Robust Feature Extraction via ℓ_∞-Norm based Nonnegative Tucker Decomposition

Bilian Chen, Jiewen Guan, Zhening Li, and Zhehao Zhou

***Abstract*—Feature extraction plays an indispensable role in image and video technology. However, it is difficult for traditional matrix based feature extraction methods to handle massive multi-dimensional data. This, alongside with the ubiquitous uncertainty (noise) in real-world data, resulted in many robust tensor based feature extraction models. However, these existing models did not consider the *worst-case* model performance (i.e., the largest fitting error among all samples), which is critically important from a robust optimization perspective. In this paper, we propose a novel robust feature extraction model via ℓ_∞-norm based nonnegative Tucker decomposition. The model is to minimize the maximum sample fitting error so as to overcome the influence of data uncertainty. Although the new model is nonconvex and nonsmooth, we design an effective iterative optimization algorithm with theoretical guarantee on its convergence for it. The performance of the new model on five real-world benchmark object classification and face recognition datasets under various corruption scenarios are evaluated, and the experimental results show the excellence of the new model by comparing to many existing models.**

***Index Terms*—Feature extraction, classification, robust optimization, nonnegative Tucker decomposition, tensors.**

I. Introduction

FEATURE extraction is a fundamental topic in many applied research areas, such as data mining, machine learning and pattern recognition, as well as image and video processing. With the advancement of data acquisition technology, massive multi-dimensional data in tensor formats are generated in various realistic scenarios, such as reconstructed images [1], video data [2], social networks [3], and multi-channel electroencephalography (EEG) [4]. In the recent decade, tensor decomposition has become an effective method for extracting features from multi-dimensional data. Among all tensor decomposition techniques, Tucker decomposition [5], which decomposes a tensor into a core tensor and multiple factor matrices, has attracted the most attention. Due to the nature of the factorization, the core tensor of Tucker decomposition is commonly regarded as the extracted features. There are many Tucker decomposition based feature extraction models in the literature, such as the multilinear principal component analysis (MPCA) [6] and higher-order discriminant analysis (HODA) [7].

Recently, it has been discovered that the quality of the extracted features can be further improved by adding prior data knowledge onto Tucker decomposition. As an example, the nonnegativity constraint can be imposed on Tucker decomposition to obtain better extracted features [8] since real-world data are usually nonnegative. Similarly, adding low-rank constraints on Tucker decomposition can also lead to more informative extracted features [9]–[11]. Moreover, to ensure that samples in the original data space and the projected feature space have consistent structural information, the local geometrical structures of data can also be integrated into Tucker decomposition [12]–[15]. Although these models incorporate different types of prior data knowledge, they share a common trait: The ℓ_2-norm (the sum of squared errors) is adopted to compute the overall fitting error to be minimized. We denote this class of models as ℓ_2-norm based models.

In reality, the observed data are often uncertain. For example, one is usually unable to exactly measure the statistics of a signal, and the observations of different measuring trials often fluctuate. Although the noise inside data is often inconspicuous, the crucial fact is that, even a slight change in data can dramatically influence the optimal solutions of the corresponding optimization problem [16]. This is also the case for feature extraction due to measurement limitations or unintentional corruptions of data. In order to avoid this drawback, several researchers proposed ℓ_1-norm (the sum of non-squared errors) based models, which are generally more robust than ℓ_2-norm based models. Cao *et al.* proposed an ℓ_1-norm based robust tensor decomposition model [17] for face clustering. Markopoulos's team studied various robust Tucker decomposition models such as ℓ_1-Tucker decomposition [18]–[20], ℓ_1-HOOI [21] and ℓ_1-HOSVD [22].

In robust optimization theory [16], [23], an effective strategy to handle data uncertainty is to emphasize the *worst-case* model performance, which is neglected by both ℓ_2- and ℓ_1-norm based models. Inspired by this idea, in this paper, we propose a novel ℓ_∞-norm based robust nonnegative Tucker decomposition model for feature extraction. The key idea of the ℓ_∞ model is to minimize the maximum fitting error among samples so as to suppress the negative effects caused by data uncertainty (recall that a slight change in data can drastically influence the optimal solutions), which guarantees that the fitting errors of all samples are uniformly well-controlled. We remark that it is clear that this functionality cannot be fulfilled by commonly-used norms such as ℓ_1 norm, $\ell_{2,1}$ norm and nuclear norm, etc. However, the ℓ_∞ model admits a nonconvex

This work was supported in part by the Youth Innovation Fund of Xiamen (Grant No. 3502Z20206049) and the National Natural Science Foundation of China (Grant No. 61836005). *(Corresponding author: Bilian Chen)*

Bilian Chen, Jiewen Guan, and Zhehao Zhou are with the Department of Automation, Xiamen University, Xiamen 361005, China, and with the Xiamen Key Laboratory of Big Data Intelligent Analysis and Decision-making, Xiamen 361005, China (emails: blchen@xmu.edu.cn; jwguan@stu.xmu.edu.cn; zhehaozhou@foxmail.com).

