frac{\|\mathcal{E}^{(k)}(I_1, \dots, I_n)\|_{\text{F}} + \sum_{i=1}^n \|\mathcal{E}_{(i)}^{(k)}(:, J_i)\|_{\text{F}}}{\|\mathcal{X}(I_1, \dots, I_n)\|_{\text{F}} + \sum_{i=1}^n \|\mathcal{X}_{(i)}(:, J_i)\|_{\text{F}}}\tag{4.14}$$

where  $\mathcal{E}^{(k)} = \mathcal{X} - \mathcal{L}^{(k)} - \mathcal{S}^{(k)}$ , so that it does not use any extra subtensor or fiber but only those we already have. We hereby summarize the computational complexity for each step from Line 2 in Table 4.1.

If we assume that the tensor size  $d$  is comparable with or greater than  $\mathcal{O}((r \log d)^{n-1})$ , we can conclude that the total computational complexity is  $\mathcal{O}(n^2 dr^2 \log^2 d)$  for RTCUR with Fiber CUR decomposition and  $\mathcal{O}(n dr^n \log^n d)$  for RTCUR with Chidori CUR decomposition. Otherwise, the computational complexity would be  $\mathcal{O}(n^2 r^{n+1} \log^{n+1} d)$  for RTCUR with Fiber CUR, and the complexity for RTCUR with Chidori CUR remains unchanged. For all tensors tested in Section 4.5, the first case holds. Therefore, in Table 4.1, we highlighted  $\mathcal{O}(n^2 dr^2 \log^2 d)$  and  $\mathcal{O}(n dr^n \log^n d)$  as the dominating terms.

Table 4.1: Computational complexity for each step from Line 2. The complexity for computing  $\mathcal{S}(\dots)$ ,  $\mathcal{R}$  are the same as their size; the complexity for  $\mathbf{C}_i \mathbf{U}_i^\dagger$  is introduced in Section 4.3; the complexity for computing  $\mathcal{L}(\dots)$  and error term is introduced in Section 4.4.3

COMPUTATIONAL COMPLEXITY	FIBER SAMPLING	CHIDORI SAMPLING
Sparse subtensor $\mathcal{S}(I_1, \dots, I_n)$ or $\mathcal{R}$	$\mathcal{O}(r^n \log^n d)$	$\mathcal{O}(r^n \log^n d)$
All $\mathcal{S}_{(i)}(:, J_i)$ or $\mathbf{C}_i$ for $n$ modes	$\mathcal{O}(n^2 r d \log d)$	$\mathcal{O}(n r^{n-1} d \log^{n-1} d)$
All $\mathbf{U}_i^\dagger$ for $n$ modes	$\mathcal{O}(n^2 r^3 \log^2 d)$	$\mathcal{O}(n r^{n+1} \log^n d)$
All $\mathbf{C}_i \mathbf{U}_i^\dagger$ for $n$ modes	<b><math>\mathcal{O}(n^2 r^2 d \log^2 d)</math></b>	<b><math>\mathcal{O}(n r^n d \log^n d)</math></b>
Low-rank subtensor $\mathcal{L}(I_1, \dots, I_n)$	$\mathcal{O}(n r^{n+1} \log^{n+1} d)$	$\mathcal{O}(n r^{n+1} \log^{n+1} d)$
All $\mathcal{L}_{(i)}(:, J_i)$ for $n$ modes	$\mathcal{O}(n^2 r^{n+1} \log^{n+1} d + \mathbf{n}^2 r^2 d \log^2 d)$	$\mathcal{O}(n^2 r^{n+1} \log^{n+1} d + n r^n d \log^n d)$
Error term $\mathcal{E}^{(k)}(I_1, \dots, I_n)$ and $\mathcal{E}_{(i)}^{(k)}(:, J_i)$	$\mathcal{O}(r^n \log^n d + n^2 dr \log d)$	$\mathcal{O}(d r^{n-1} \log^{n-1}(d))$

#### 4.4.4 Four Variants of RTCUR

In Section 4.3, we discussed two versions of tensor CUR decomposition: Fiber CUR decomposition and Chidori CUR decomposition. Each of the decomposition methods could derive two slightly different RTCUR algorithms depending on if we fix sample indices through all iterations (see Line 2). As a result of this, we obtain four variants in total. We give different suffixes for each variant of RTCUR algorithm: RTCUR-FF, RTCUR-FC, RTCUR-RF, and RTCUR-RC. We will showcase experimental results for all variants in Section 4.5. The first letter in the suffix indicate whether we fix the sample indices through all iterations: ‘F’ stands for ‘fix’, where the variant uses

fixed sample indices through all iterations; ‘R’ stands for ‘resampling’, where the variant resamples  $\{I_i\}_{i=1}^n$  and  $\{J_i\}_{i=1}^n$  in each iteration. The second letter indicate which type of CUR decomposition we use in RTCUR. ‘F’ represents that RTCUR is derived from Fiber CUR decomposition and ‘C’ stands for Chidori CUR. For Fiber CUR, the amount of fibers to be sampled refers to [111, Corollary 5.2], i.e.,  $|I_i| = vr_i \log(d_i)$ ,  $|J_i| = vr_i \log\left(\prod_{j \neq i} d_j\right)$  and  $J_i$  is sampled independently from  $I_i$ . Here,  $v$  denotes the sampling constant, a hyper-parameter that will be tuned in the experiments. For Chidori CUR,  $|I_i| = vr_i \log(d_i)$  and  $J_i = \otimes_{j \neq i} I_j$ . Of these four variants, RTCUR-FF requires minimal data accessibility and runs slightly faster than other variants. The resampling variants access more data and take some extra computing; for example, the denominator of (4.14) has to be recomputed per iteration. However, accessing more redundant data means resampling variants have better chances of correcting any “unlucky” sampling over the iterations. Thus, we expect resampling variants to have superior outlier tolerance than fixed sampling variants. Additionally, the fixed sampling variants have an efficiency advantage over the resampling variants under specific conditions (e.g., when re-accessing the data is expensive).

The difference between Chidori variants and Fiber variants has similar properties: if we choose the same  $v$  and let  $|I_i| = vr_i \log(d_i)$  for both Chidori and Fiber CUR described in Section 4.3, the Chidori variants generally access more tensor entries compared to the Fiber variants. Therefore, with the same sampling constant  $v$ , Chidori variants requires more computing time in each iteration. Nevertheless, Chidori variants can tolerate more dense outliers with these extra entries than Fiber variants. We will further investigate their computational efficiency and practical performance in Section 4.5.

**Remark 7.** [98, Algorithm 3] The tensor CUR decomposition represented in Theorem 4 (ii) is also in Tucker decomposition form. We can efficiently convert the tensor CUR decomposition to HOSVD with Algorithm 3 [98].



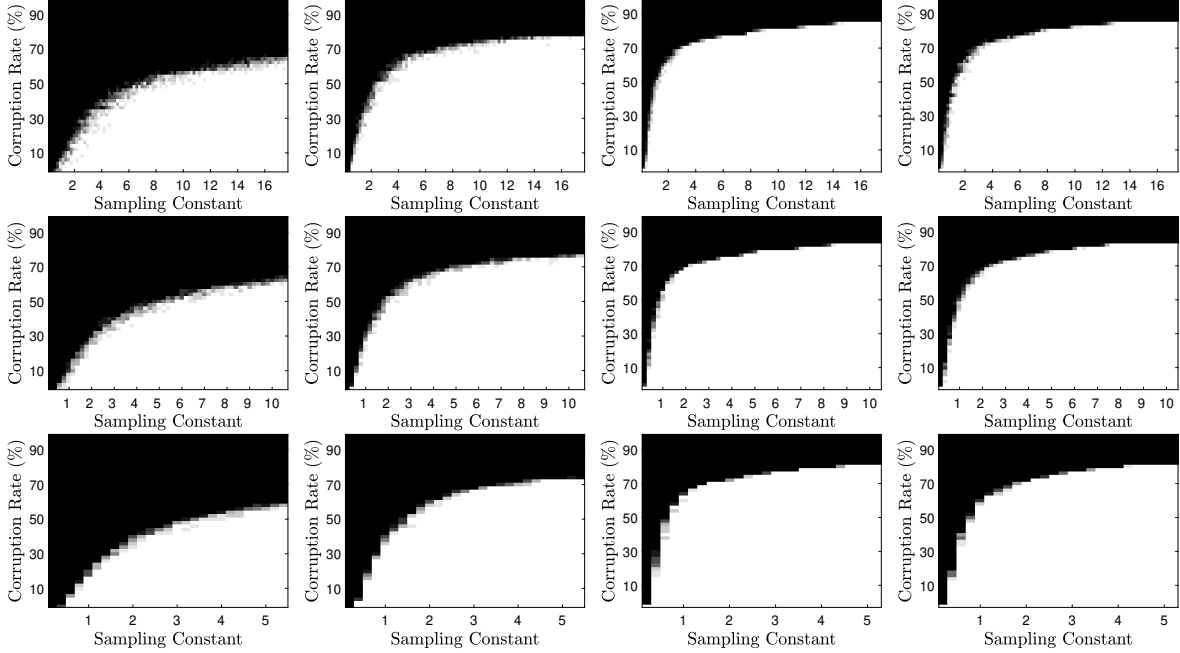


Figure 4.3: Empirical phase transition in corruption rate  $\alpha$  and sampling constant  $v$ . Left to Right: RTCUR-FF, RTCUR-RF, RTCUR-FC, RTCUR-RC, **Top:**  $r = 3$ . **Middle:**  $r = 5$ . **Bottom:**  $r = 10$ .

---

**Algorithm 3:** Conversion from CUR to HOSVD

---

**Input** :  $\mathcal{R}, \mathbf{C}_i, \mathbf{U}_i$  : CUR decomposition of the tensor  $\mathcal{A}$

$[\mathbf{Q}_i, \mathbf{R}_i] = \text{qr}(\mathbf{C}_i \mathbf{U}_i^\dagger)$  for  $i = 1, \dots, n$

$\mathcal{T}_1 = \mathcal{R} \times_1 \mathbf{R}_1 \times_2 \cdots \times_n \mathbf{R}_n$

Compute HOSVD of  $\mathcal{T}_1$  to find  $\mathcal{T}_1 = \mathcal{T} \times_1 \mathbf{V}_1 \times_2 \cdots \times_n \mathbf{V}_n$

**Output:**  $[[\mathcal{T}; \mathbf{Q}_1 \mathbf{V}_1, \dots, \mathbf{Q}_n \mathbf{V}_n]]$ : HOSVD decomposition of  $\mathcal{A}$

---

In contrast, converting HOSVD to a tensor CUR decomposition is not as straightforward.

## 4.5 Numerical Experiments

In this section, we conduct numerical experiments to compare the empirical performance of RTCUR with the state-of-the-art robust matrix/tensor PCA algorithms: Riemannian gradient descent (RGD) [101], alternating direction method of multipliers (ADMM) [93], accelerated alternating projections (AAP) [12], and iterative robust

CUR (IRCUR) [87]. Of the mentioned algorithms, RGD and ADMM are designed for the TRPCA task, whereas AAP and IRCUR are designed for the traditional matrix RPCA task.

In each subsection, we evaluate all four proposed variants, RTCUR-FF, RTCUR-RF, RTCUR-FC, and RTCUR-RC, except that the network clustering experiment only uses fixed sampling (RTCUR-FF and RTCUR-FC). The latter choice for the network clustering task was made because the coauthorship network is very sparse, and resampling variants diminish the core tensor with a more significant probability.

The rest of this section is organized as follows: Section 4.5.1 contains two synthetic experiments. In particular, Section 4.5.1.1 studies the empirical relation between the outlier tolerance and sample size for RTCUR and Section 4.5.1.2 shows the speed advantage of RTCUR compared to the state-of-the-art methods. In Sections 4.5.2 and 4.5.3, we apply RTCUR to two real-world problems, color video background subtraction and face modeling. In Section 4.5.4, we apply RTCUR on network clustering applications and discuss the results.

We utilize the codes of all compared algorithms from the authors' websites, and the parameters are hand-tuned for their best performance. For RTCUR, we sample  $|I_i| = vr_i \log(d_i)$  (and  $|J_i| = vr_i \log\left(\prod_{j \neq i} d_j\right)$  for Fiber variants) for all  $i$ , and  $v$  is called the *sampling constant* through this section. All the tests are executed from Matlab R2020a on an Ubuntu workstation with an Intel i9-9940X CPU and 128GB RAM. The related codes are provided in <https://github.com/huangl3/RTCUR>.

### 4.5.1 Synthetic Examples

For the synthetic experiments, we use  $d := d_1 = \dots = d_n$  and  $r := r_1 = \dots = r_n$ . The observed tensor  $\mathcal{X}$  is composed as  $\mathcal{X} = \mathcal{L}^* + \mathcal{S}^*$ . To generate  $n$ -mode  $\mathcal{L}^* \in \mathbb{R}^{d \times \dots \times d}$  with Tucker rank  $(r, \dots, r)$ , we take  $\mathcal{L}^* = \mathcal{Y} \times_1 Y_1 \times_2 \dots \times_n Y_n$  where  $\mathcal{Y} \in \mathbb{R}^{r \times \dots \times r}$  and  $\{Y_i \in \mathbb{R}^{d \times r}\}_{i=1}^n$  are Gaussian random tensor and matrices with standard normal entries. To generate the sparse outlier tensor  $\mathcal{S}^*$ , we uniformly sample  $\lfloor \alpha d^n \rfloor$  entries

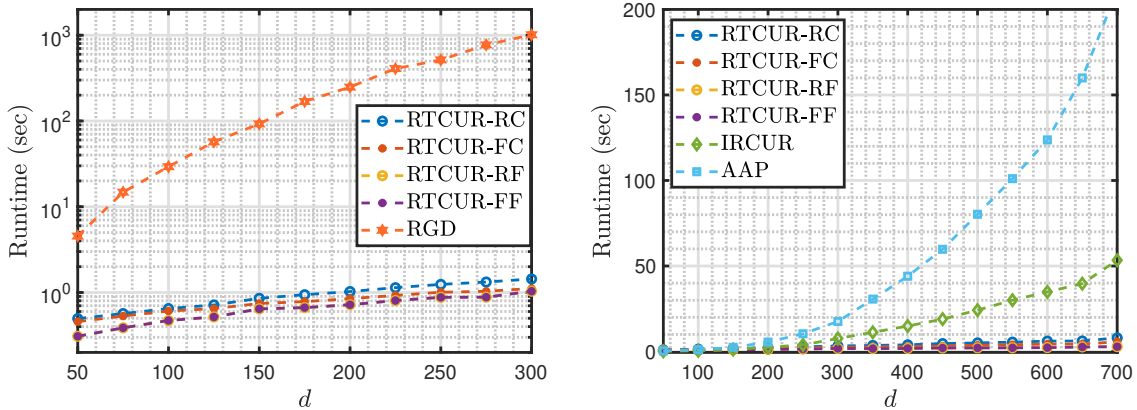


Figure 4.4: Runtime vs. dimension comparison among RTCUR-F, RTCUR-R, GD, AAP, and IRCUR on tensors with size  $d \times d \times d$  and Tucker rank  $(3, 3, 3)$ . The RGD method proceeds relatively slowly for larger tensors, so we only test the RGD runtime for tensors with a size smaller than 300 for each mode.

to be the support of  $\mathcal{S}^*$  and the values of the non-zero entries are uniformly sampled from the interval  $[-\mathbb{E}(|\mathcal{L}_{i_1, \dots, i_n}^*|), \mathbb{E}(|\mathcal{L}_{i_1, \dots, i_n}^*|)]$ .

#### 4.5.1.1 Phase Transition

We study the empirical relation between the outlier corruption rate  $\alpha$  and sampling constant  $\nu$  for all four variants of RTCUR. The tests are conducted on  $300 \times 300 \times 300$  (i.e.,  $n = 3$  and  $d = 300$ ) problems with  $r = 3, 5$ , or  $10$ . For all the variants, the thresholding parameters are set to be  $\zeta^{(0)} = \|\mathcal{L}\|_\infty$  and  $\gamma = 0.7$ . The stopping condition is  $e^{(k)} < 10^{-5}$  and an example is considered successfully solved if  $\|\mathcal{L}^* - \mathcal{L}^{(k)}\|_F / \|\mathcal{L}^*\|_F \leq 10^{-3}$ . For each pair of  $\alpha$  and  $\nu$ , we generate 10 test examples.