Zhening Li is with School of Mathematics and Physics, University of Portsmouth, Portsmouth PO1 3HF, UK (email: zheningli@gmail.com).

and nonsmooth objective function, which poses challenge on its optimization. In order to effectively optimize it, we propose an iterative optimization algorithm based on second-order cone program (SOCP), and then theoretically prove its convergence and analyze its computational complexity. The performance of the proposed ℓ_∞ model is tested via image classification and face recognition problems under various corruption conditions, with comparisons to ℓ_1- and ℓ_2-norm based nonnegative Tucker decomposition models and nine classical and state-of-the-art (SOTA) feature extraction methods. In summary, the main contributions of the paper are highlighted as follows:

1) We propose a novel ℓ_∞-norm based robust nonnegative Tucker decomposition model for feature extraction. The model can effectively suppress the negative influence caused by data uncertainty. The generality of the ℓ_∞ model makes it flexible in extension, such as adding prior data knowledge.

2) We develop an effective iterative optimization algorithm to solve the ℓ_∞ model. We also theoretically show the convergence of the algorithm and analyze its computational complexity. We remark that our algorithm has a clear difference in comparison with existing iterative updating strategies as a necessary SOCP based subroutine is also involved here.

3) We design a variety of corruptions to test the effectiveness and robustness of the proposed ℓ_∞ model based on five real-world benchmark datasets. Experimental results show that our model has a superior performance over the competitors.

The remainder of the paper is organized as follows. We briefly overview related work in Section II and introduce preliminaries in Section III. Then, we formally propose the ℓ_∞ model in Section IV and design its optimization algorithm in Section V. In Section VI, we perform comprehensive experiments to test the new model. Finally, we conclude this paper in Section VII.

II. Related Work

In this section, we review related nonrobust/robust tensor decomposition based feature extraction methods. Since feature extraction is a general topic, our review is not only restricted to Tucker decomposition based methods.

A. Nonrobust Tensor Decomposition based Feature Extraction

Tensor decomposition is an effective and powerful method to extract features from multi-dimensional data. Phan and Cichocki [8] imposed orthogonality and nonnegativity constraints in HODA [7], and the resulted model showed promising results in image data and EEG data. Idaji *et al.* [24] proposed higher-order spectral regression discriminant analysis (HOSRDA) model, which transformed HODA into a regression problem. Jukic *et al.* [25] proposed a new tensor decomposition model based on mutual information maximization, which can include higher-order statistical information in data. Li *et al.* [14] proposed a graph regularized tensor decomposition model to preserve the local geometrical structures of data. Yin and Ma [15] adopted the Laplacian eigenmaps [26] as a regularization term to improve Tucker decomposition, so as to capture the nonlinear structure of data. In order to extract features from incomplete tensor data, Shi *et al.* [27] proposed a tensor decomposition model that can perform feature extraction and missing entry estimation simultaneously. Fu *et al.* [9] constructed a tensor decomposition based low-rank sparse representation model by adding low-rank constraints on the factor matrices and a sparse constraint on the core tensor. We remark that low-rankness is an important objective that has inspired many critical techniques in modern machine learning such as making deep neural networks lightweight [28]. Zhou *et al.* [29] proposed a multiple rank-R decomposition method to learn compact representations for dynamic texture video coding. Khokher *et al.* [30] employed tensor Tucker decomposition to extract features for dynamic scene recognition. Liu *et al.* [31] proposed to jointly optimize CANDECOMP/PARAFAC (CP) rank and Tucker rank for low-rank tensor approximation. Xu *et al.* [32] proposed a novel reconstruction method for hyperspectral computational imaging based on collaborative Tucker3 tensor decomposition. He *et al.* [33] proposed a streaming tensor ring decomposition based method for visual data recovery. These feature extraction methods also have applications in other fields. For example, Tang *et al.* further studied tensor completion based methods for social-aware image tag refinement [34] and large-scale social image retagging [35]. Lebedev *et al.* [36] proposed a simple method for accelerating the computation of convolutional neural networks based on fine-tuned tensor CP decomposition. However, the aforementioned methods mainly focus on adding different prior data knowledge on tensor decomposition to improve the quality of the extracted features, but ignore the uncertainty (noise) in data by using the ℓ_2-norm based error.

B. Robust Tensor Decomposition based Feature Extraction

To extract features from noisy data, researchers proposed many ℓ_1-norm based tensor decomposition models in recent years. It turns out that the ℓ_1-norm is more robust to noise than the ℓ_2-norm. Zhang and Ding [37] replaced the ℓ_2-norm of the orthogonal Tucker decomposition by the ℓ_1-norm to suppress the impact caused by data noise. Markopoulos *et al.* [19] designed two efficient algorithms for the ℓ_1-norm based Tucker2 model [18]. Markopoulos *et al.* [22] proposed the ℓ_1-norm based HOSVD model and Chachlakis *et al.* [21] proposed the ℓ_1-norm based HOOI model. Wu [38] developed a streaming tensor low-rank representation method with error term regularized by ℓ_1-norm, which is capable of handling dynamic data. On the other hand, tensor singular value decomposition (t-SVD) [39] based robust feature extraction models also attracted much attention. Lu *et al.* [40] proposed the tensor robust principal component analysis (TRPCA) model, which simultaneously optimizes the t-SVD based nuclear norm of the reconstructed data and the ℓ_1-norm of errors. However, TRPCA cannot effectively deal with outliers, as it uses the ℓ_1-norm instead of the $\ell_{2,1}$-norm. To this end, Zhou and Feng [41] proposed the outlier-robust tensor PCA (ORTPCA) that adopts the $\ell_{2,1}$-norm to compute the error. Besides, Jia *et al.* [42] adopted the low-rank tensor learning with $\ell_{2,1}$-norm regularization to recover 'missing' knowledge in cross-modality action recognition. Chen *et al.* [43] proposed to learn the low-rank tensor representation and affinity matrix in a joint manner, and imposed an $\ell_{2,1}$-norm based regularizer on top of them for alleviating the negative effects led by noise and outliers. Jia *et al.* [44] designed a specific tensor low-rank representation method with $\ell_{2,1}$-norm regularization, which is tailored for multi-view spectral clustering. Jia *et al.* [45] proposed a low-rank tensor representation method with $\ell_{2,1}$-norm regularization for semi-supervised subspace clustering, which globally explores the information of supervision. Beyond the above norms, there are also other norms applied to enhance robustness. For example, Liu *et al.* [46] employed the ℓ_p-regression where $p \in (0, 2)$ to increase the outlier resistance for low-rank tensor completion.

We highlight that [43], [44], [46] are the most closely related works to ours that were published in this journal. However, among all the above methods, no work has considered the ℓ_∞-norm to enhance system robustness by maintaining the worst-case model performance, which is the most distinctive part of our model. We also stress here that the ℓ_∞-norm used in our scheme has also been adopted in other fields to enhance robustness, such as deep learning [47]–[49], adversarial training [50], control theory [51], etc. However, as these fields are not quite related to tensor factorization, we do not elaborate on them here.

III. Preparation

A. Notations and Tensor Operations

Throughout this paper, we uniformly use calligraphic letters, capital letters, boldface lowercase letters, and non-bold lowercase letters to denote tensors, matrices, vectors, and scalars. For example, a tensor $\mathcal{G}$, a matrix A, a vector $\mathbf{y}$, and a scalar i. We use subscript to denote an element of a tensor, a matrix, or a vector, e.g., $\mathcal{G}_{ijk}$ as the (i, j, k)th entry of a third-order tensor $\mathcal{G}$, A_{ij} as the (i, j)th entry of a matrix A, y_i as the ith entry of a vector $\mathbf{y}$. The identity matrix in $\mathbb{R}^{d\times d}$ is denoted by I_d. The Kronecker product is denoted as $\otimes$ and the element-wise product is denoted by $*$. For a matrix $X = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{m\times n}$, $\mathrm{vec}(X) = [\mathbf{x}_1^T, \dots, \mathbf{x}_n^T]^T \in \mathbb{R}^{mn}$. For a dth order tensor $\mathcal{G} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$ with $d \geq 3$, we denote its mode-k matricization (or unfolding) as $G_{(k)} \in \mathbb{R}^{n_k\times\prod_{1\leq i\leq d,\, i\neq k} n_i}$, in which the $(i_1, i_2, \dots, i_d)$th entry of the tensor $\mathcal{G}$ is mapped to the (i_k, j)th entry of the matrix $G_{(k)}$ where

$$j = 1 + \sum_{1\leq s\leq d,\, s\neq k} (i_s - 1) \prod_{1\leq t\leq s-1,\, t\neq k} n_t.$$

The k-rank of $\mathcal{G}$, denoted by $\mathrm{rank}_k(\mathcal{G})$, is defined as the rank of $\mathcal{G}_{(k)}$. A dth order tensor $\mathcal{G}$ with $\mathrm{rank}_k(\mathcal{G}) = r_k$ for $k = 1, 2, \dots, d$ is called a rank-$(r_1, r_2, \dots, r_d)$ tensor. The mode-k product of $\mathcal{G}$ by a matrix $U \in \mathbb{R}^{m\times n_k}$, denoted by $\mathcal{G} \times_k U \in \mathbb{R}^{n_1\times\cdots\times n_{k-1}\times m\times n_{k+1}\times\cdots\times n_d}$, is defined by

$$(\mathcal{G} \times_k U)_{i_1\dots i_{k-1} j i_{k+1}\dots i_d} = \sum_{i_k=1}^{n_k} \mathcal{G}_{i_1\dots i_d} U_{j i_k}. \tag{1}$$

It is easy to verify that

$$\mathcal{Y} = \mathcal{G} \times_k U \Longleftrightarrow Y_{(k)} = U G_{(k)}.$$

The Frobenius norm of a tensor is defined as

$$\|\mathcal{G}\|_F := \left(\sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} {\mathcal{G}_{i_1 i_2\dots i_d}}^2 \right)^{1/2}.$$

For more details about tensor operations, the readers are referred to the review paper [52].

B. Nonnegative Tucker Decomposition

Nonnegative Tucker decomposition (NTD) [53] is a commonly used model for feature extraction. Given a data sample $\mathcal{X}^{(0)} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$ and the dimension of the core tensor $r_1 \times r_2 \times \cdots \times r_d$, NTD of $\mathcal{X}^{(0)}$ is the following optimization model

$$\begin{aligned} \min_{\mathcal{G}^{(0)}, A^{(j)}} \quad & \left\| \mathcal{X}^{(0)} - \mathcal{G}^{(0)} \times_1 A^{(1)} \times_2 A^{(2)} \cdots \times_d A^{(d)} \right\|_F \\ \text{s.t.} \quad & \mathcal{G}^{(0)} \geq 0,\ \{A^{(j)}\}_{j=1}^d \geq 0, \end{aligned} \tag{2}$$

where $\mathcal{G}^{(0)} \in \mathbb{R}^{r_1\times r_2\times\cdots\times r_d}$ is the core tensor for $\mathcal{X}^{(0)}$, $A^{(j)} \in \mathbb{R}^{n_j\times r_j}$ is the mode-j factor matrix and the mode-j product $\times_j$ is defined in (1) for $j = 1, 2, \dots, d$.