We summarize the experimental results in Figure 4.3, where a white pixel means all 10 test examples are successfully solved under the corresponding problem parameter setting, and a black pixel means all 10 test cases fail. First, one can observe that the Chidori variants, RTCUR-FC, and RTCUR-RC, can recover the low-rank tensor with higher outlier rate than the Fiber variants with the same sampling constant  $\nu$ . This behavior is as expected because with the same  $\nu$ , Chidori sampling will access more data from the tensor and hence performs more stable than Fiber sampling. On the other

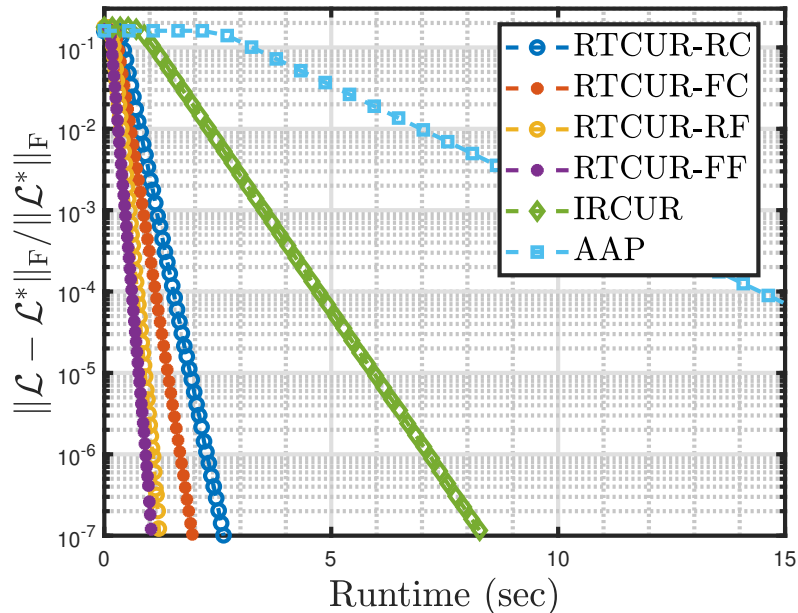


Figure 4.5: Runtime vs. relative error comparison among RTCUR-F, RTCUR-R, AAP, and IRCUR on tensor with size  $500 \times 500 \times 500$  and Tucker rank  $(3, 3, 3)$ .

hand, RTCUR-RF has slightly better performance than RTCUR-FF as resampling the fibers will gain access to more tensor entries during the whole process. Also, we notice that smaller  $r$  tolerates more outliers since a larger  $r$  leads the TRPCA task to a more complex problem. Taking larger  $v$  always provides better outlier tolerance. However, larger  $v$  means RTCUR needs to sample larger subtensors and more fibers, which leads to more computational time. According to this experiment, we recommend  $v \in [3, 5]$  to balance outlier tolerance and speed, and this constant could be further tuned according to the expected outlier rate for the task.

#### 4.5.1.2 Computational Efficiency

In this section, we compare the computational efficiency between RTCUR and the state-of-the-art tensor/matrix RPCA algorithms mentioned in Section 4.5. For matrix methods, we first unfold an  $n$ -mode  $d \times \dots \times d$  tensor to a  $d \times d^{m-1}$  matrix, then solve the matrix RPCA problem with rank  $r$ . Because the target tensor rank for ADMM is tubal-rank instead of tucker-rank [93], we do not apply ADMM to this fixed-tucker-

Table 4.2: Video information and runtime comparison (in seconds) for color video background subtraction task.

	VIDEO SIZE	RTCUR-FF	RTCUR-RF	RTCUR-FC	RTCUR-RC	ADMM	AAP	IRCUR
<i>Shoppingmall</i>	$256 \times 320 \times 3 \times 1250$	<b>3.53</b>	5.83	10.68	10.75	783.67	50.38	15.71
<i>Highway</i>	$240 \times 320 \times 3 \times 440$	<b>3.15</b>	5.47	6.80	7.55	168.55	18.10	3.87
<i>Crossroad</i>	$350 \times 640 \times 3 \times 600$	<b>6.15</b>	13.33	8.46	12.01	1099.3	97.85	35.47
<i>Port</i>	$480 \times 640 \times 3 \times 1000$	<b>11.04</b>	18.34	26.63	27.93	2934.3	154.30	71.64
<i>Parking-lot</i>	$360 \times 640 \times 3 \times 400$	<b>3.79</b>	4.52	6.62	8.14	854.50	34.70	17.38

rank experiment, though it was included in our previous work [115]. For all tests, we set the underlying low-rank component having tucker-rank  $[3, 3, 3]$  and add 20% corruption. We set parameters  $\nu = 3$ ,  $\zeta^{(0)} = \|\mathcal{L}\|_\infty$ ,  $\gamma = 0.7$  for all four variants of RTCUR. The reported runtime is averaged over 20 trials.

In Figure 4.4, we consider the problem of 3-mode TRPCA with varying dimension  $d$  and compare the total runtime (all methods halt when relative error  $e^{(k)} < 10^{-5}$ ). One can see that all variants of RTCUR are substantially faster than the compared algorithms when  $d$  is large.

In Figure 4.5, we evaluate the convergence behavior of the tested algorithms. We exclude RGD and ADMM because running them on this experiment is too expensive. We find all the tested algorithms converge linearly, and variants of RTCUR run the fastest. Moreover, as discussed in Section 4.4.4, the Fiber sampling has a running time advantage over Chidori sampling with the same sampling constant, and fixed sampling runs slightly faster than re-sampling in all tests.

#### 4.5.2 Color Video Background Subtraction

We apply the four variants of RTCUR and aforementioned tensor/matrix RPCA algorithms on the color video background subtraction task. We obtain 5 color video datasets from various sources: *Shoppingmall* [116], *Highway* [117], *Crossroad* [117], *Port* [117], and *Parking-lot* [118].

Since a monochromatic frame usually does not have low rank structure [119], we vectorize each color channel of each frame into a vector and construct a (height  $\cdot$  width)  $\times 3 \times$  frames tensor. The targeted Tucker rank is  $\mathbf{r} = (3, 3, 3)$  for all videos.

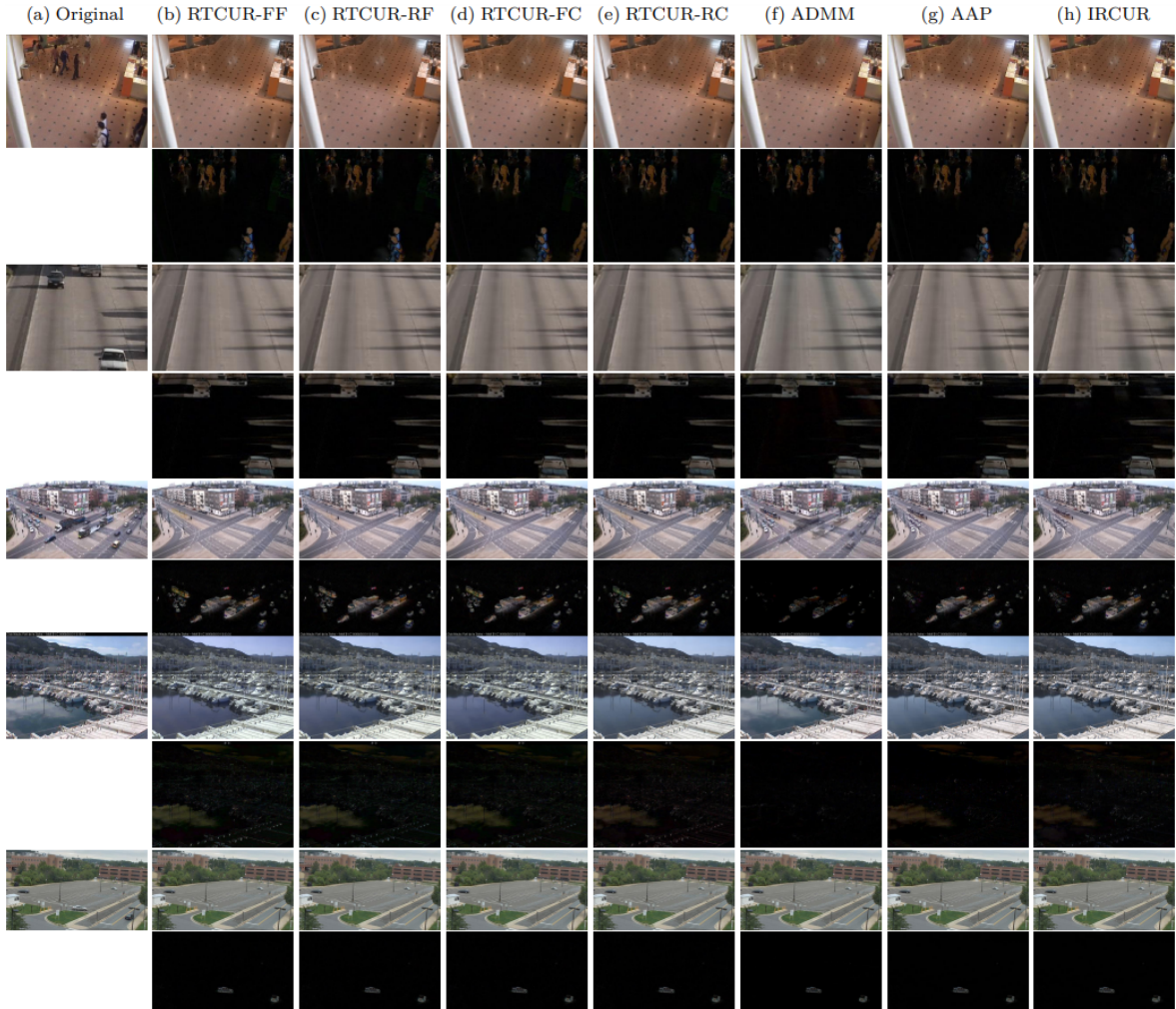


Figure 4.6: Visual results for color video background subtraction. The **first two rows** are separated backgrounds and foregrounds corresponding to a frame from *Shopping-mall*, the **3rd and 4th rows** are separated backgrounds and foregrounds corresponding to a frame from *Highway*, the **5th and 6th** are separated backgrounds and foregrounds corresponding to a frame from *Crossroad*, **7th and 8th** correspond to a frame from *Port*, and the **last two rows** correspond to a frame from *Parking lot*, except the first column which is the original frame.

For those matrix algorithms, including AAP and IRCUR, we unfold the tensor to a  $(\text{height} \cdot \text{width}) \times (3 \cdot \text{frames})$  matrix and rank 3 is used. We set RTCUR parameters  $v = 2$ ,  $\zeta^{(0)} = 255$ ,  $\gamma = 0.7$  in this experiment. We exclude RGD from this experiment because the disk space required for RGD exceeds our server limit.

Among the tested videos, *Shoppingmall*, *Highway*, and *Parking-lot* are normal speed videos with almost static background. *Crossroad* and *Port* are outdoor time-lapse

videos, hence their background colors change slightly between different frames. We observe that all tested algorithms perform very similar for videos with static background and produce visually desirable output. On the other hand, the color of the extracted background varies slightly among different algorithms on time-lapse videos. Since the color of the background keeps changing slightly for the time-lapse videos, we cannot decide the ground-truth color of the background, hence we do not rank the performance of different algorithms. The runtime for results along with video size information are summarized in Table 4.2. By comparing the runtime of four variants of RTCUR, we can observe that the experiment result generally agrees with the analysis on computational efficiency in Section 4.4.4. All RTCUR variants accomplish the background subtraction task faster than the guideline methods. In addition, we provide some selected visual results in Figure 4.6.

### 4.5.3 Robust Face Modeling

In this section, we use the UT Dallas database [120] for the task of robust face modeling. This dataset contains a face speech video that lasts approximately 5 seconds. The resolution of the video is  $360 \times 540$  and 40 non-successive frames are extracted. We mark out a monochromatic square on different color channels for 10 distinct frames per color. Similar to Section 4.5.2, we vectorize each frame and construct a  $(\text{height} \cdot \text{width}) \times 3 \times \text{frame}$  data tensor. The tensor is unfolded to a  $(\text{height} \cdot \text{width}) \times (3 \cdot \text{frame})$  matrix for matrix RPCA methods. We use Tucker rank  $(3, 3, 3)$  for tensors methods and rank 3 for matrix methods. Figure 4.7 presents the test examples and visual results; Table 4.3 summarizes the runtime for each method applied on this task. Notice that the matrix methods fail to detect the monochromatic outlier blocks since they lose the structural connection between color channels after matricization, albeit they spend less time on this task. In contrast, all variants of RTCUR successfully detect the outlier blocks. The other two TPRCA methods, ADMM and RGD, partially detect the outlier blocks.

Table 4.3: Runtime comparison (in seconds) for face modeling task. The matrix RPCA approaches (AAP and IRCUR) meet the termination condition earlier with the unfolded tensor, but they failed to detect the artificial noise in this task (see Figure 4.7).

METHOD	RUNTIME	METHOD	RUNTIME
RTCUR-FF	2.247	ADMM	30.61
RTCUR-RF	2.289	AAP	1.754
RTCUR-FC	2.319	IRCUR	1.307
RTCUR-RC	2.701	RGD	1430.8

#### 4.5.4 Network Clustering

In this section, we apply our RTCUR algorithm, and the TRPCA algorithm from [101], Riemannian gradient descent (RGD), for the community detection task on the co-authorship network data from [121] and compare their results and efficiency. This dataset contains 3248 papers with a list of authors for each paper (3607 authors in total); hence could naturally serve as the adjacency matrix for the weighted co-authorship graph. The original paper for this dataset tests a number of community detection algorithms, including network spectral clustering, profile likelihood, pseudo-likelihood approach, etc., on a selected subset with 236 authors and 542 papers. To obtain this subset, we generate an undirected graph for the 3607 authors and put an edge between two authors if they have co-authored two or more papers. We then take the largest connected component, and all the nodes in this component are the subset of authors we are interested in. Since we aim to explore the higher-order interactions among this co-authorship network, we convert this connected component into a 3-mode adjacency tensor  $\mathcal{T}$  and apply TRPCA algorithms to obtain the low-rank component from  $\mathcal{L}$ . We construct the adjacency tensor  $\mathcal{T}$  with the following rules:

- For any two connected authors  $(i, j)$ , which means author  $i$  and  $j$  have worked together for at least two papers, we set  $\mathcal{T}_{\mathfrak{S}(i,i,j)} = \mathcal{T}_{\mathfrak{S}(i,j,j)} = 1$
- For any three pairwise connected authors  $(i, j, k)$ , we set  $\mathcal{T}_{\mathfrak{S}(i,j,k)} = 1$ . Notice that these three authors may not appear in one paper at the same time, but each pair of them have worked together for at least two papers.



Here  $\mathfrak{S}(S)$  denotes all permutations of the set  $S$ . Therefore the adjacency tensor  $\mathcal{T}$  is symmetric. Now we apply RTCUR with different sampling constants as well as the TRPCA algorithm RGD [101] to learn the low-rank component  $\mathcal{L}$  with Tucker rank  $(4, 4, 4)$ , which is used to infer the communities in this network. Then we apply the SCORE algorithm [121] as the clustering algorithm on the low-rank tensor  $\mathcal{L}$ . We use SCORE instead of other traditional clustering algorithms such as spectral clustering because SCORE could mitigate the influence of node heterogeneity [121]. We plot the results from each TRPCA algorithm in Figure 4.8.

The clustering of the “High-Dimensional Data Analysis” co-authorship network is an unsupervised task, which means the ground truth of labeling an author with a certain community does not exist. Therefore, we do not focus on qualitatively evaluating each result, but we present the new findings from higher-order interactions among co-authors and analyze the results from different choices of parameters. Previous studies on this co-authorship network generally provide three clusters with names: “Carroll–Hall” group, “North Carolina” community, and “Fan and others” group [121, 101]. Among them, “Carroll–Hall” group generally includes researchers in nonparametric and semi-parametric statistics, functional estimation, and high-dimensional statistics; “North Carolina” group generally includes researchers from Duke University, University of North Carolina, and North Carolina State University; “Fan and Others” group includes primarily the researchers collaborating closely with Jianqing Fan or his co-authors, and other researchers who do not obviously belong to the first two groups [101]. For conciseness, we will make use of the same name of each group as in previous studies on this co-authorship network.