C. Workflow of Feature Extraction from Tensor Data via NTD

We here briefly introduce the workflow of feature extraction introduced in the literature [54], which will also be used in this paper. Consider a training data tensor $\mathcal{X} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_d\times k}$ stacked by k multiway training samples $\{\mathcal{X}^{(i)}\}_{i=1}^k$ belonging to c categories, and a test data tensor $\mathcal{Y} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_d\times t}$ stacked by t multiway test samples $\{\mathcal{Y}^{(i)}\}_{i=1}^t$ also belonging to the same c categories. The goal of feature extraction via NTD is to learn feature extractors $\{A^{(j)}\}_{j=1}^d$ from the training data $\mathcal{X}$ via the NTD model (or its variants) and apply the learned feature extractors to extract features from the test data $\mathcal{Y}$ [54]. This procedure is also vividly illustrated in Figure 1.

D. Feature Extraction via ℓ_2-Norm and ℓ_1-Norm based NTD

Consider a training data tensor $\mathcal{X} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_d\times k}$. The traditional ℓ_2-norm based feature extraction model aims to solve the following optimization problem based on the Tucker decomposition model (2)

$$\begin{aligned} \min_{\mathcal{G}^{(i)}, A^{(j)}} \quad & \textstyle\sum_{i=1}^k \left\| \mathcal{X}^{(i)} - \mathcal{G}^{(i)} \times_1 A^{(1)} \cdots \times_d A^{(d)} \right\|_F^2 \\ \text{s.t.} \quad & \{\mathcal{G}^{(i)}\}_{i=1}^k \geq 0,\ \{A^{(j)}\}_{j=1}^d \geq 0, \end{aligned} \tag{3}$$

where $\mathcal{G}^{(i)} \in \mathbb{R}^{r_1\times r_2\times\cdots\times r_d}$ is the core tensor of $\mathcal{X}^{(i)}$ for $i = 1, 2, \dots, k$, and $A^{(j)} \in \mathbb{R}^{n_j\times r_j}$ is the mode-j factor matrix for $j = 1, 2, \dots, d$. For brevity, we call the model (3) to be the ℓ_2 model in this paper. As we can see, the ℓ_2 model computes the sum of all *squared* errors, for which the attention (weight) paid to different samples is the same. Due to the square involved, the model is sensitivity to data noise. But thanks to the smoothness of the objective function, the ℓ_2 model is relatively easy to optimize.

In a similar vein, the ℓ_1-norm based feature extraction model takes the following form

$$\begin{aligned} \min_{\mathcal{G}^{(i)}, A^{(j)}} \quad & \textstyle\sum_{i=1}^k \left\| \mathcal{X}^{(i)} - \mathcal{G}^{(i)} \times_1 A^{(1)} \cdots \times_d A^{(d)} \right\|_F \\ \text{s.t.} \quad & \{\mathcal{G}^{(i)}\}_{i=1}^k \geq 0,\ \{A^{(j)}\}_{j=1}^d \geq 0, \end{aligned} \tag{4}$$

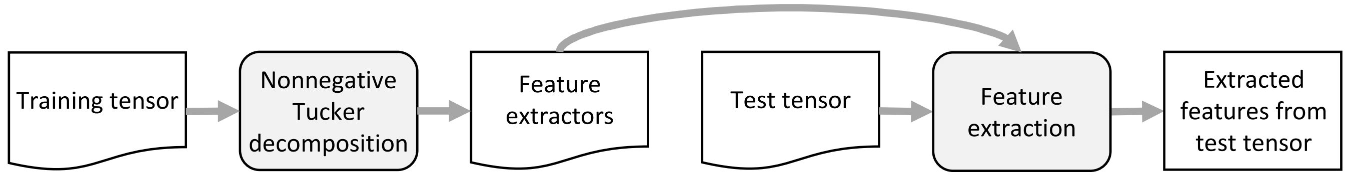


Fig. 1: Workflow of feature extraction from tensor data via nonnegative Tucker decomposition.

which is called the ℓ_1 model in this paper for short. In contrast to the ℓ_2 model, the ℓ_1 model directly computes the sum of all *non-squared* errors. This increases the robustness of the model to noise. However, we observe that the ℓ_1 model also pays the same attention to different samples. Besides, we also see that the ℓ_1 model introduces nonsmoothness onto its objective function, which increases optimization difficulty.