The top two plots of Figure 4.8 are existing results from [101]. With SCORE as the clustering method, the original tensor and the RGD output both successfully reveal the two groups: the “Carroll–Hall” group and a “North Carolina” community, with slightly different clustering result for researchers who do not obviously belong to one group, such as Debajyoti Sinha, Michael J Todd, and Abel Rodriguez. One can observe that Fiber RTCUR detected the “Carroll–Hall” group. However, Fiber RTCUR labels

most authors not having a strong connection with Peter Hall, Raymond Carroll, and Jianqing Fan as the “North Carolina” community. Similarly, Chidori RTCUR with  $v = 2$  generates the center of the “Carroll–Hall” group as one cluster and categorizes most authors with a lower number of co-authorships and not co-authored with kernel members into the “Fan and others” group. We infer that the tendency to cluster most members into one group is due to insufficient sampling. The co-authorship tensor is very sparse, with only about 2% entries being non-zero, so the feature of each node may not be sufficiently extracted from Fiber sampling or Chidori sampling with small  $v$ . From the middle two plots, we can observe that most authors in the largest group have very close first and second principal components in the two-dimension embedding, providing the evidence that the algorithm ignored some non-zeros entries for nodes with fewer numbers of connections during the sampling process.

Note that the sampling constant of Fiber sampling should be  $r \log d$  times the constant of Chidori sampling in order to access the same amount of data from the original tensor, where  $d$  denotes the number of authors in this experiment). So we only test the Chidori sampling with a larger sampling constant on the co-authorship tensor  $\mathcal{L}$  for efficiency. The result is shown in the bottom:  $v = 6$  for bottom-left and  $v = 11$  for bottom-right of Figure 4.8. Both settings generate the “Carroll–Hall” group with authors having strong ties to Peter Hall and Raymond Carroll, such as Richard Samworth, Hans-Georg Muller, Anastasios Tsiatis, Yuanyuan Ma, Yuedong Wang, Lan Zhou, etc. The “Fan and others” is also successfully detected, including co-authors of high-degree node Jianqing Fan such as Hua Liang, Yingying Fan, Haibo Zhou, Yong Zhou, Jiancheng Jiang, Qiwei Yao, etc. The sizes of the three clusters generated from these two settings are more balanced than the result from RTCUR with smaller  $v$ . Therefore we can conclude that, in this real-world network clustering task, different choices of sampling constant provide the same core members of each group, and the group size is more balanced with a larger amount of sampling data at the cost of computation efficiency. Table 4.4 shows the runtime for each algorithm and sampling method.

Table 4.4: Runtime comparison (in seconds) of TRPCA algorithms: RGD and RTCUR.

METHOD	RUNTIME
RGD [122]	120.5
RTCUR-FF with $v = 6$	3.526
RTCUR-FC with $v = 2$	0.571
RTCUR-FC with $v = 6$	4.319
RTCUR-FC with $v = 11$	7.177

## 4.6 Conclusion and Future Work

This paper presents a highly efficient algorithm RTCUR for large-scale TRPCA problems. RTCUR is developed by introducing a novel inexact low-Tucker-rank tensor approximation via Fiber CUR decomposition, whose structure significantly reduces computational complexity. Specifically, RTCUR has per iteration computational complexity  $\mathcal{O}(n^2d(r \log d)^2)$  with Fiber CUR and  $\mathcal{O}(nd(r \log d)^n)$  with Chidori CUR, compared to the minimum of  $\mathcal{O}(rd^n)$  for HOSVD-based algorithms. Numerical experiments on synthetic and real-world data sets also demonstrate the efficiency advantage of RTCUR against other state-of-the-art tensor/matrix RPCA algorithms. Additionally, the fixed sampling variants of RTCUR only require partial information from the input tensor for the TRPCA task.

We argue for three lines of future research. First, it will be essential to investigate a theoretical convergence guarantee of RTCUR. Second, it will be feasible to extend RTCUR to the partially observed setting (i.e., when only a small random portion of the tensor could be observed). Third, it is worth studying the stability of RTCUR with dense but small additive perturbations.

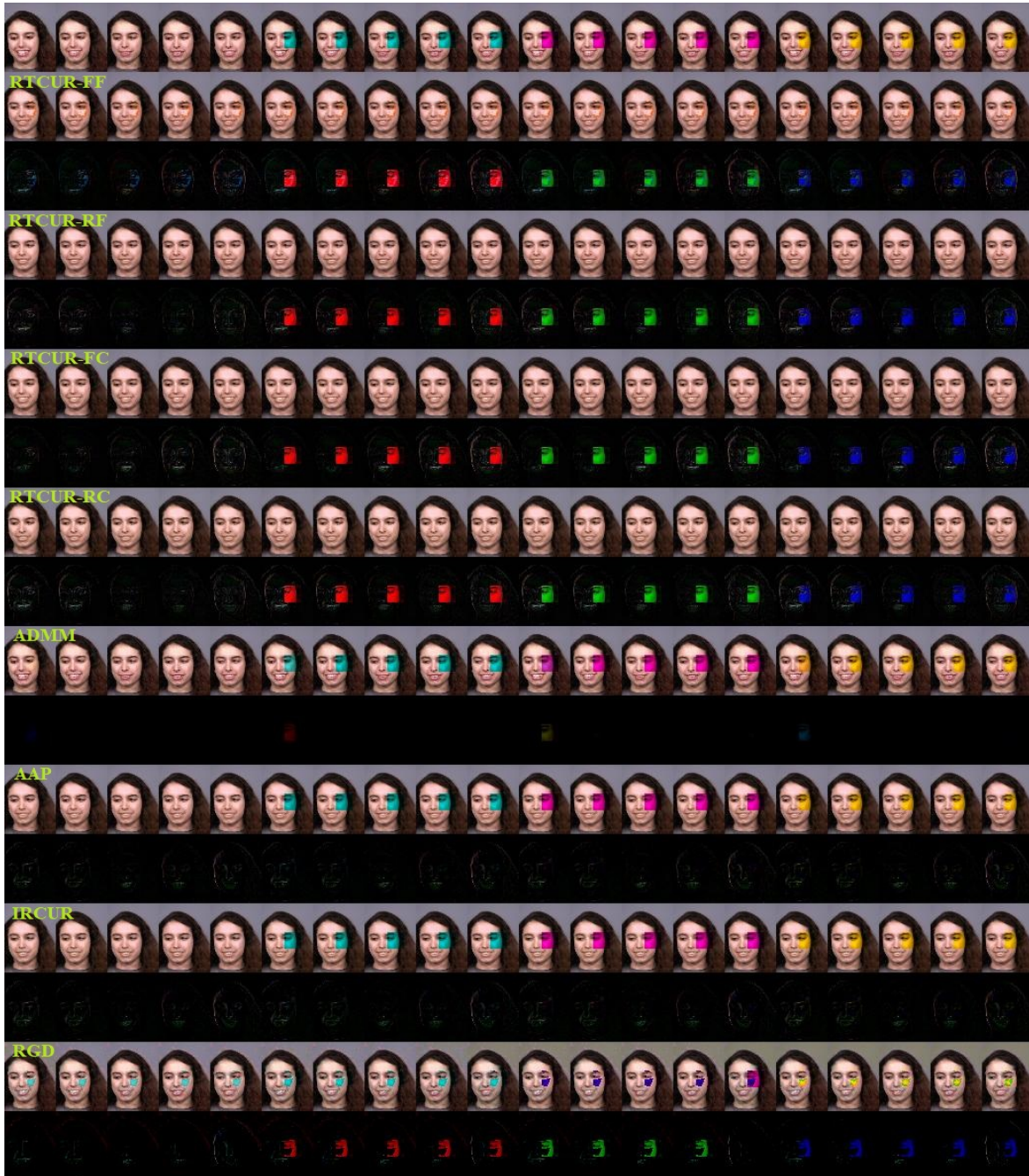


Figure 4.7: Visual results for robust face modeling. The **top** row contains the corrupted faces, the **second and third** rows are the recovered faces and detected outliers outputted by RTCUR-FF; the **fourth and fifth** rows are results from RTCUR-RF; the **sixth and seventh** rows are results from RTCUR-FC; the **eighth and ninth** rows are results from RTCUR-RC; the **tenth and eleventh** rows are results from ADMM; the **twelfth and thirteenth** rows are results from AAP; the **fourteenth and fifteenth** rows are results from IRCUR; the **sixteenth and seventeenth** rows are results from RGD.

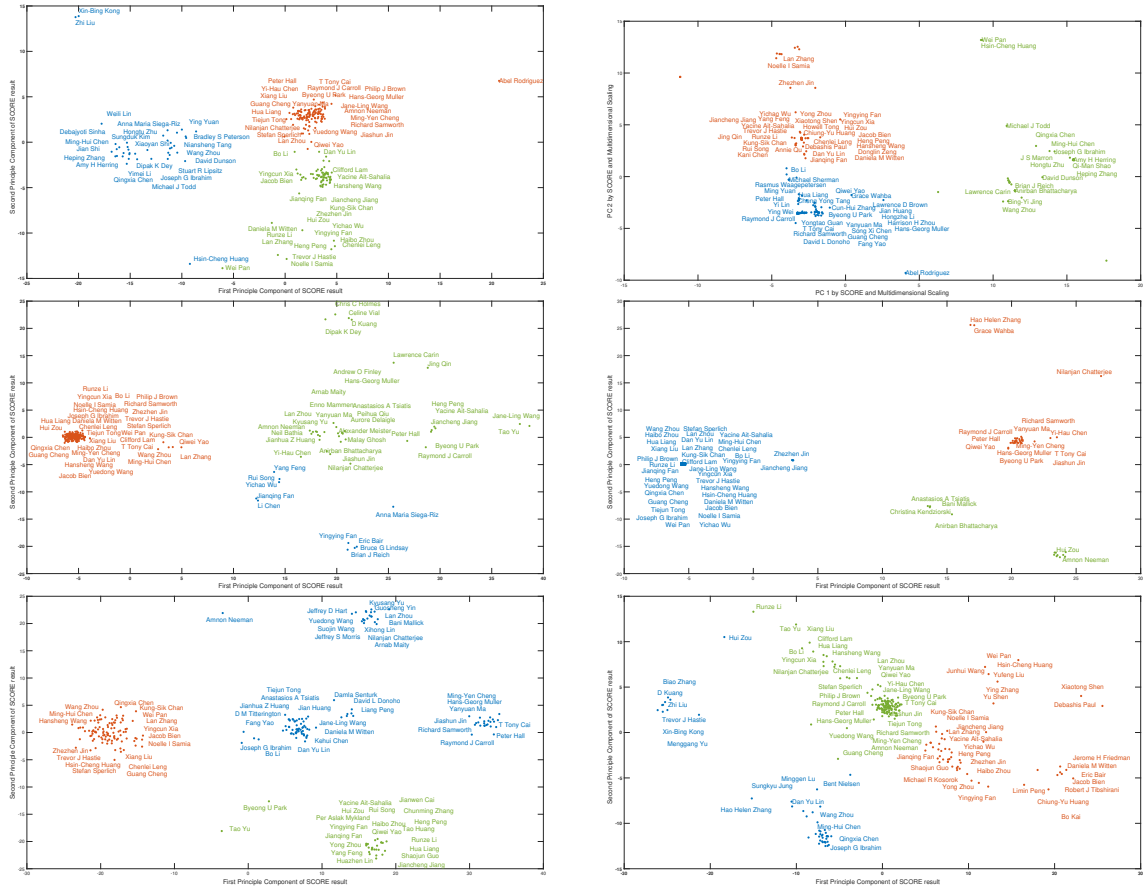


Figure 4.8: Three communities detected in the “High-Dimensional Data Analysis” co-authorship network with SCORE[121], RGD [101], and RTCUR . **Top left:** Result from SCORE on original tensor; **top right:** Result from RGD and SCORE; **middle left:** Result from RTCUR-FF and SCORE, with  $v = 6$ ; **middle right:** Result from RTCUR-FC and SCORE, with  $v = 2$ ; **bottom left:** Result from RTCUR-FC and SCORE, with  $v = 6$ ; **bottom right:** Result from RTCUR-FC and SCORE, with  $v = 11$ . All TRPCA methods are applied on the adjacency tensor  $\mathcal{T}$  with Tucker rank  $= (4, 4, 4)$ .

## CHAPTER 5

### Inference of Media Bias and Content Quality

The content of this chapter is from my previous paper *Inference of Media Bias and Content Quality Using Natural-Language Processing*. The co-authors are Denali Moli-tor, Deanna Needell, and Mason A. Porter. I performed computational experiments, formulated and implemented methods, analyzed and interpreted the output of these methods, and wrote and edited the manuscript. Denali conducted initial computa-tional experiments, formulated and implemented methods, and edited the manuscript. Deanna formulated and developed the project, advised on the project, and wrote and edited the manuscript. Mason formulated and developed the project, advised on the project, and wrote and edited the manuscript.

In this chapter, we present a quantitative framework to infer both political bias and content quality of media outlets from text, and we illustrate this framework with empirical experiments with real-world data. We apply a bidirectional long short-term memory (LSTM) neural network to a data set of more than 1 million tweets to generate a two-dimensional ideological-bias and content-quality measurement for each tweet. We then infer a “media-bias chart” of (bias, quality) coordinates for the media outlets by integrating the (bias, quality) measurements of the tweets of the media outlets. We also apply a variety of baseline machine-learning methods, such as a naive-Bayes method and a support-vector machine (SVM), to infer the bias and quality values for each tweet. All of these baseline approaches are based on a bag-of-words approach. We find that the LSTM-network approach has the best performance of the examined methods. Our results illustrate the importance of leveraging word order into machine-learning methods in text analysis.

## 5.1 Introduction

Mass media is a fundamental part of modern society. Media outlets provide windows to the world and influence public knowledge, attitudes, and behavior [123]. They disseminate news and information, help educate the public, provide entertainment, and influence the spread of ideologies and opinions [124]. However, media outlets have biases (including potentially very strong ones), and their influential societal roles make it important to examine such biases [125, 126]. Biased ideologies can impact people’s choices (e.g., through their attitudes on topics like abortion [127]), their sharing behavior on social media [128], and more. Additionally, media can exacerbate political polarization by intensifying or even creating ideological “echo chambers” [129, 130] and enhancing so-called “pernicious polarization” [131], which divides societies into “Us versus Them” camps along a focal dimension of difference that overshadows other similarities and differences. There is a long history of quantitative studies of voting blocs and ideological biases of politicians [132, 133, 134]. Researchers have quantitatively examined the ideological biases of politicians in a variety of situations, including in social media [135, 136, 137, 138, 139, 140] and in television interviews [141]. Media outlets help broadcast the messages of public figures (such as politicians), which, in turn, influences public biases and opinions. Moreover, the quantitative study of the ideologies of politicians also helps guide investigations of the political biases of other entities, such as private citizens and media outlets, and one can estimate the ideological positions of individuals based on the media outlets with which they engage [142].

There are a variety of approaches to quantify ideological bias. It is common to use a liberal–conservative (i.e., “Left–Right”) political spectrum as a one-dimensional (1D) spectrum when analyzing ideological biases [138, 143, 144, 145]. This places these ideological biases in the context of common political polarities. A Left–Right dimension also arises in data-driven inference of ideological positions in multiple dimensions [132, 146]. In the United States, it is traditional to place the Democratic political party on the Left and the Republican political party on the Right. Ideological views man-

ifest in conversations and other “digital footprints” on social-media platforms [147], such as in posts by politicians on Twitter. For example, Anmol et al. [144] manually counted selected words in tweets that are related to COVID-19 and concluded that Republican politicians post more tweets that are related to business and the economy and that Democratic politicians concentrate more on public health. Xiao et al. [148] examined political polarities in textual data from social-media platforms (specifically, from Twitter and Parler) and quantified such ideological biases on tweets from politicians and media outlets by assigning polarity scores to words, hashtags, and other objects (“tokens”) in social-media posts. Waller et al. [140] introduced a multidimensional framework to summarize the ideological views, with a focus on traditional forms of identity (specifically, they considered age, gender, and political partisanship) of the posters and commenters, of Reddit posts around the time of the 2016 United States presidential election. Similarly, Gordon et al. [149] argued that one cannot fully capture political bias using a single axis with two binary ideological extremes (such as Republican and Democrat); instead, one should use multidimensional approaches. Moreover, models that aim to analyze the content of media outlets should incorporate not only measures of outlet biases but also measures of outlet quality [150, 151]. In this chapter, we use natural-language processing (NLP) [152] of textual data from tweets by media outlets to infer (Left–Right, low–high) coordinates for these outlets, where the first dimension describes the political bias of a media outlet and the second dimension represents its quality.

Neural-network models that are based on deep learning have been useful for many NLP tasks, including speech recognition [153], sentiment classification, answer selection, and textual entailment [154]. By using deep neural networks instead of traditional machine-learning (ML) approaches (such as a naive-Bayes method), one can significantly improve the performance of tasks like text classification [155]. Moreover, neural networks that exploit input word sequences can produce more accurate results than methods that rely on a bag-of-words approach [156, 157]. Recurrent neural networks (RNN) are a trendy deep-learning architecture to analyze sequential textual data. For



example, Socher et al. [158] used an RNN to study binary sentiments (i.e., favorable or unfavorable) from a data set of movie reviews. In this work, we apply a specific type of RNN called a long short-term memory (LSTM) neural network to infer ideological and quality coordinates of media outlets based on the textual content of their tweets. We also compare the results of using an LSTM network to those from several traditional ML methods that have been used in previous sentiment analysis studies. These traditional approaches include a naive-Bayes method, a support-vector machine (SVM), an artificial neural network (ANN), a decision tree, and a random forest. All of these traditional approaches use a bag-of-words approach.

To further motivate our work, consider the Ad Fontes Media-Bias Chart (AFMBC) [159], which is a two-dimensional (2D) visualization (see Figure 5.1) of the political ideologies and content qualities of about 100 media outlets. The AFMBC shows the positions of media outlets with ideological biases along one axis and content qualities along the other axis. The AFMBC uses data from 1,818 online articles and 98 cable news shows, which were rated in 2019 by a politically balanced team of analysts [159]. In the AFMBC, the bias and quality scores of each media outlet are the mean scores of each rated news item. Typically, 15–20 news items were used to evaluate a media outlet; at least three analysts rated each news item.

By necessity, the AFMBC uses a small number of items from each media outlet, although media outlets produce a wealth of content. By applying modern data-science techniques, one can leverage such abundant data through an algorithmic process of rating documents for both bias and quality. For example, Widmer et al. [160] applied penalized logistic regression and latent Dirichlet allocation (LDA) to a corpus of about 40,000 transcribed television episodes and examined the potential influence that national cable television can have on local newspapers. By measuring the textual similarities between the content of national television channels (specifically, Fox News, CNN, and MSNBC) and local newspaper content, they found that the content of local newspapers with more viewership of a given cable channel has greater textual similarity with the content of that cable channel than with it does with the other examined ca-

ble channels. They suggested the possibility that national cable television propagates slants and partisan biases to local newspapers and can thereby polarize local news content. In this chapter, we explore the potential to quantify the political ideologies and content qualities of media outlets based on the tweets that they posted in a specific time window. We use a data set from the Harvard GWU Libraries Dataverse [161] of more than 30 million tweets from more than 4,000 news outlets. We then remove tweets that were not posted by media outlets in the AFMBC. This leaves about 1.4 million tweets, to which we apply ML techniques to infer bias and quality coordinates for each tweet. There is no absolute truth in the numerical values of a media outlet’s bias and quality scores. Therefore, we evaluate the performance of the ML algorithms by comparing their outputs with the AFMBC. The 2D coordinates that we obtain from an algorithm can provide insight into ideological polarization and can serve as an input to opinion-dynamics models, such as the one in [162] that incorporates media outlets.

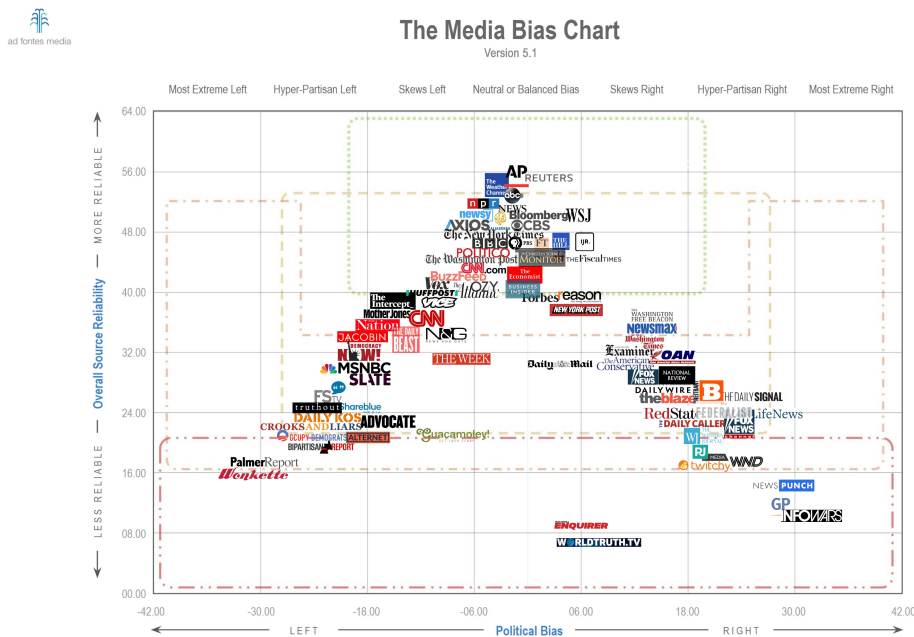


Figure 5.1: The Ad Fontes Media-Bias Chart (version 5.1). When we use data from this bias chart (see Section 5.5.1), we will normalize the bias scores to  $[-1, 1]$  and the quality scores to  $[0, 1]$ . [We reproduce this figure, with permission that is granted by our purchase of a Standard License, from Ad Fontes. See [https://adfontesmedia.com/copyright-and-usage-info/?utm\\_source=StaticMBCPage](https://adfontesmedia.com/copyright-and-usage-info/?utm_source=StaticMBCPage).]

### 5.1.1 Our Contributions

As of May 2020 (based on manual inspection), 65 of the media outlets in the AFMBC maintained active Twitter accounts. The other media outlets in the AFMBC either did not have a Twitter account or had a suspended account at that time. Between 4 August 2016 and 12 May 2020, these 65 accounts generated 1.4 million tweets; these tweets are tabulated in the George Washington University (GWU) Libraries Dataverse [161] in the Harvard Dataverse. We use several existing supervised ML algorithms (a naive-Bayes method, an SVM, an ANN, a decision tree, a random forest, and an LSTM network) to infer the ideological biases and content qualities of each tweet. We then compute the bias and quality scores for each media outlet by calculating the means of the biases and qualities of their tweets. In Figure 5.2, we show our workflow for inferring the bias and quality scores of the tweets and media outlets.

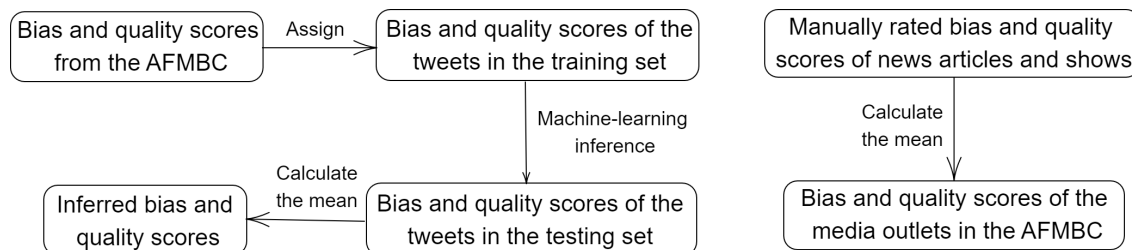


Figure 5.2: Flowcharts for inferring ideological biases and content qualities of the media outlets. We show (left) the workflow for our proposed approach and (right) the workflow for the AFMBC [159].

We observe a strong linear correlation between the manually rated political ideologies in the AFMBC (which plays the role of a “ground truth” in our study) and the political ideologies that are inferred by the LSTM network. We observe a similarly strong positive linear correlation between the AFMBC quality scores and the inferred quality scores. This suggests that algorithms can successfully produce reasonable (ideology, quality) scores for articles and other text from media outlets. We generate a 2D media-bias chart for the examined media outlets and compare it with the AFMBC. We then compare the results of an LSTM neural-network approach to the results of several

traditional methods (a naive-Bayes method, an SVM, a decision tree, a random forest, and an ANN). We find that the LSTM approach (which considers word sequences) outperforms these baseline methods, which each use a bag-of-words approach.

### 5.1.2 Organization of this Chapter

In Section 5.2, we discuss several ML methods that we use to infer the ideological biases and content qualities of tweets. In Section 5.3, we briefly describe the employed data, which includes the manually rated news articles and shows from Ad Fontes [159] and the media-outlet tweets from the GWU Libraries Dataverse [161]. In Section 5.4, we propose a scheme to construct a media-bias chart from the tweet content of media outlets. We also briefly describe our preprocessing of the tweets. In Section 5.5, we introduce how we evaluate the performance of each ML method and compare the performance of different ML methods. In Section 5.6, we conclude and discuss future work. In Appendix A, we list the 65 examined media outlets and the number of tweets for each of them.

## 5.2 Background and Related Work on NLP Methods

Because of the increased bounty and accessibility of machine-readable text [163], researchers have applied many supervised ML techniques to analyze and classify textual data [164]. One can categorize supervised NLP methods into (1) sequential approaches (which account for word order) and (2) non-sequential approaches (which do not). Non-sequential methods transform a document into a bag of words before subsequent analysis [165]. Such methods were the typical type of approach in early NLP studies with ML techniques [166, 167, 168], and they are still the main approach for smaller data sets (e.g., ones with fewer than 5,000 data points [169]). Importantly, sequential models (e.g., RNNs [170]) transform a document into a sequential input and use associated contextual information when mapping from an input sequence to an output sequence [171]. Therefore, we expect the inference of political ideologies to be more

effective with a sequential approach than with a bag-of-words approach [158].

In this section, we briefly discuss several non-sequential approaches (which we employ as baseline methods) and a sequential approach that we use to infer a media-bias chart from the tweets of media outlets. In Table 5.1, we summarize our key notation for this chapter.

Table 5.1: Key notation in Chapter 5.

symbol	explanation	example or further information
$\mathbf{X}$	an entire data set or a training set	example: the entire set of tweets
$\mathbf{x}$ or $\mathbf{x}_i$	an input feature vector	example: one tweet
$x_j$	the $j$ th entry of a vector $\mathbf{x}$	example: one word
$y_k$	the label of the $k$ th input	example: the label of the $k$ th tweet
$\mathbf{w}$	a vector of weights	we use these in the SVM and the neural networks
$b$	the bias term in $\mathbf{w} \cdot \mathbf{x} + b$	we use these in the SVM and the neural networks
$c$	the index of a label	it ranges from 1 to $n$ when there are $n$ classes in total

### 5.2.1 Naive-Bayes Method

A naive-Bayes method is a simple approach that has been used successfully for text categorization [168, 172, 173], which is the common NLP task of assigning each text document in a corpus to a category  $c \in \{1, \dots, n\}$ .

We start with Bayes' rule

$$\mathbb{P}(y_i = c \mid \mathbf{x}_i) = \frac{\mathbb{P}(y_i = c)\mathbb{P}(\mathbf{x}_i \mid y_i = c)}{\mathbb{P}(\mathbf{x}_i)}.$$

Using the chain rule, we compute the probability that the current item  $i$  (for example, the  $i$ th tweet), with feature vector  $\mathbf{x}_i = [x_{i_1}, \dots, x_{i_n}]^T$  (for example, the sequence of words in the  $i$ th tweet), is in category  $c$ . This yields

$$\begin{aligned} \mathbb{P}(y_i = c \mid x_{i_1}, \dots, x_{i_n}) &= \mathbb{P}(x_{i_1} \mid x_{i_2}, \dots, x_{i_n}, y_i = c) \times \dots \\ &\quad \times \mathbb{P}(x_{i_{n-1}} \mid x_{i_n}, y_i = c) \times \mathbb{P}(x_{i_n} \mid y_i = c) \times \mathbb{P}(y_i = c). \end{aligned}$$

The word “naive” appears in the method’s name because a naive-Bayes approach

assumes that all words have independent probabilities of appearing in a document. That is,

$$\mathbb{P}(x_{i_1} \mid x_{i_2}, \dots, x_{i_n}, y_i = c) = \mathbb{P}(x_{i_1} \mid y_i = c).$$

In a naive-Bayes approach, one needs both  $\mathbb{P}(y_i = c)$  and  $\mathbb{P}(x_j \mid y_i = c)$  to compute the output probability  $\mathbb{P}(y_i = c \mid x_{i_1}, \dots, x_{i_n})$ . One substitutes the probability  $\mathbb{P}(y_i = c)$  for the relative frequency of class  $c$  in the training set. We obtain the conditional probability  $\mathbb{P}(x_j \mid y_i = c)$  using maximum a posteriori (MAP) estimation [174].

### 5.2.2 Support-Vector Machines (SVMs)

SVMs have been employed often in classification tasks [175, 176, 177]. For example, Go et al. [172] used SVMs for sentiment classification of Twitter data. Gopi et al. [169] used an SVM approach for tweet classification using positive and negative opinions.

In a traditional SVM, one is given a training data set  $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  and seeks binary class labels, which we denote by  $+1$  and  $-1$ . One seeks the maximum-margin hyperplane that separates the data points of those two different classes. One attempts to minimize the hinge loss

$$\left[ \frac{1}{n} \sum_{i=1}^n \max\{0, 1 - y_i (\mathbf{w} \cdot \mathbf{x}_i + b)\} \right] + \lambda \|\mathbf{w}\|^2, \quad (5.1)$$

where the parameter  $\lambda > 0$  determines the trade-off between the margin size and the labeling accuracy, which is equal to the number of correctly assigned labels divided by the number of elements in the training set. The loss function (5.1) is convex, so one can use a common convex-optimization approach (e.g., gradient descent) to successfully minimize it [178].

The original SVM setting is directly applicable only to binary classification. For classification problems with three or more labels, one needs to split the classification task into multiple binary-classification tasks.

### 5.2.3 Decision Trees and Random Forests

A decision tree is a flowchart-like structure in which each node  $t$  of the tree splits the flow of a procedure into two sets of classes based on the information gain  $I(t) = H(\mathbf{X}) - H(\mathbf{X} | t)$ , where

$$H(\mathbf{X}) = - \sum_c \mathbb{P}(y_i = c) \log_2(\mathbb{P}(y_i = c))$$

is the entropy of a data set and

$$H(\mathbf{X} | t) = \sum_{d \in s(t)} p(d) H(\mathbf{X} | d)$$

is the conditional entropy of a data set after the split at node  $t$ , where  $s(t)$  denotes the set of all possible splits at  $t$  with respect to one variable.

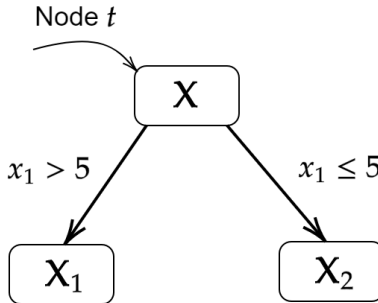


Figure 5.3: A schematic illustration of a simple decision tree. This example uses  $x_1 > 5$  as the criterion to split the source data  $\mathbf{X}$  into  $\mathbf{X}_1$  and  $\mathbf{X}_2$ .

In Figure 5.3, we give a schematic illustration of splitting a source data set. In this example, the full data set is  $\mathbf{X}$ , the subsets  $\mathbf{X}_1$  and  $\mathbf{X}_2$  are disjoint, and  $\mathbf{X} = \mathbf{X}_1 \cup \mathbf{X}_2$ . The entropy of the data set after the split at node  $t$  is

$$H(\mathbf{X} | t) = \frac{|\mathbf{X}_1|}{|\mathbf{X}|} H(\mathbf{X}_1) + \frac{|\mathbf{X}_2|}{|\mathbf{X}|} H(\mathbf{X}_2).$$

The root node is always split into two new nodes (which are called “children”), and we further split any new node that has data with multiple labels. We continue the

splitting process recursively (i.e., following a tree structure), using the criterion of maximum information gain at each node until we obtain a set of leaf nodes in which each node has data with a single label. See Breiman et al. [179] for other splitting strategies and stopping conditions for decision trees.