IV. Robust Feature Extraction via ℓ_∞-Norm based Nonnegative Tucker Decomposition

As discussed above, the ℓ_2 model (3) is relatively easy to solve since the objective function is smooth but it is sensitive to data noise which directly affects the quality of the extracted features. The ℓ_1 model (4) can mitigate the disadvantage of the ℓ_2 model to some extent by using the ℓ_1-norm but it still underestimates the importance of different samples since it pays the same attention to every sample. This motivates us to develop new methods to emphasize more important samples so as to further enhance the ability for handling data uncertainty in feature extraction. We resort to the techniques in robust optimization theory [16] to make an improvement. The philosophy of robust optimization is to guarantee that even in the worst-case scenario the model is still effective. Such an idea exactly falls into our needs and suggests to control the largest fitting error of Tucker decomposition among all samples, i.e., $\max_{1\le i\le k} \|\mathcal{X}^{(i)} - \mathcal{G}^{(i)} \times_1 A^{(1)} \cdots \times_d A^{(d)}\|_F$. Keeping the notations defined in Section III-D, the above discussion leads us to the following ℓ_∞ model

$$\begin{array}{ll} \min\limits_{\mathcal{G}^{(i)},A^{(j)}} & \max\limits_{1\le i\le k} \left\|\mathcal{X}^{(i)} - \mathcal{G}^{(i)} \times_1 A^{(1)} \cdots \times_d A^{(d)}\right\|_F \\ \text{s.t.} & \{\mathcal{G}^{(i)}\}_{i=1}^k \ge 0,\ \{A^{(j)}\}_{j=1}^d \ge 0. \end{array} \tag{5}$$

We remark that, distinct from the ℓ_2 and ℓ_1 models mentioned above, the attention in (5) has been paid to the sample with the largest fitting error, which exactly makes up the shortage in the ℓ_2 and ℓ_1 models. As a consequence, in the ℓ_∞ model, even for the worst data sample its Tucker decomposition performance can be guaranteed and the robustness of the whole feature extraction process can thus be increased.

Although the ℓ_∞ model is promising in terms of its model functionality, its objective function is nonsmooth and this makes the optimization hard. In the next section, we design an effective iterative algorithm based on SOCP to solve the ℓ_∞ model.

V. Algorithm and Analysis

A. Solution Method

The proposed ℓ_∞ model (5) is nonconvex and its objective function is nonsmooth, making it difficult to be solved directly. Therefore, we propose to solve $\mathcal{G}^{(i)}$'s and $A^{(j)}$'s iteratively and alternatively. First of all, we decompose (5) into the following two subproblems.

*1) **Update $\mathcal{G}^{(i)}$'s by fixing $A^{(j)}$'s:*** This is to solve a set of problems in the following form

$$\begin{array}{ll} \min\limits_{\mathcal{G}^{(i)}} & \left\|\mathcal{X}^{(i)} - \mathcal{G}^{(i)} \times_1 A^{(1)} \cdots \times_d A^{(d)}\right\|_F^2 \\ \text{s.t.} & \mathcal{G}^{(i)} \ge 0, \end{array} \tag{6}$$

for $i = 1, 2, \ldots, k$. Note that the square added to the objective function of (6) will not affect the optimal solutions of (6) but make the optimization process easier. Since (6) has the same format for every i, we may combine all these problems into a whole, as a simpler form below in the analysis

$$\begin{array}{ll} \min\limits_{\mathcal{G}} & \left\|\mathcal{X} - \mathcal{G} \times_1 A^{(1)} \cdots \times_d A^{(d)}\right\|_F^2 \\ \text{s.t.} & \mathcal{G} \ge 0, \end{array} \tag{7}$$

where $\mathcal{G} \in \mathbb{R}^{r_1\times r_2\times\cdots\times r_d\times k}$ is obtained by stacking $\mathcal{G}^{(i)}$'s. It is not difficult to show that the solution to (7) gives the solutions to (6) for all $i = 1, 2, \ldots, k$ in one hit. The updating rule to solve (7) can be derived from its nonnegative matrix factorization (NMF) counterpart, as

$$\mathcal{G} \leftarrow \mathcal{G} * \frac{\mathcal{X} \times_1 A^{(1)T} \cdots \times_d A^{(d)T}}{\mathcal{G} \times_1 A^{(1)T}A^{(1)} \cdots \times_d A^{(d)T}A^{(d)}}. \tag{8}$$

*2) **Update $A^{(j)}$'s by fixing $\mathcal{G}^{(i)}$'s:*** This is to solve a set of problems in the following form

$$\begin{array}{ll} \min\limits_{A^{(j)}} & \max\limits_{1\le i\le k} \left\|\mathcal{X}^{(i)} - \mathcal{G}^{(i)} \times_1 A^{(1)} \cdots \times_d A^{(d)}\right\|_F \\ \text{s.t.} & A^{(j)} \ge 0, \end{array} \tag{9}$$

for $j = 1, 2, \ldots, d$. Problem (9) can be rewritten in the matrix form as

$$\begin{array}{ll} \min\limits_{A^{(j)}} & \max\limits_{1\le i\le k} \left\|X_{(j)}^{(i)} - A^{(j)}G_{(j)}^{(i)}A^{(\backslash j)T}\right\|_F \\ \text{s.t.} & A^{(j)} \ge 0, \end{array}$$

where $A^{(\backslash j)} := A^{(d)} \otimes \cdots \otimes A^{(j+1)} \otimes A^{(j-1)} \otimes \cdots \otimes A^{(1)}$. This model can then be equivalently transformed to an SOCP

$$\begin{array}{lll} \min\limits_{A^{(j)}} & t_j & \\ \text{s.t.} & \left\|X_{(j)}^{(i)} - A^{(j)}G_{(j)}^{(i)}A^{(\backslash j)T}\right\|_F \le t_j & i = 1, 2, \ldots, k \\ & A^{(j)} \ge 0. & \end{array} \tag{10}$$