A random-forest classification approach uses a set of decision trees. It selects a random sample from a training set using some probability distribution and then fits trees to these samples [179]. A random-forest classifier consists of  $N$  trees, where one specifies the value  $N$ . We use  $N = 100$  to balance efficiency and performance and to avoid overfitting. To classify a new data set, we pass each sample of that data set to each of the  $N$  trees. The forest chooses a class with the most votes of the  $N$  possible votes; if there is a tie for the largest vote total, it uniformly randomly selects one of the classes with the most votes. In one example of sentiment analysis using a random-forest classifier, Bilal et al. [180] identified positive, negative, and neutral sentiments in a set of documents.

#### **5.2.4 Artificial Neural Networks (ANNs)**

Artificial neural networks (ANNs) are popular classifiers that have been used for various text-classification problems [181]. We use a fully-connected (i.e., “dense”) feedforward ANN to infer media bias and back-propagation (BP) to train this ANN. Because of the feedforward structure, the nodes are not part of any cycles. BP is an iterative gradient-based algorithm that we use to minimize the mean-squared error between the actual output and the desired output. See Schmidhuber [182] for more details about ANNs and training methods for them. ANNs have been used for a variety of tasks in sentiment analysis [183, 184, 185]. Zharmagambetov and Pak [181] used an ANN and a word-embedding model to classify a data set of movie reviews with positive and negative sentiments.



### 5.2.5 Long Short-Term Memory (LSTM) Neural Networks

An LSTM neural network is one type of RNN architecture. Unlike a traditional RNN, an LSTM network has a “forget” gate that determines whether or not to pass information from one memory cell to the next memory cell. LSTM networks have achieved good performance on various NLP tasks, including machine translation, next-word inference, and binary sentiment classification [186, 187, 188]. In Figure 5.4, we show the structure of a single LSTM cell. The scalar  $c$  denotes the “memory” that is passed between each cell. In the center panel (which indicates the  $t$ th cell), the sigmoid activation function  $\sigma$  on the left is a forget gate. When  $\sigma$  outputs 0, the previous memory  $c_{t-1}$  is “forgotten” by multiplying it by 0; when  $\sigma$  outputs 1, one uses  $c_{t-1}$  and passes it to the next cell. The middle sigmoid activation function  $\sigma$  is called the input gate, which decides the input value for updating the memory. The right sigmoid activation function  $\sigma$  is called the “output gate” and determines the output value. The hyperbolic tangent is another activation function, and the output is  $h_t$ . In Section 5.4.2, we will explain the architecture of the bidirectional LSTM network model that we use to infer the ideological biases and content qualities of the media outlets.

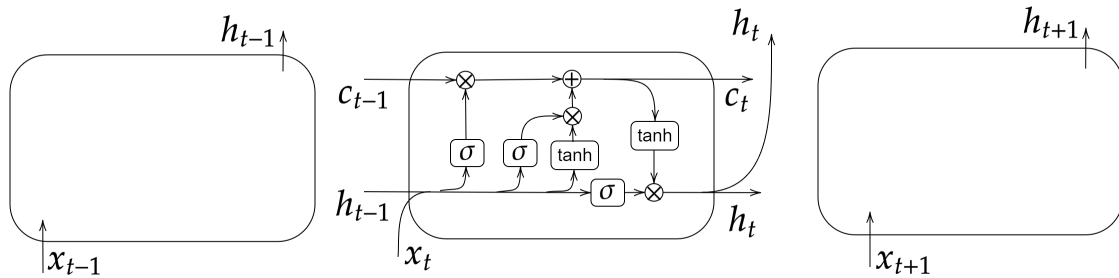


Figure 5.4: A schematic illustration of a memory cell in an LSTM neural network. The center panel is the  $t$ th cell, on which we focus. The quantity  $x_t$  is the input (e.g., a sequence of integers that represents all words in one document),  $c_t$  is the “memory-cell state” that is passed to the next cell, and the “hidden state”  $h_t$  is also passed to the next cell. The function  $\sigma$  is a sigmoid activation function, and the hyperbolic tangent  $\tanh$  is another activation function. In Section 5.4.2, we precisely specify the functions of the gates in each cell.

### 5.3 Data Sets

We use tweets from media outlets to examine the ideological biases and content qualities of those outlets. Tweets by media outlets are more readily available and larger in number than shows and articles from these outlets. We use data from the NEWSOUTLETTWEET data set of media tweets from Littman et al. [161]. This data set has a list of tweet IDs from the Twitter accounts of 9,636 news outlets. The tweets from these media outlets were collected between 4 August 2016 and 12 May 2020. The tweet IDs are 18-digit integers that provide access (with `twitter.com/` as the prefix) to the corresponding tweets. We use the Hydrator application [189] to extract the content of all 39,695,156 tweets in NEWSOUTLETTWEET; we refer to these tweets as the “hydrated NEWSOUTLETTWEET” data set. For each of the 102 media outlets in the AFMBC, we manually search for their official Twitter account and check if it is in the account list in NEWSOUTLETTWEET. If one media outlet has multiple official Twitter accounts — e.g., *The New York Times* has several Twitter accounts, such as `@NYTSports` and `@nytopinion` — we manually determine its primary Twitter account and keep only this account. (For example, we use `@nytimes` for *The New York Times*.) This yields 65 media outlets that are part of both the AFMBC and NEWSOUTLETTWEET. We list these media outlets and the associated Twitter accounts in Appendix A.

We select the tweets in the hydrated NEWSOUTLETTWEET data set that were posted by the 65 media outlets. There are 1,417,030 such tweets in total. Except for Bloomberg (for which there are 127 tweets), the numbers of tweets of the media outlets range from about 1,000 to about 100,000. We obtain bias and quality scores for each media outlet using the bias and quality scores of articles and shows in the MEDIASOURCERATINGS data set [159]. For each article, the ideological bias lies in one of seven categories: most extreme Left, hyperpartisan Left, skews Left, neutral, skews Right, hyperpartisan Right, and most extreme Right. Each category spans 12 rating units (except for the neutral category, which spans 13 units from  $-6$  to  $6$ ), so the numerical scale consists of all integers between  $-42$  and  $+42$ . The seven categories

of bias were defined by analysts, and the 12 units within each category allow nuanced distinctions in the amount of bias [159]. The overall reliability of each article by each media outlet was divided by the analysts into eight categories, with eight units each. This yields a numerical scale that consists of all integers between 1 (the least reliable) and 64 (the most reliable) [159]. The number of rating units was selected because it is convenient for the aspect ratio of visual displays.

Each media outlet in the AFMBC has between 15 and 25 reviewed articles and shows (in total, including both types of media) in the MEDIASOURCERATINGS data set. For each media outlet, we use the mean of the bias scores and the mean of the quality scores as representative quality and bias scores.

## 5.4 Generation of a Media-Bias Chart

We preprocess the tweet data from [161] and input it into an LSTM neural network to generate a bias score and a quality score for each tweet.<sup>1</sup> Using the (bias, quality) scores of the tweets (in the testing set), we generate a media-bias chart — with coordinates for both ideological bias and content quality — for the media outlets and then interpret it. The numbers of tweets of the media outlets range between 127 (for Bloomberg) and 93,259 (for Reuters); see Figure 5.5. For our data preprocessing, we select 10,000 tweets uniformly at random with replacement (so we do bootstrap sampling, and the sampled tweets can contain duplicates, especially for media outlets with smaller numbers of tweets) for each media outlet. With this procedure, our sample data set has  $65 \times 10,000$  tweets. The training set (a uniformly random subset) consists of a fixed proportion of the sampled tweets, to which we assign bias and quality scores that are equal to the bias and quality scores of the corresponding media outlets in the AFMBC. The media outlets thereby contribute equally to our training set regardless of how often they tweeted.

---

<sup>1</sup>Our code for filtering the tweets from NEWSOUTLETTWEET and applying ML algorithms to this data set is available at <https://gitlab.com/zchao3/MediaSentiment.git>.

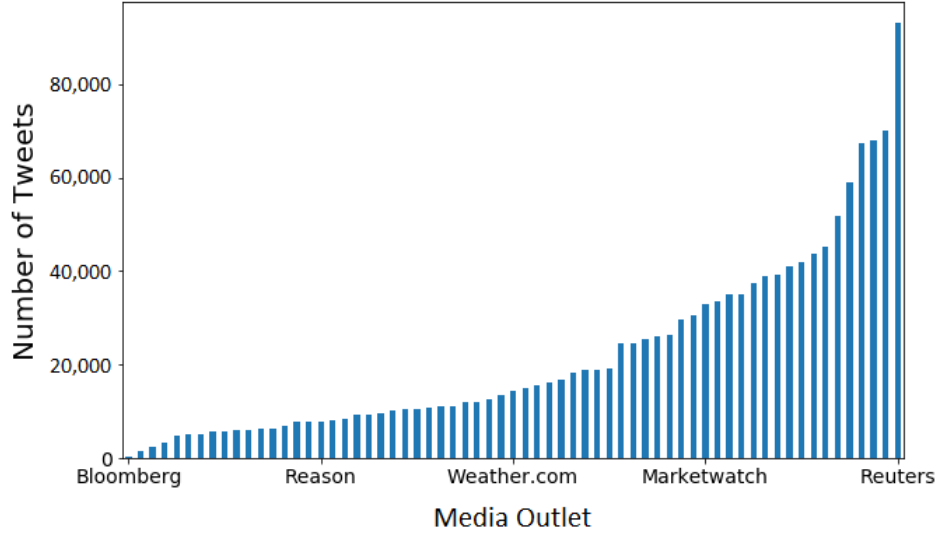


Figure 5.5: The number of tweets of the Twitter accounts of the 65 examined media outlets. Because of space considerations, we only show a few outlet names in this plot. For the complete list of media outlets, see Table A.1.

#### 5.4.1 Text Preprocessing

We apply the following text-preprocessing scheme to our data set. We remove all stop words (“and”, “you”, “to”, and so on) in the built-in list of stop words in the Python package NLTK, and we then remove all hyperlinks. We build a vocabulary using the 5,000 words (where we treat hashtags as words) with the highest frequencies in the sampled data set using the `COUNTVECTORIZER` function in the Python package SKLEARN. Each word that remains in a tweet is a token. We convert all words that are not in the vocabulary into “OOV” tokens and add “PAD” tokens at the end of each tweet so that all tweets have the same length (i.e., the same number of tokens) as the longest tweet.

#### 5.4.2 Bidirectional LSTM Neural Network

We input the processed tweets (which are sequences of tokens) into a bidirectional LSTM neural network to infer their bias and quality scores. The LSTM network that we use has three layers, which have different purposes. Each layer transforms an input

sequence into another sequence of scalars or into a sequence of vectors. The first layer is a fully-connected word-embedding layer [190] that transforms each word into a vector of a user-selected length. The second layer is a bidirectional LSTM layer that consists of a forward LSTM and a backward LSTM. The number of memory cells (see Section 5.2.5) in each LSTM is equal to the length of the input sequence. Each LSTM layer transforms an input sequence of vectors into a sequence of scalars. For each cell in the LSTM layer, we apply the following transition functions:

$$\begin{aligned}
 f_t &= \sigma(\mathbf{w}_f \cdot [h_{t-1}, \mathbf{x}_t] + b_f), \\
 i_t &= \sigma(\mathbf{w}_i \cdot [h_{t-1}, \mathbf{x}_t] + b_i), \\
 \tilde{c}_t &= \tanh(\mathbf{w}_c \cdot [h_{t-1}, \mathbf{x}_t] + b_c), \\
 c_t &= f_t c_{t-1} + i_t \tilde{c}_t, \\
 o_t &= \sigma(\mathbf{w}_o \cdot [h_{t-1}, \mathbf{x}_t] + b_o), \\
 h_t &= o_t \tanh(c_t).
 \end{aligned}$$

Following common practice, we refer to  $f_t$ ,  $i_t$ ,  $o_t$ ,  $c_t$ , and  $h_t$  as a forget gate, an input gate, an output gate, a memory-cell state, and a hidden state, respectively [191, 192, 193]. (The difference between a “state” and a “gate” is that the value of the former is passed to the next cell, but that is not the case for the latter.) The sigmoid function  $\sigma(x) = \frac{1}{1+e^{-x}}$  and tanh function are the activation functions. Finally, we append a fully-connected concatenation layer [194] that transforms the outputs from the bidirectional LSTM into a pair of scalars in  $[-1, 1] \times [0, 1]$  that encode a tweet’s bias score and quality score. Both the embedding layer and the concatenation layer use a matrix–vector product and a rectified linear unit (ReLU) as an activation function. We use the standard ReLU activation function [195]

$$\begin{aligned}
 f(x) &= \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases} \\
 &= \max\{0, x\} = x 1_{x>0}.
 \end{aligned}$$

In Figure 5.6, we show a schematic illustration of the bidirectional LSTM architecture and the workflow in our computational experiments. We initialize all parameters — including the parameters in the embedding layer, the concatenation layer, and the weight vectors  $\mathbf{w}_f$ ,  $\mathbf{w}_i$ ,  $\mathbf{w}_C$ , and  $\mathbf{w}_o$  — with independent uniform random real numbers in  $[0, 1]$ . As a loss function, we use the mean-square error (MSE)  $\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$ , where  $y_i$  denotes the  $i$ th training label and  $\hat{y}_i$  denotes the  $i$ th inferred label, and we train the weight vectors with the Adams optimizer [45] to minimize the loss function.

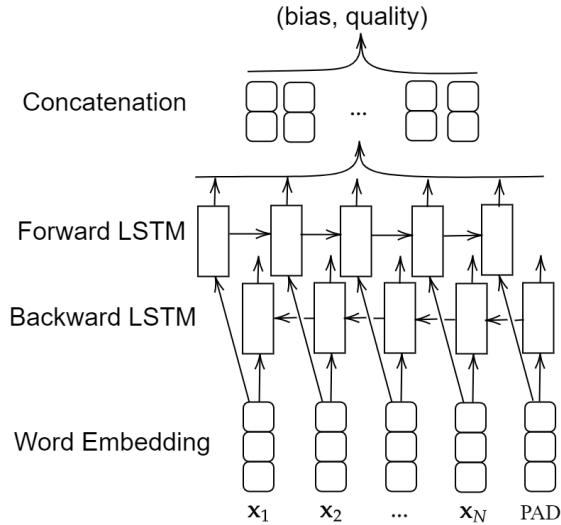


Figure 5.6: The structure of our bidirectional LSTM. The concatenation layer is a fully-connected layer that takes output sequences from both a forward LSTM and a backward LSTM as input. The output of the concatenation layer is a pair of scalars for the inferred bias and quality scores. The term “PAD” indicates the padding tokens that we append to a tweet so that all tweets have the same length (i.e., the same number of tokens).

### 5.4.3 Training and Results

We use a tweet-level split of testing and training data. We split the 650,000 tweets uniformly at random with 80% of them in the training set and 20% in the testing set. We apply the trained model to the testing tweets and assign bias scores and quality scores to each of these tweets. We group the testing set by media outlet, and we then

compute the mean bias score and mean quality score of each group and assign this pair of scores to the corresponding media outlet. We normalize the bias scores to  $[-1, 1]$ , and we normalize the quality scores to  $[0, 1]$ . In Figure 5.7, we plot these scores along with the normalized bias and quality scores from the AFMBC.

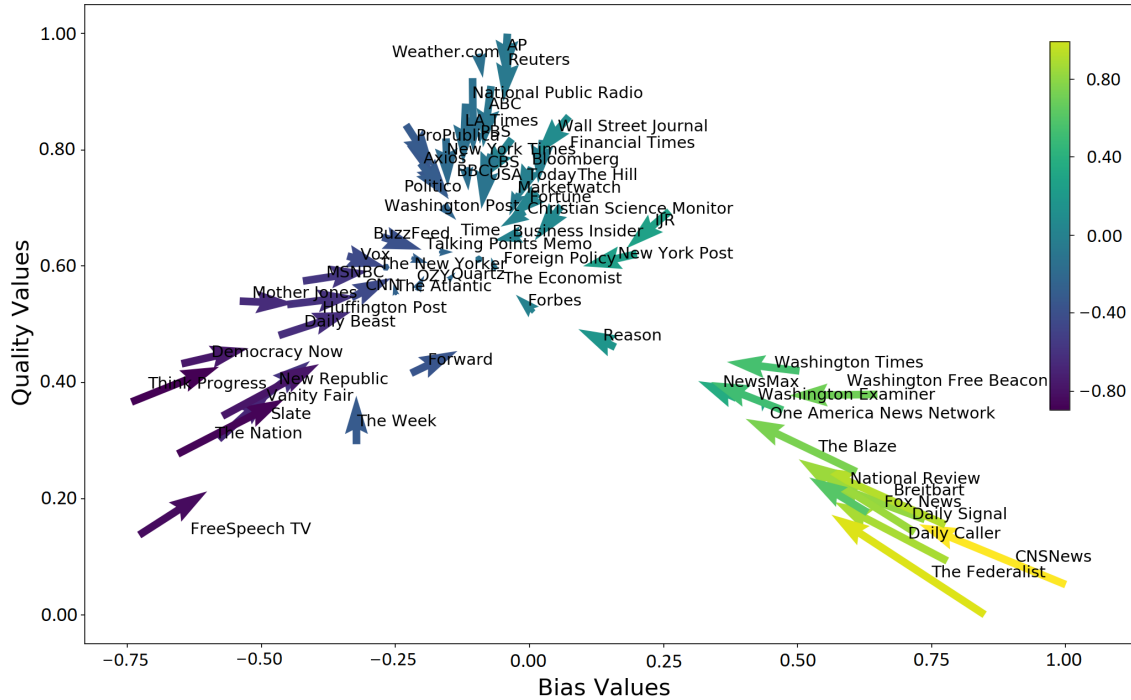


Figure 5.7: A comparison of the (bias, quality) scores from the AFMBC to the (bias, quality) scores that we obtain using an LSTM network. Each arrow starts from a media outlet’s coordinates in the AFMBC and terminates at the output coordinates from the LSTM network. The color of each arrow indicates the media outlet’s position on the Left–Right political spectrum (i.e., its ideological-bias score) in the AFMBC.

## 5.5 Evaluation of the LSTM Network’s Performance

We examine the performance of inferring a 2D media-bias chart of (bias, quality) coordinates for each media outlet using a variety of approaches (see Section 5.2). As we will see, the LSTM network (which incorporates sequential data) outperforms the other approaches, which use only non-sequential information.

In the output of each method, we observe that many tweets do not have a strong

ideological bias. This is the case even for many tweets from media outlets with a strong ideological bias (e.g., with an absolute value of at least 20 in the interval  $[-42, 42]$ ) in the AFMBC. Therefore, it is sensible that the mean bias scores of the tweets of the media outlets tend to be closer to the center of the ideological spectrum than their bias scores from the AFMBC (see Figure 5.7).

We calculate Pearson correlation coefficients, which are invariant under scaling and shifting of its arguments, to examine linear correlations between the inferred bias and quality scores and the AFMBC bias and quality scores. We use these coefficients to evaluate the performance of each method; a large Pearson correlation coefficient suggests that a method yields a spectrum with a reasonable ordering of the scores. For example, a large Pearson correlation coefficient for the bias score indicates a good performance at inferring the relative locations of media outlets on a Left–Right ideological spectrum. The Pearson correlation coefficient of two vectors  $\mathbf{y}$  and  $\hat{\mathbf{y}}$  is

$$\rho(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\mathbb{E}[(\mathbf{y} - \mu_{\mathbf{y}})(\hat{\mathbf{y}} - \mu_{\hat{\mathbf{y}}})]}{\sigma_{\mathbf{y}}\sigma_{\hat{\mathbf{y}}}},$$

where  $\mathbf{y}$  consists of the media-outlet bias or quality coordinates from the AFMBC and  $\hat{\mathbf{y}}$  consists of the algorithmically inferred media-outlet coordinates from tweets.

### 5.5.1 Computational Experiments

In addition to the LSTM-network approach on which we focus, we examine results from five other methods: a naive-Bayes (NB) method, a support-vector machine (SVM), a decision tree (DT), a random forest (RF), and an artificial neural network (ANN) with fully-connected layers.

We compare the (bias, quality) coordinates that we obtain for each media outlet from these methods. For each tweet, we remove the stop words and hyperlinks (see Section 5.4.1), and we use the 5,000 most-frequent words (see Section 5.4.2) to build a vocabulary. Our results vary slightly (with the Pearson correlation differing within about  $\pm 0.03$  in our experiments) when we use vocabularies with 2,000 and 10,000 words.



We apply an 80–20 train–test split of the data (see Section 5.4) and generate (bias, quality) coordinates for each media outlet by calculating the mean of each method’s output for the tweets from the same media outlet. We also test each method with train–test splits that we base on the media outlets themselves. In this media-outlet split, we select 52 media outlets (i.e., 80% of them) uniformly at random. We train each method using all of the tweets from these 52 media outlets and apply the trained method to the tweets of the remaining 13 media outlets to infer (bias, quality) coordinates of all remaining media outlets.

### 5.5.2 Results

We now compare and discuss our results from the various approaches for inferring (bias, quality) coordinates. We consider both a media-level split (see Section 5.5.1) and a tweet-level split (see Section 5.4.3).

For the media-outlet split, we uniformly randomly split the media outlets into five groups of 13 media outlets each, and we apply 5-fold cross-validation to generate the coordinates for all media outlets. That is, using each of the five groups as a withheld testing group, we train each method with the tweets from the other four groups and then evaluate the method with the tweets from the withheld group. We then compute the media bias and quality scores by calculating means of the scores of the tweets (see Section 5.4.3) and compute the Pearson correlation coefficients from these results. We show the mean Pearson correlation (which we average over five trials) for the media-outlet split in Table 5.2.

Method	Bias-Score Correlation	Quality-Score Correlation
NB	0.662	0.713
SVM	0.652	0.736
DT	0.665	0.684
RF	0.780	0.779
MLP	0.809	0.811
LSTM	<b>0.832</b>	<b>0.825</b>

Table 5.2: The Pearson correlations between the bias and quality scores in the AFMBC and the corresponding scores that we obtain from ML methods using a media-outlet split. For each method, the standard deviation of the correlations from the five different train–test splits in our 5-fold cross validation is smaller than 0.01. We show the means of the five different train–test splits for both the bias-score and quality-score correlations. We show the best results in bold.

We also use 5-fold cross validation for the tweet-level split. We uniformly randomly split all tweets into five equal-sized groups. Using each of the five groups as a withheld testing group, we train each method using the other four groups and then evaluate the method using the withheld group. We show the mean Pearson correlation (which we average over five different train–test splits) for the tweet-level split in Table 5.3.

Method	Bias-Score Correlation	Quality-Score Correlation
NB	0.861	0.805
SVM	0.909	0.898
DT	0.856	0.861
RF	0.925	0.907
MLP	0.916	0.949
LSTM	<b>0.977</b>	<b>0.964</b>

Table 5.3: The Pearson correlations between the bias and quality scores in the AFMBC and the corresponding scores that we obtain from ML methods using a tweet-level split. For each method, the standard deviation of the correlations from the five different train–test splits in our 5-fold cross validation is smaller than 0.01. We show the means of the five different train–test splits for both the bias-score and quality-score correlations. We show the best results in bold.

From Table 5.2 and Table 5.3, we see that the LSTM-network approach that incorporates word sequences in its input performs better than the other five methods, which all use a bag-of-words approach. This demonstrates that it is advantageous to

account for sequential information. We also observe that all methods perform better on the tweet-level split than they do on the media-outlet split. One possible reason is that different media outlets use different word choices for different tweets. Moreover, media outlets with similar ideological biases and quality scores on the AFMBC may tend to post about different topics. For example, `Weather.com` and `The Economist` have the same ideological-bias score (of  $-2.43$ ) on the AFMBC, but one expects posts by `Weather.com` to be related to weather forecasts, which one does not expect to see in many posts by `The Economist`.

## 5.6 Conclusions and Discussion

Media outlets have a significant and multifaceted impact on public discourse [196]. It is thus important to examine their ideological biases and heterogeneous quality levels. It can be very insightful to infer ideological positions and sentiments from the textual data of entities [138, 144], such as for media outlets. In this work, we used a bidirectional long short-term memory (LSTM) neural network and several other machine-learning approaches to infer ideological biases and quality levels of media outlets based on tweets from their Twitter accounts. For both biases and qualities, we found a large correlation between the scores that we inferred from tweets and the scores in the Ad Fontes Media-Bias Chart (AFMBC). We compared a variety of ML approaches that use a bag-of-words approach with the LSTM-network approach, which incorporates word sequences, and we found that the LSTM approach outperforms the others. We thus conclude that the information from word order is a significant contributor to our natural-language-processing (NLP) task. We expect that this is also true for many other NLP tasks.

In this chapter, we demonstrated that ML methods can successfully infer ideological biases and quality levels of textual data from media outlets. However, our study has several limitations. For example, we used only Twitter data and we only considered that data from a particular time window. It is essential to integrate different types

of news sources, such as long articles and videos, to better infer the ideological biases and content quality of a media outlet. Additionally, we only inferred a single point in a 2D (bias, quality) space for each media outlet, but it is more realistic to represent the bias and quality of a media outlet using a probability distribution. It is also desirable to extend the analysis of ideological bias to multiple dimensions (e.g., with different dimensions for different political issues, such as social and economic issues) and to develop better approaches to evaluate inferred (bias, quality) coordinates and associated media-bias charts. Our work provides a proof of concept for such studies, and it is important to explore these extensions.

## APPENDIX A

### List of Media Outlets and their Number of Tweets

In Table A.1, we list the 65 media outlets that we use in our experiments. We also indicate the number of tweets from each media outlet. The tweets were collected by Littman et al. [161] between 4 August 2016 and 12 May 2020. In Section 5.3, we described our process of manually selecting a Twitter account for each media outlet. In Table A.1, we also give each media outlet’s bias and quality scores from Ad Fontes [159].

Outlet	Ideological Bias	Content Quality	Number of Tweets
ABC	-1.85	49.87	39243
AP	-1.06	52.19	35132
Axios	-5.74	47.30	10647
BBC	-3.03	46.27	5685
Bloomberg	-0.85	47.52	148
Breitbart	18.99	30.64	9335
Business Insider	-0.38	43.28	58868
BuzzFeed	-7.06	43.17	16214
CBS	-1.85	46.84	39004
CNN	-8.55	40.49	68014
CNSNews	25.75	27.75	5570
Christian Science Monitor	-0.21	44.27	24660
Daily Beast	-12.04	38.80	18088
Daily Caller	20.06	28.80	25948

Daily Signal	19.97	30.41	7812
Democracy Now	-16.71	37.54	8132
Financial Times	0.62	47.47	14966
Fiscal Times	1.52	44.54	3250
Forbes	0.20	39.84	19143
Foreign Policy	-1.65	41.69	10496
Fortune	0.43	45.09	18971
Forward	-5.69	37.12	8247
Fox News	18.50	30.08	67254
FreeSpeech TV	-18.74	29.95	10970
Huffington Post	-11.64	40.17	18679
IJR	6.72	44.31	4867
LA Times	-3.06	49.09	29475
MSNBC	-10.88	41.21	25312
Marketwatch	-0.54	45.11	32766
Mother Jones	-13.92	40.32	11903
National Public Radio	-2.73	50.22	16591
National Review	16.23	30.95	11012
New Republic	-12.83	36.03	6097
New York Post	5.15	42.42	30435
New York Times	-4.01	47.54	43772
NewsMax	9.94	36.02	5016
OZY	-5.43	40.80	5991
One America News Network	11.26	35.88	6128
PBS	-2.37	47.79	5046
Politico	-5.24	46.45	51742
ProPublica	-5.93	48.14	1430
Quartz	-3.89	41.26	12646
Reason	4.12	38.28	7849

Reuters	-0.95	51.79	93120
Slate	-14.93	34.20	37344
Talking Points Memo	-5.67	42.24	6739
The Atlantic	-6.41	40.59	10146
The Blaze	15.70	32.76	9091
The Economist	-2.43	42.19	34927
The Federalist	21.86	26.42	7767
The Hill	0.09	46.26	70065
The Nation	-16.89	33.54	2462
The New Yorker	-6.90	41.83	13520
The Week	-8.31	33.98	9579
Think Progress	-19.12	35.85	10546
Time	-4.35	42.50	40979
USA Today	-2.03	46.12	24613
Vanity Fair	-14.75	35.22	6064
Vox	-8.75	42.33	15497
Wall Street Journal	1.89	48.52	41903
Washington Examiner	12.17	35.48	33592
Washington Free Beacon	16.71	36.19	12062
Washington Post	-4.18	44.57	45178
Washington Times	12.97	37.23	26358
Weather.com	-2.43	51.30	14345

Table A.1: The published bias and quality scores of the media outlets in the Ad Fontes Media-Bias Chart (version 5.1) and the number of tweets of each media outlet.

## REFERENCES

- [1] J. Liu, P. Musialski, P. Wonka, and J. Ye, “Tensor completion for estimating missing values in visual data,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 208–220, 2012.
- [2] X. Li, D. Xu, H. Zhou, and L. Li, “Tucker tensor regression and neuroimaging analysis,” *Statistics in Biosciences*, vol. 10, no. 3, pp. 520–545, 2018.
- [3] T. Liu, M. Yuan, and H. Zhao, “Characterizing spatiotemporal transcriptome of the human brain via low-rank tensor decomposition,” *Statistics in Biosciences*, vol. 14, no. 3, pp. 485–513, 2022.
- [4] Y. Luo, Y. Xin, E. Hochberg, R. Joshi, O. Uzuner, and P. Szolovits, “Subgraph augmented non-negative tensor factorization (santf) for modeling clinical narrative text,” *Journal of the American Medical Informatics Association*, vol. 22, no. 5, pp. 1009–1019, 2015.
- [5] J. Chambua, Z. Niu, A. Yousif, and J. Mbelwa, “Tensor factorization method based on review text semantic similarity for rating prediction,” *Expert Systems with Applications*, vol. 114, pp. 629–638, 2018.
- [6] X. Zheng, W. Ding, Z. Lin, and C. Chen, “Topic tensor factorization for recommender system,” *Information Sciences*, vol. 372, no. 5, pp. 276–293, 2016.
- [7] T. Song, Z. Peng, S. Wang, W. Fu, X. Hong, and P. S. Yu, “Based cross-domain recommendation through joint tensor factorization,” in *International conference on database systems for advanced applications*, pp. 525–540, Springer, 2017.
- [8] J.-Y. Jiang, Z. Chao, A. L. Bertozzi, W. Wang, S. D. Young, and D. Needell, “Learning to predict human stress level with incomplete sensor data from wearable devices,” in *Proceedings of the 28th ACM International conference on information and knowledge management*, pp. 2773–2781, 2019.
- [9] H. Abdi and L. J. Williams, “Principal component analysis,” *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.
- [10] D. D. Lee and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.
- [11] J.-F. Cai, E. J. Candès, and Z. Shen, “A singular value thresholding algorithm for matrix completion,” *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.
- [12] H. Cai, J.-F. Cai, and K. Wei, “Accelerated alternating projections for robust principal component analysis,” *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 685–717, 2019.