Algorithm 1: Optimization algorithm for the ℓ_∞ model

1 **Input:** A training data tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d \times k}$ with intrinsic ranks $\{r_1, r_2, \ldots, r_d\}$ (which can be estimated by Algorithm 2) and a convergence threshold ϵ.
Output: Feature extractors $\{A^{(j)}\}_{j=1}^d$.
2 Randomly initialize $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d \times k} \geq 0$ and $A^{(j)} \in \mathbb{R}^{n_j \times r_j} \geq 0$ for $j = 1, 2, \ldots, d$;
3 **while** *change in objective function* $\geq \epsilon$ **do**
4   Update $\mathcal{G}$ by (8);
5   **for** $j = 1, 2, \ldots, d$ **do**
6   Update $A^{(j)}$ by solving (10);
7 **return** $\{A^{(j)}\}_{j=1}^d$.

Recall that the SOCP is a widely-used convex optimization model and can be solved quickly by existing convex program solvers, such as MOSEK [55] to be used in this paper.

In summary, the whole optimization algorithm to solve the ℓ_∞ model is illustrated in Algorithm 1. Specifically, we update the core tensor in Line 4 and the factor matrices in Lines 5-6.

B. Convergence Analysis

We now analyze the convergence of Algorithm 1 in this part. For convenience, we denote the objective function of (5) as $\mathcal{J}(\mathcal{G}, A^{(1)}, \ldots, A^{(d)})$.

Theorem V.1. *$\mathcal{J}(\mathcal{G}, A^{(1)}, \ldots, A^{(d)})$ is nonnegative and nonincreasing in each iteration of Algorithm 1. Therefore, $\mathcal{J}(\mathcal{G}, A^{(1)}, \ldots, A^{(d)})$ will converge to a local minimum.*

Proof. It is obvious that $\mathcal{J}(\mathcal{G}, A^{(1)}, \ldots, A^{(d)})$ is bounded below by zero as it is the maximum of several Frobenius norms. We next show that $\mathcal{J}(\mathcal{G}, A^{(1)}, \ldots, A^{(d)})$ is nonincreasing in each iteration of Algorithm 1 by two parts.
1) **Update $\mathcal{G}$**: Since the updating rule of $\mathcal{G}$ is directly derived from its NMF counterpart, the convergence proof of NMF [56] can be easily applied to our case. We omit the proof here for the interest of space and conclude that

$$\mathcal{J}(\mathcal{G}_{t+1}, A_t^{(1)}, \ldots, A_t^{(d)}) \leq \mathcal{J}(\mathcal{G}_t, A_t^{(1)}, \ldots, A_t^{(d)}). \quad (11)$$

2) **Update $A^{(j)}$'s**: Since we update $A^{(j)}$ by solving an SOCP problem to obtain an optimal solution, it is guaranteed that $A_{t+1}^{(j)} = \arg\min_{A \geq 0} \mathcal{J}(\mathcal{G}_{t+1}, A_{t+1}^{(1)}, \ldots, A_{t+1}^{(j-1)}, A, A_t^{(j+1)}, \ldots, A_t^{(d)})$. By the optimality of $A_{t+1}^{(j)}$ to the SOCP and the feasibility of $A_t^{(j)}$ to the SOCP, it naturally holds that

$$\begin{aligned}&\mathcal{J}(\mathcal{G}_{t+1}, A_{t+1}^{(1)}, \ldots, A_{t+1}^{(j-1)}, A_{t+1}^{(j)}, A_t^{(j+1)}, \ldots, A_t^{(d)}) \\ \leq &\mathcal{J}(\mathcal{G}_{t+1}, A_{t+1}^{(1)}, \ldots, A_{t+1}^{(j-1)}, A_t^{(j)}, A_t^{(j+1)}, \ldots, A_t^{(d)}).\end{aligned}$$

As a result, when all $A^{(j)}$'s have been updated, we shall have

$$\mathcal{J}(\mathcal{G}_{t+1}, A_{t+1}^{(1)}, \ldots, A_{t+1}^{(d)}) \leq \mathcal{J}(\mathcal{G}_{t+1}, A_t^{(1)}, \ldots, A_t^{(d)}). \quad (12)$$

Combining (11) and (12), we have

$$\mathcal{J}(\mathcal{G}_{t+1}, A_{t+1}^{(1)}, \ldots, A_{t+1}^{(d)}) \leq \mathcal{J}(\mathcal{G}_t, A_t^{(1)}, \ldots, A_t^{(d)}),$$

which implies that the objective function of (5) is nonincreasing in each iteration. This further shows that $\mathcal{J}(\mathcal{G}, A^{(1)}, \ldots, A^{(d)})$ will converge to a local minimum. $\square$

C. Complexity Analysis

In this part, we analyze the computational complexity of Algorithm 1. Recall that d is the number of modes of the tensor data, k is the number of data samples, and n_j and r_j are the dimensions of the jth mode of the data and the core tensor, respectively. Here we assume that $r_j \ll n_j$ and $r_j \ll \prod_{1 \leq i \leq d, i \neq j} n_i$ for $j = 1, 2, \ldots, d$, which usually hold in reality.