- [13] E. J. Candès and B. Recht, “Exact matrix completion via convex optimization,” *Foundations of Computational mathematics*, vol. 9, no. 6, pp. 717–772, 2009.
- [14] E. J. Candès and Y. Plan, “Matrix completion with noise,” *Proceedings of the IEEE*, vol. 98, no. 6, pp. 925–936, 2010.
- [15] D. Molitor and D. Needell, “Matrix completion for structured observations,” *arXiv:1801.09657*, 2018.
- [16] J. L. Schafer and J. W. Graham, “Missing data: our view of the state of the art.,” *Psychological methods*, vol. 7, no. 2, pp. 147–177, 2002.
- [17] J. Chen and J. Yang, “Low-rank matrix completion based on maximum likelihood estimation,” in *2013 2nd IAPR Asian Conference on Pattern Recognition*, pp. 261–265, IEEE, 2013.
- [18] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 8, pp. 30–37, 2009.
- [19] P. Paatero and U. Tapper, “Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values,” *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.
- [20] B. Recht, M. Fazel, and P. A. Parrilo, “Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization,” *SIAM review*, vol. 52, no. 3, pp. 471–501, 2010.
- [21] X. Han, J. Wu, L. Wang, Y. Chen, L. Senhadji, and H. Shu, “Linear total variation approximate regularized nuclear norm optimization for matrix completion,” in *Abstract and Applied Analysis*, vol. 2014, Hindawi, 2014.
- [22] J. Bakker, M. Pechenizkiy, and N. Sidorova, “What’s your current stress level? detection of stress patterns from gsr sensor data,” in *2011 IEEE 11th international conference on data mining workshops*, pp. 573–580, IEEE, 2011.
- [23] O. M. Mozos, V. Sandulescu, S. Andrews, D. Ellis, N. Bellotto, R. Dobrescu, and J. M. Ferrandez, “Stress detection using wearable physiological and sociometric sensors,” *International journal of neural systems*, vol. 27, no. 02, p. 1650041, 2017.
- [24] L. Canzian and M. Musolesi, “Trajectories of depression: unobtrusive monitoring of depressive states by means of smartphone mobility traces analysis,” in *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*, pp. 1293–1304, ACM, 2015.
- [25] N. Jaques, S. Taylor, A. Sano, R. Picard, *et al.*, “Predicting tomorrow’s mood, health, and stress level using personalized multitask learning and domain adaptation,” in *IJCAI 2017 Workshop on Artificial Intelligence in Affective Computing*, pp. 17–33, 2017.

- [26] A. Bogomolov, B. Lepri, M. Ferron, F. Pianesi, and A. S. Pentland, “Daily stress recognition from mobile phone data, weather conditions and individual traits,” in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 477–486, ACM, 2014.
- [27] R. LiKamWa, Y. Liu, N. D. Lane, and L. Zhong, “Moodscope: Building a mood sensor from smartphone usage patterns,” in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pp. 389–402, ACM, 2013.
- [28] R. Garrett, S. Liu, and S. D. Young, “A longitudinal analysis of stress among incoming college freshmen,” *Journal of American college health*, vol. 65, no. 5, pp. 331–338, 2017.
- [29] B. Cheng, “Emotion recognition from physiological signals using adaboost,” in *International Conference on Applied Informatics and Communication*, pp. 412–417, Springer, 2011.
- [30] G. Gosztolya, R. Busa-Fekete, and L. Toth, “Detecting autism, emotions and social signals using adaboost,” in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pp. 220–224, 2013.
- [31] M. A. Chikh, M. Saidi, and N. Settouti, “Diagnosis of diabetes diseases using an artificial immune recognition system2 (airs2) with fuzzy k-nearest neighbor,” *Journal of medical systems*, vol. 36, no. 5, pp. 2721–2729, 2012.
- [32] M. Shouman, T. Turner, and R. Stocker, “Applying k-nearest neighbour in diagnosing heart disease patients,” *International Journal of Information and Education Technology*, vol. 2, no. 3, pp. 220–223, 2012.
- [33] J. Yang and X. Yuan, “Linearized augmented lagrangian and alternating direction methods for nuclear norm minimization,” *Mathematics of computation*, vol. 82, no. 281, pp. 301–329, 2013.
- [34] L. Vandenberghe and S. Boyd, “Semidefinite programming,” *SIAM review*, vol. 38, no. 1, pp. 49–95, 1996.
- [35] E. J. Candès and T. Tao, “The power of convex relaxation: Near-optimal matrix completion,” *arXiv:0903.1476*, 2009.
- [36] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, “Hierarchical attention networks for document classification,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489, 2016.
- [37] J.-Y. Jiang, M. Zhang, C. Li, M. Bendersky, N. Golbandi, and M. Najork, “Semantic text matching for long-form documents,” in *The World Wide Web Conference*, pp. 795–806, ACM, 2019.

- [38] T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, 2015.
- [39] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [40] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv:1406.1078*, 2014.
- [41] R. Jozefowicz, W. Zaremba, and I. Sutskever, “An empirical exploration of recurrent network architectures,” in *Proceedings of the 32nd International Conference on Machine Learning, ICML ’15*, pp. 2342–2350, 2015.
- [42] M. W. Gardner and S. Dorling, “Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences,” *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.
- [43] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
- [44] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, “Tensorflow: A system for large-scale machine learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016.
- [45] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *ArXiv:1412.6980*, 2014.
- [46] A. C. Kokaram, R. D. Morris, W. J. Fitzgerald, and P. J. Rayner, “Interpolation of missing data in image sequences,” *IEEE Transactions on Image Processing*, vol. 4, no. 11, pp. 1509–1519, 1995.
- [47] A. M. Buchanan and A. W. Fitzgibbon, “Damped newton algorithms for matrix factorization with missing data,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 2, pp. 316–322, IEEE, 2005.
- [48] F.-T. Sun, C. Kuo, H.-T. Cheng, S. Buthpitiya, P. Collins, and M. Griss, “Activity-aware mental stress detection using physiological sensors,” in *International conference on Mobile computing, applications, and services*, pp. 282–301, Springer, 2010.
- [49] J. H. Hansen and B. D. Womack, “Feature analysis and neural network-based classification of speech under stress,” *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 4, pp. 307–313, 1996.
- [50] C. J. Appellof and E. R. Davidson, “Strategies for analyzing data from video fluorometric monitoring of liquid chromatographic effluents,” *Analytical Chemistry*, vol. 53, no. 13, pp. 2053–2056, 1981.

- [51] P. Comon, “Tensor decompositions, state of the art and applications.,” *arXiv:0905.0454*.
- [52] D. Kressner, M. Steinlechner, and B. Vandereycken, “Low-rank tensor completion by riemannian optimization,” *BIT Numerical Mathematics*, vol. 54, no. 2, pp. 447–468, 2014.
- [53] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, “Tag recommendations based on tensor dimensionality reduction,” in *Proceedings of the 2008 ACM conference on Recommender systems*, pp. 43–50, ACM, 2008.
- [54] J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert, “Low-rank matrix factorization with attributes,” *arXiv:cs/0611124*, 2006.
- [55] Y. Amit, M. Fink, N. Srebro, and S. Ullman, “Uncovering shared structures in multiclass classification,” in *Proceedings of the 24th international conference on Machine learning*, pp. 17–24, ACM, 2007.
- [56] J.-F. Cai, E. J. Candès, and Z. Shen, “A singular value thresholding algorithm for matrix completion,” *SIAM Journal on optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.
- [57] T. T. Cai, W.-X. Zhou, *et al.*, “Matrix completion via max-norm constrained optimization,” *Electronic Journal of Statistics*, vol. 10, no. 1, pp. 1493–1525, 2016.
- [58] E. J. Candès and Y. Plan, “Matrix completion with noise,” *Proceedings of the IEEE*, vol. 98, no. 6, pp. 925–936, 2010.
- [59] E. J. Candès and B. Recht, “Exact matrix completion via convex optimization,” *Foundations of Computational mathematics*, vol. 9, no. 6, p. 717, 2009.
- [60] P. Chen and D. Suter, “Recovering the missing components in a large noisy low-rank matrix: Application to sfm,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 8, pp. 1051–1063, 2004.
- [61] S. Foucart, D. Needell, R. Pathak, Y. Plan, and M. Wootters, “Weighted matrix completion from non-random, non-uniform sampling patterns,” *arXiv:1910.13986*, 2019.
- [62] D. F. Gleich and L.-h. Lim, “Rank aggregation via nuclear norm minimization,” in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 60–68, ACM, 2011.
- [63] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, “Using collaborative filtering to weave an information tapestry,” *Communications of the ACM*, vol. 35, no. 12, pp. 61–71, 1992.

- [64] A. Chambolle, V. Caselles, D. Cremers, M. Novaga, and T. Pock, “An introduction to total variation for image analysis,” *Theoretical foundations and numerical methods for sparse recovery*, vol. 9, pp. 263–340, 2010.
- [65] T. Zhang, J. Wang, L. Xu, and P. Liu, “Using wearable sensor and nmf algorithm to realize ambulatory fall detection,” in *International conference on natural computation*, pp. 488–491, Springer, 2006.
- [66] Z. Liu and L. Vandenberghe, “Interior-point method for nuclear norm approximation with application to system identification,” *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 3, pp. 1235–1256, 2009.
- [67] T.-Y. Ji, T.-Z. Huang, X.-L. Zhao, T.-H. Ma, and G. Liu, “Tensor completion using total variation and low-rank matrix factorization,” *Information Sciences*, vol. 326, pp. 243–257, 2016.
- [68] T. G. Kolda and B. W. Bader, “Tensor decompositions and applications,” *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.
- [69] F. L. Hitchcock, “The expression of a tensor or a polyadic as a sum of products,” *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.
- [70] F. L. Hitchcock, “Multiple invariants and generalized rank of a p-way matrix or tensor,” *Journal of Mathematics and Physics*, vol. 7, no. 1-4, pp. 39–79, 1928.
- [71] C. J. Hillar and L.-H. Lim, “Most tensor problems are np-hard,” *Journal of the ACM (JACM)*, vol. 60, no. 6, pp. 1–39, 2013.
- [72] J. D. Carroll and J.-J. Chang, “Analysis of individual differences in multidimensional scaling via an n-way generalization of “eckart-young” decomposition,” *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.
- [73] L. De Lathauwer, B. De Moor, and J. Vandewalle, “A multilinear singular value decomposition,” *SIAM journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.
- [74] Y. Xu, R. Hao, W. Yin, and Z. Su, “Parallel matrix factorization for low-rank tensor completion,” *Inverse Problems and Imaging*, vol. 9, no. 2, pp. 601–624, 2015.
- [75] L. Li, F. Jiang, and R. Shen, “Total variation regularized reweighted low-rank tensor completion for color image inpainting,” in *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 2152–2156, IEEE, 2018.
- [76] P. Getreuer, “Rudin-osher-fatemi total variation denoising using split bregman,” *Image Processing On Line*, vol. 2, pp. 74–95, 2012.
- [77] Z. Fang, X. Yang, L. Han, and X. Liu, “A sequentially truncated higher order singular value decomposition-based algorithm for tensor completion,” *IEEE transactions on cybernetics*, vol. 49, no. 5, pp. 1956–1967, 2018.

- [78] N. Vannieuwenhoven, R. Vandebril, and K. Meerbergen, “A new truncation strategy for the higher-order singular value decomposition,” *SIAM Journal on Scientific Computing*, vol. 34, no. 2, pp. A1027–A1052, 2012.
- [79] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan, “Tensor robust principal component analysis: Exact recovery of corrupted low-rank tensors via convex optimization,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5249–5257, 2016.
- [80] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma, “RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2233–2246, 2012.
- [81] M. Ding, T.-Z. Huang, T.-Y. Ji, X.-L. Zhao, and J.-H. Yang, “Low-rank tensor completion using matrix factorization based on tensor train rank and total variation,” *Journal of Scientific Computing*, vol. 81, no. 2, pp. 941–964, 2019.
- [82] E. Candes and J. Romberg, “Sparsity and incoherence in compressive sampling,” *Inverse Problems*, vol. 23, no. 3, p. 969, 2007.
- [83] E. J. Candès, X. Li, Y. Ma, and J. Wright, “Robust principal component analysis?,” *Journal of the ACM (JACM)*, vol. 58, no. 3, pp. 1–37, 2011.
- [84] P. Netrapalli, N. U N, S. Sanghavi, A. Anandkumar, and P. Jain, “Non-convex robust PCA,” in *Advances in Neural Information Processing Systems*, vol. 27, 2014.
- [85] T. Bouwmans, S. Javed, H. Zhang, Z. Lin, and R. Otazo, “On the applications of robust PCA in image and video processing,” *Proceedings of the IEEE*, vol. 106, no. 8, pp. 1427–1457, 2018.
- [86] H. Cai, J.-F. Cai, T. Wang, and G. Yin, “Accelerated structured alternating projections for robust spectrally sparse signal recovery,” *IEEE Transactions on Signal Processing*, vol. 69, pp. 809–821, 2021.
- [87] H. Cai, K. Hamm, L. Huang, J. Li, and T. Wang, “Rapid robust principal component analysis: CUR accelerated inexact low rank estimation,” *IEEE Signal Processing Letters*, vol. 28, pp. 116–120, 2020.
- [88] H. Cai, K. Hamm, L. Huang, and D. Needell, “Robust CUR decomposition: Theory and imaging applications,” *SIAM Journal on Imaging Sciences*, vol. 14, no. 4, pp. 1472–1503, 2021.
- [89] H. Cai, J. Liu, and W. Yin, “Learned robust pca: A scalable deep unfolding approach for high-dimensional outlier detection,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.

- [90] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, “Robust face recognition via sparse representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2008.
- [91] Y. Hu, J.-X. Liu, Y.-L. Gao, and J. Shang, “DSTPCA: Double-sparse constrained tensor principal component analysis method for feature selection,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pp. 1481–1491, 2019.
- [92] L. Li, W. Huang, I. Y.-H. Gu, and Q. Tian, “Statistical modeling of complex backgrounds for foreground object detection,” *IEEE Transactions on Image Processing*, vol. 13, no. 11, pp. 1459–1472, 2004.
- [93] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan, “Tensor robust principal component analysis with a new tensor nuclear norm,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 925–938, 2019.
- [94] J. Chen and Y. Saad, “On the tensor svd and the optimal low rank orthogonal approximation of tensors,” *SIAM journal on Matrix Analysis and Applications*, vol. 30, no. 4, pp. 1709–1734, 2009.
- [95] L. R. Tucker, “Some mathematical notes on three-mode factor analysis,” *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.
- [96] J. D. Carroll and J.-J. Chang, “Analysis of individual differences in multidimensional scaling via an n-way generalization of “eckart-young” decomposition,” *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.
- [97] Z. Zhang, G. Ely, S. Aeron, N. Hao, and M. Kilmer, “Novel methods for multilinear data completion and de-noising based on tensor-SVD,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3842–3849, 2014.
- [98] H. Cai, K. Hamm, L. Huang, and D. Needell, “Mode-wise tensor decompositions: Multi-dimensional generalizations of CUR decompositions,” *The Journal of Machine Learning Research*, vol. 22, no. 185, pp. 1–36, 2021.
- [99] N. Vaswani and P. Narayanamurthy, “Static and dynamic robust pca and matrix completion: A review,” *Proceedings of the IEEE*, vol. 106, no. 8, pp. 1359–1379, 2018.
- [100] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, “A survey of multilinear subspace learning for tensor data,” *Pattern Recognition*, vol. 44, no. 7, pp. 1540–1551, 2011.
- [101] J.-F. Cai, J. Li, and D. Xia, “Generalized low-rank plus sparse tensor estimation by fast riemannian optimization,” *Journal of the American Statistical Association*, pp. 1–17, 2022.

- [102] Y. Liu, L. Chen, and C. Zhu, “Improved robust tensor principal component analysis via low-rank core matrix,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 6, pp. 1378–1389, 2018.
- [103] S. E. Sofuoglu and S. Aiyente, “A two-stage approach to robust tensor decomposition,” in *2018 IEEE Statistical Signal Processing Workshop (SSP)*, pp. 831–835, IEEE, 2018.
- [104] Y. Hu and D. B. Work, “Robust tensor recovery with fiber outliers for traffic events,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 1, pp. 1–27, 2020.
- [105] B. Huang, C. Mu, D. Goldfarb, and J. Wright, “Provable low-rank tensor recovery,” *Optimization-Online*, vol. 4252, no. 2, pp. 455–500, 2014.
- [106] Q. Gu, H. Gui, and J. Han, “Robust tensor decomposition with gross corruption,” *Advances in Neural Information Processing Systems*, vol. 27, pp. 1422–1430, 2014.
- [107] S. A. Goreňov, N. L. Zamarashkin, and E. E. Tyrtysnikov, “Pseudo-skeleton approximations,” *Doklady Akademii Nauk*, vol. 343, no. 2, pp. 151–152, 1995.
- [108] K. Hamm and L. Huang, “Perspectives on CUR decompositions,” *Applied and Computational Harmonic Analysis*, vol. 48, no. 3, pp. 1088–1099, 2020.
- [109] M. W. Mahoney, M. Maggioni, and P. Drineas, “Tensor-CUR decompositions for tensor-based data,” *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 957–987, 2008.
- [110] C. F. Caiafa and A. Cichocki, “Generalizing the column–row matrix decomposition to multi-way arrays,” *Linear Algebra and its Applications*, vol. 433, no. 3, pp. 557–573, 2010.
- [111] K. Hamm and L. Huang, “Stability of sampling for CUR decompositions,” *Foundations of Data Science*, vol. 2, no. 2, p. 83, 2020.
- [112] G. Bergqvist and E. G. Larsson, “The higher-order singular value decomposition: Theory and an application [lecture notes],” *IEEE signal processing magazine*, vol. 27, no. 3, pp. 151–154, 2010.
- [113] L. De Lathauwer, B. De Moor, and J. Vandewalle, “On the best rank-1 and rank- $(r_1, \dots, r_n)$  approximation of higher-order tensors,” *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1324–1342, 2000.
- [114] A. Rajwade, A. Rangarajan, and A. Banerjee, “Image denoising using the higher order singular value decomposition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 4, pp. 849–862, 2012.



- [115] H. Cai, Z. Chao, L. Huang, and D. Needell, “Fast robust tensor principal component analysis via fiber cur decomposition,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 189–197, 2021.
- [116] E. López-Rubio, R. M. Luque-Baena, and E. Domínguez, “Foreground detection in video sequences with probabilistic self-organizing maps,” *International Journal of Neural Systems*, vol. 21, no. 03, pp. 225–246, 2011.
- [117] T. Bouwmans, L. Maddalena, and A. Petrosino, “Scene background initialization: A taxonomy,” *Pattern Recognition Letters*, vol. 96, pp. 3–11, 2017.
- [118] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis, *et al.*, “A large-scale benchmark dataset for event recognition in surveillance video,” in *CVPR 2011*, pp. 3153–3160, IEEE, 2011.
- [119] Z. Chao, L. Huang, and D. Needell, “Tensor completion through total variation with initialization from weighted hosvd,” in *2020 Information Theory and Applications Workshop (ITA)*, pp. 1–8, IEEE, 2020.
- [120] A. J. O’Toole, J. Harms, S. L. Snow, D. R. Hurst, M. R. Pappas, J. H. Ayyad, and H. Abdi, “A video database of moving faces and people,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 812–816, 2005.
- [121] P. Ji and J. Jin, “Coauthorship and citation networks for statisticians,” *The Annals of Applied Statistics*, vol. 10, no. 4, pp. 1779–1812, 2016.
- [122] J.-F. Cai, J. Li, and D. Xia, “Generalized low-rank plus sparse tensor estimation by fast riemannian optimization,” *arXiv:2103.08895*, 2021.
- [123] A. S. Gerber, D. Karlan, and D. Bergan, “Does the media matter? A field experiment measuring the effect of newspapers on voting behavior and political opinions,” *American Economic Journal: Applied Economics*, vol. 1, no. 2, pp. 35–52, 2009.
- [124] J. Amedie, “The impact of social media on society,” *Pop Culture Intersections*, vol. 2, 2015. Available at [https://scholarcommons.scu.edu/eng1\\_176/2](https://scholarcommons.scu.edu/eng1_176/2).
- [125] T. South, B. Smart, M. Roughan, and L. Mitchell, “Information flow estimation: A study of news on Twitter,” *Online Social Networks and Media*, vol. 31, p. 100231, 2022.
- [126] J. H. Tien, M. C. Eisenberg, S. T. Cherng, and M. A. Porter, “Online reactions to the 2017 ‘Unite the Right’ rally in charlottesville: Measuring polarization in Twitter networks using media followership,” *Applied Network Science*, vol. 5, no. 1, pp. 1–27, 2020.
- [127] B. J. Rye and A. Underhill, “Pro-choice and pro-life are not enough: An investigation of abortion attitudes as a function of abortion prototypes,” *Sexuality & Culture*, vol. 24, pp. 1829–1851, 2020.

- [128] T. Cicchini, S. M. Del Pozo, E. Tagliazucchi, and P. Balenzuela, “News sharing on Twitter reveals emergent fragmentation of media agenda and persistent polarization,” *European Physical Journal — Data Science*, vol. 11, no. 1, p. 48, 2022.
- [129] M. F. Schober, J. Pasek, L. Guggenheim, C. Lampe, and F. G. Conrad, “Social media analyses for social measurement,” *Public Opinion Quarterly*, vol. 80, no. 1, pp. 180–211, 2016.
- [130] C. A. Bail, L. P. Argyle, T. W. Brown, J. P. Bumpus, H. Chen, M. F. Hunzaker, J. Lee, M. Mann, F. Merhout, and A. Volfovsky, “Exposure to opposing views on social media can increase political polarization,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 115, no. 37, pp. 9216–9221, 2018.
- [131] J. McCoy and T. Rahman, “Polarized democracies in comparative perspective: Toward a conceptual framework,” in *International Political Science Association Conference*, vol. 26, (Poznan, Poland), pp. 16–42, 2016.
- [132] K. T. Poole and H. Rosenthal, *Congress: A Political-Economic History of Roll Call Voting*. Oxford, United Kingdom: Oxford University Press, 1997.
- [133] S. A. Rice, “The identification of blocs in small political bodies,” *American Political Science Review*, vol. 21, no. 3, pp. 619–627, 1927.
- [134] L. Sirovich, “A pattern analysis of the second Renquist U.S. Supreme Court,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, no. 13, pp. 7432–7437, 2003.
- [135] B. A. Huberman, D. M. Romero, and F. Wu, “Social networks that matter: Twitter under the microscope,” *First Monday*, vol. 14, no. 1–5, 2009. Available at <https://doi.org/10.5210/fm.v14i1.2317>.
- [136] D. Maynard and A. Funk, “Automatic detection of political opinions in tweets,” in *Extended Semantic Web Conference* (R. García-Castro, D. Fensel, and G. Antoniou, eds.), pp. 88–99, 2011.
- [137] J. DiGrazia, K. McKelvey, J. Bollen, and F. Rojas, “More tweets, more votes: Social media as a quantitative indicator of political behavior,” *PloS ONE*, vol. 8, no. 11, p. e79449, 2013.
- [138] D. Preoțiu-Pietro, Y. Liu, D. Hopkins, and L. Ungar, “Beyond binary labels: Political ideology prediction of Twitter users,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Vancouver, Canada), pp. 729–740, Association for Computational Linguistics, 2017.

- [139] M. Lai, M. Tambuscio, V. Patti, G. Ruffo, and P. Rosso, “Stance polarity in political debates: A diachronic perspective of network homophily and conversations on Twitter,” *Data & Knowledge Engineering*, vol. 124, p. 101738, 2019.
- [140] I. Waller and A. Anderson, “Quantifying social organization and political polarization in online platforms,” *Nature*, vol. 600, pp. 264–268, 2021.
- [141] E. Huls and J. Varwijk, “Political bias in TV interviews,” *Discourse & Society*, vol. 22, no. 1, pp. 48–65, 2011.
- [142] I. V. Kozitsin, “Opinion dynamics of online social network users: A micro-level analysis,” *The Journal of Mathematical Sociology*, 2021.
- [143] A. K. Dixit and J. W. Weibull, “Political polarization,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 104, no. 18, pp. 7351–7356, 2007.
- [144] A. Panda, D. Siddarth, and J. Pal, “COVID, BLM, and the polarization of US politicians on Twitter,” *ArXiv:2008.03263*, 2020.
- [145] M. Prior, “Media and political polarization,” *Annual Review of Political Science*, vol. 16, pp. 101–127, 2013.
- [146] A. Boche, J. B. Lewis, A. Rudkin, and L. Sonnet, “The new [Voteview.com](http://Voteview.com): Preserving and continuing Keith Poole’s infrastructure for scholars, students and observers of Congress,” *Public Choice*, vol. 176, no. 1–2, pp. 17–32, 2018.
- [147] M. Kitchener, N. Anantharama, S. D. Angus, and P. A. Raschky, “Predicting political ideology from digital footprints,” *ArXiv:2206.00397*, 2022.
- [148] Z. Xiao, J. Zhu, Y. Wang, P. Zhou, W. Lam, M. A. Porter, and Y. Sun, “Detecting political biases of named entities and hashtags on Twitter,” *arXiv:2209.08110*, 2022.
- [149] J. Gordon, M. Babaeianjelodar, and J. Matthews, “Studying political bias via word embeddings,” in *Companion Proceedings of the Web Conference 2020*, pp. 760–764, 2020.
- [150] G. Pennycook and D. G. Rand, “Fighting misinformation on social media using crowdsourced judgments of news source quality,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 116, no. 7, pp. 2521–2526, 2019.
- [151] J. Allen, B. Howland, M. Mobius, D. Rothschild, and D. J. Watts, “Evaluating the fake news problem at the scale of the information ecosystem,” *Science Advances*, vol. 6, no. 14, p. eaay3539, 2020.

- [152] A. Ferrario and M. Nägelin, “The art of natural language processing: classical, modern and contemporary approaches to text document classification,” *Modern and Contemporary Approaches to Text Document Classification (March 1, 2020)*, 2020.
- [153] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černocký, “Strategies for training large scale neural network language models,” in *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pp. 196–201, Institute of Electrical and Electronics Engineers, 2011.
- [154] W. Yin, K. Kann, M. Yu, and H. Schütze, “Comparative study of CNN and RNN for natural language processing,” *ArXiv:1702.01923*, 2017.
- [155] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, “Efficient processing of deep neural networks: A tutorial and survey,” *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.
- [156] M. Iyyer, P. Enns, J. Boyd-Graber, and P. Resnik, “Political ideology detection using recursive neural networks,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1113–1122, 2014.
- [157] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, “Recent advances in recurrent neural networks,” *ArXiv:1801.01078*, 2017.
- [158] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, 2013.
- [159] V. Otero, “Interactive media bias chart.” Available at <https://adfontesmedia.com/interactive-media-bias-chart/> [accessed 26 September 2020], Jan 2022. Accessed 26 September 2020.
- [160] P. Widmer, S. Galletta, and E. Ash, “Media slant is contagious,” *ArXiv:2202.07269*, 2022.
- [161] J. Littman, L. Wrubel, D. Kerchner, and Y. Bromberg Gaber, “News outlet tweet ids,” 2017.
- [162] H. Z. Brooks and M. A. Porter, “A model for the influence of media on the ideology of content in online social networks,” *Physical Review Research*, vol. 2, p. 023041, 2020.
- [163] A. van Loon, “Three families of automated text analysis,” May 2022. Available at [osf.io/preprints/socarxiv/htnej](https://osf.io/preprints/socarxiv/htnej).
- [164] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, “Text classification algorithms: A survey,” *Information*, vol. 10, no. 4, p. 150, 2019.

- [165] E. Cambria and B. White, “Jumping NLP curves: A review of natural language processing research,” *IEEE Computational Intelligence Magazine*, vol. 9, no. 2, pp. 48–57, 2014.
- [166] A. Bakliwal, P. Arora, A. Patil, and V. Varma, “Towards enhanced opinion classification using NLP techniques.,” in *Proceedings of the Workshop on Sentiment Analysis where AI meets Psychology (SAAIP 2011)*, pp. 101–107, 2011.
- [167] M. Kanakaraj and R. M. R. Guddeti, “Performance analysis of ensemble methods on Twitter sentiment analysis using NLP techniques,” in *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)*, pp. 169–170, Institute of Electrical and Electronics Engineers, 2015.
- [168] C. Troussas, M. Virvou, K. J. Espinosa, K. Llaguno, and J. Caro, “Sentiment analysis of Facebook statuses using naive Bayes classifier for language learning,” in *IISA 2013*, pp. 1–6, 2013.
- [169] A. P. Gopi, R. N. S. Jyothi, V. L. Narayana, and K. S. Sandeep, “Classification of tweets data based on polarity using improved RBF kernel of SVM,” *International Journal of Information Technology*, 2020. Available at <https://doi.org/10.1007/s41870-019-00409-4>.
- [170] T. Mikolov, M. Karafiát, L. Burget, J. Cernocky, and S. Khudanpur, “Recurrent neural network based language model,” in *Interspeech*, vol. 2, (Makuhari, Chiba, Japan), pp. 1045–1048, 2010.
- [171] K. Kawakami, *Supervised sequence labelling with recurrent neural networks*. PhD thesis, Technical University of Munich, 2008.
- [172] A. Go, R. Bhayani, and L. Huang, “Twitter sentiment classification using distant supervision,” *CS224N Project Report, Stanford University*, 2009. Available at <https://www-cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf>.
- [173] C. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press, 1999.
- [174] H. Zhang, “The optimality of naive bayes,” in *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*, pp. 562–567, 2004.
- [175] H. Bhavsar and M. H. Panchal, “A review on support vector machine for data classification,” *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 1, no. 10, pp. 185–189, 2012.
- [176] M. Sheykhmousa, M. Mahdianpari, H. Ghanbari, F. Mohammadimanesh, P. Ghamisi, and S. Homayouni, “Support vector machine versus random forest for remote sensing image classification: A meta-analysis and systematic review,”

*IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 6308–6325, 2020.

- [177] D. C. Toledo-Perez, J. Rodriguez-Resendiz, R. A. Gomez-Loenzo, and J. C. Jauregui-Correa, “Support vector machine-based EMG signal classification techniques: A review,” *Applied Sciences*, vol. 9, no. 20, p. 4402, 2019.
- [178] J. D. M. Rennie and N. Srebro, “Loss functions for preference levels: Regression with discrete ordered labels,” in *IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling*, 2005.
- [179] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. New York, NY, USA: Routledge, 2017.
- [180] M. Bilal, H. Israr, M. Shahid, and A. Khan, “Sentiment classification of roman-urdu opinions using naive Bayesian, decision tree and KNN classification techniques,” *Journal of King Saud University-Computer and Information Sciences*, vol. 28, no. 3, pp. 330–344, 2016.
- [181] A. S. Zharmagambetov and A. A. Pak, “Sentiment analysis of a document using deep learning approach and decision trees,” in *2015 Twelve International Conference on Electronics Computer and Computation (ICECCO)*, pp. 1–4, 2015.
- [182] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015.
- [183] M. Ghiassi, J. Skinner, and D. Zimbra, “Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network,” *Expert Systems with applications*, vol. 40, no. 16, pp. 6266–6282, 2013.
- [184] A. Sharma and S. Dey, “An artificial neural network based approach for sentiment analysis of opinionated text,” in *Proceedings of the 2012 ACM Research in Applied Computation Symposium*, pp. 37–42, 2012.
- [185] A. Sharma and S. Dey, “A document-level sentiment analysis approach using artificial neural network and sentiment lexicons,” *ACM SIGAPP Applied Computing Review*, vol. 12, no. 4, pp. 67–75, 2012.
- [186] Y. Wang, M. Huang, X. Zhu, and L. Zhao, “Attention-based LSTM for aspect-level sentiment classification,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 606–615, 2016.
- [187] T. Young, D. Hazarika, S. Poria, and E. Cambria, “Recent trends in deep learning based natural language processing [review article],” *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018.
- [188] D. Tang, B. Qin, X. Feng, and T. Liu, “Target-dependent sentiment classification with long short term memory,” *ArXiv:1512.01100*, 2015.

- [189] DocNow, “Hydrator.” Available at <https://github.com/docnow/hydrator> (accessed 26 September 2022), Jan 2020.
- [190] A. Gulli and S. Pal, *Deep Learning with Keras*. Birmingham, United Kingdom: Packt Publishing, 2017.
- [191] G. Rao, W. Huang, Z. Feng, and Q. Cong, “LSTM with sentence representations for document-level sentiment classification,” *Neurocomputing*, vol. 308, pp. 49–57, 2018.
- [192] Y. Yu, X. Si, C. Hu, and J. Zhang, “A review of recurrent neural networks: LSTM cells and network architectures,” *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, 2019.
- [193] Z. Zhao, W. Chen, X. Wu, P. C. Chen, and J. Liu, “LSTM network: A deep learning approach for short-term traffic forecast,” *IET Intelligent Transport Systems*, vol. 11, no. 2, pp. 68–75, 2017.
- [194] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.
- [195] A. F. Agarap, “Deep learning using rectified linear units (ReLU),” *ArXiv:1803.08375*, 2018.
- [196] M. Chaudhuri, “Ethics in the new media, print media and visual media landscape,” *Journal of Global Communication*, vol. 7, no. 1, pp. 84–95, 2014.