Theorem V.2. *Given an acceptable duality gap ϵ from an SOCP solver, the computational complexity for one iteration of Lines 4-6 in Algorithm 1 is $\mathcal{O}(d(\prod_{i=1}^d n_i)^{3.5} k^{3.5} \ln(\epsilon^{-1}))$.*

Proof. The proof consists of two parts.
1) **Update $\mathcal{G}$**: This subproblem involves many tensor and matrix multiplications. Computing $A^{(j)T} A^{(j)}$ costs $\mathcal{O}(n_j {r_j}^2)$ time for $j = 1, 2, \ldots, d$. Computing the numerator and the denominator of (8) costs $\mathcal{O}(k \sum_{i=1}^d (\prod_{j=1}^i r_j)(\prod_{j=i}^d n_j))$ and $\mathcal{O}(k(\prod_{j=1}^d r_j) \sum_{i=1}^d r_i)$ time, respectively. Since $r_i \ll n_i$ for $i = 1, 2, \ldots, d$, the total time complexity of updating $\mathcal{G}$ is $\mathcal{O}(\sum_{i=1}^d n_i {r_i}^2 + k \sum_{i=1}^d r_i (\prod_{j=1}^{i-1} r_j)(\prod_{j=i}^d n_j))$.
2) **Update $A^{(j)}$'s**: Let us denote $\Psi_i = I_{n_j} \otimes A^{(\setminus j)} G_{(j)}^{(i)T}$ and then equivalently rewrite (10) as

$$\begin{aligned}\min_{\mathrm{vec}(A^{(j)T})} \quad & t_j \\ \text{s.t.} \quad & \left\| \Psi_i \,\mathrm{vec}\left(A^{(j)T}\right) - \mathrm{vec}\left(X_{(j)}^{(i)T}\right) \right\|_2 \leq t_j \quad i = 1, \ldots, k \\ & \mathrm{vec}\left(A^{(j)T}\right) \geq 0.\end{aligned}$$

In order to formulate the above SOCP model, $A^{(\setminus j)} G_{(j)}^{(i)T}$ needs to be computed in $\mathcal{O}((\prod_{1 \leq i \leq j, i \neq j} n_i)(\prod_{i=1}^d r_i))$ time for $j = 1, 2, \ldots, d$. Therefore, the total time complexity to compute $\{\Psi_i\}_{i=1}^k$ is $\mathcal{O}(k(\prod_{1 \leq i \leq d, i \neq j} n_i)(\prod_{i=1}^d r_i + r_j {n_j}^2))$. This SOCP has $k + n_j r_j$ second-order cone constraints, each having a dimension of $\prod_{i=1}^d n_i + 1$ or 2. According to the implementation in MOSEK for solving SOCPs [57], [58], given an acceptable duality gap $\epsilon > 0$, the computational complexity for updating $A^{(j)}$ is $\mathcal{O}((k(\prod_{i=1}^d n_i + 1) + 2 n_j r_j)^{3.5} \ln(\epsilon^{-1}))$, reduced to be $\mathcal{O}((\prod_{i=1}^d n_i)^{3.5} k^{3.5} \ln(\epsilon^{-1}))$. This easily dominates the previous computation complexity for $\{\Psi_i\}_{i=1}^k$. Therefore, the total computational complexity for updating $\{A^{(j)}\}_{j=1}^d$ is $\mathcal{O}(d(\prod_{i=1}^d n_i)^{3.5} k^{3.5} \ln(\epsilon^{-1}))$.

Combining the above two steps of analysis, the computational complexity for one iteration of Lines 4-6 in Algorithm 1 is $\mathcal{O}(d(\prod_{i=1}^d n_i)^{3.5} k^{3.5} \ln(\epsilon^{-1}))$ as the complexity of updating $\mathcal{G}$ is dominated by the one of updating $A^{(j)}$'s. $\square$

If we let $\prod_{j=1}^d n_j = n$ (the total number of elements in a sample), then the above computational complexity will be $\mathcal{O}(d n^{3.5} k^{3.5} \ln(\epsilon^{-1}))$.

VI. Experiments

In this section, we evaluate the performance of the proposed ℓ_∞ model following the workflow introduced in Section III-C. The experiments are implemented in MATLAB 2020b, and all experiments are run on a Ubuntu server with 3.70-GHz i9-10900K CPU, 64-GB main memory. We use MATLAB Tensor Toolbox 2.6 [59] whenever tensor operations are called. To solve the SOCP (10), we use MOSEK [55], a package for specifying and solving convex optimization problems. The source code of the proposed ℓ_∞ model is publicly available at https://github.com/zhzhouxmu/linf.

A. Datasets

We adopt five real-world benchmark object classification and face recognition datasets as below. These datasets are standard and have been extensively used in related fields.

- **COIL**[1]: COIL is a gray image database consisting of 1,440 images of 20 different objects each of which is associated with 72 images with different rotation degrees. For each object, we select 8 images for training and 64 images for testing.
- **YALE**[2]: YALE face database contains 165 grayscale images of 15 individuals, each of which has 11 images, showing changes in lighting conditions and facial expressions (e.g., normal, happy, etc.). For each individual, we select 8 images for training and 3 images for testing.
- **UMIST**[3]: The UMIST face database consists of 565 images of 20 people, where everyone covers a series of poses. Research objects in UMIST include race, gender and appearance. Since the numbers of images of different research objects are different, about one quarter of the images of each object are used for training and the remaining three quarters are used for testing.
- **COIL-100**[4]: COIL-100 is similar to COIL, and consists of 7,200 images of 100 different objects. For each object, we select 5 images for training and 67 images for testing.
- **YALE-B**[5]: YALE-B is similar to YALE, and contains 2414 grayscale images of 38 individuals. For each individual, we select 20 images for training and leave the remaining for testing.

For all the above datasets, we resize each image to 32×32 and then normalize each pixel to lie within 0 and 1 by dividing each pixel by the maximum one. The statistics of the datasets are listed in Table I, including the divided training and test data.

B. Noise Design

In this part, we design a variety of noise disturbance for the five datasets. The notations and corresponding explanations about the noise types are given as follows[6].

1) Noise for COIL: Four types of noise are designed for the COIL dataset, as described below.

- ms-n_1-n_2-n_3 (resp. ms-n_1-n_2-n_3-n_4): In every eight images of the training set, $n_1\%$, $n_2\%$ and $n_3\%$ (resp. $n_1\%$, $n_2\%$, $n_3\%$ and $n_4\%$) pixels are removed from the first three (resp. four) images, respectively.
- sp-n_1-n_2-n_3 (resp. sp-n_1-n_2-n_3-n_4): In every eight images of the training set, the first three (resp. four) images are corrupted by the *salt and pepper* noise with density $n_1\%$, $n_2\%$ and $n_3\%$ (resp. $n_1\%$, $n_2\%$, $n_3\%$ and $n_4\%$), respectively.

2) Noise for YALE: Two types of noise are designed for the YALE dataset, as described below.

- ms-n_1-n_2-n_3: In every eight images of the training set, $n_1\%$, $n_1\%$, $n_2\%$, $n_2\%$, $n_3\%$ and $n_3\%$ pixels are removed from the first six images, respectively.
- sp-n_1-n_2-n_3: In every eight images of the training set, the first six images are corrupted by the *salt and pepper* noise with density $n_1\%$, $n_1\%$, $n_2\%$, $n_2\%$, $n_3\%$ and $n_3\%$, respectively.

3) Noise for UMIST: Two types of noise are designed for the UMIST dataset, as described below.

- ms-n_1-n_2-n_3: In 45 randomly selected images of the training set, $n_1\%$, $n_2\%$ and $n_3\%$ pixels are removed from 25, 15 and 5 images, respectively.
- sp-n_1-n_2-n_3: In 45 randomly selected images of the training set, 25, 15, and 5 images are corrupted by the *salt and pepper* noise with density $n_1\%$, $n_2\%$ and $n_3\%$, respectively.

4) Noise for COIL-100: Two types of noise are designed for the COIL-100 dataset, as described below.

- ms-n_1-n_2-n_3: In every five images of the training set, $n_1\%$, $n_2\%$ and $n_3\%$ pixels are removed from the first three images, respectively.
- sp-n_1-n_2-n_3: In every five images of the training set, the first three images are corrupted by the *salt and pepper* noise with density $n_1\%$, $n_2\%$ and $n_3\%$, respectively.

5) Noise for YALE-B: Two types of noise are designed for the YALE-B dataset, as described below.

- ms-n_1-n_2-n_3: In every twenty images of the training set, $n_1\%$, $n_2\%$, $n_3\%$ pixels are removed from the first to fifth, the sixth to tenth, and the eleven to fifth images, respectively.
- sp-n_1-n_2-n_3: In every twenty images of the training set, the first to fifth, the sixth to tenth, and the eleven to fifth images are corrupted by the *salt and pepper* noise with density $n_1\%$, $n_2\%$ and $n_3\%$, respectively.

6) Noisy Image Generation: In summary, (a) on the COIL dataset, we generate 34 different noisy scenarios, (b) on the UMIST, YALE, COIL-100, YALE-B datasets, we generate 18 different noisy scenarios for each dataset.

These generated noisy data will be used in the subsequent experiments (please refer to Table II to Table VI for their specifications).

C. Implementation Details

1) Rank Estimation: Since our model is based on Tucker decomposition, we have to designate the size of the core tensor $\mathcal{G}$. In our case, if we know a priori that the data tensor is rank-$(r_1, r_2, \ldots, r_d, k)$, then the core tensor $\mathcal{G}$ shall be defined in $\mathbb{R}^{r_1 \times \cdots \times r_d \times k}$. This, however, is often not available in reality, and rank estimation has to be conducted. In this paper, we adopt the traditional method introduced in [8] to estimate the rank of the data tensor. In short, r_i is estimated by the number of dominant eigenvalues of $X_{(i)} X_{(i)}^T \in \mathbb{R}^{n_i \times n_i}$. This rank estimation algorithm is described in Algorithm 2, where $\mathrm{diag}(\cdot)$ is used to extract the diagonal elements of a square matrix to be a vector.

[1] See https://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php.
[2] See http://vision.ucsd.edu/content/yale-face-database.
[3] See https://see.xidian.edu.cn/vipsl/database_Face.html.
[4] See https://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php.
[5] See http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html.
[6] We remark there that our proposed scheme is not designed for handling outliers in data, and therefore we do not design experiments to test its anti-outlier functionality.

TABLE I: Statistics of the five real-world datasets

Dataset	COIL	YALE	UMIST	COIL-100	YALE-B
Samples	1,440	165	565	7,200	2,414
Classes	20	15	20	100	38
Image size	32×32	32×32	32×32	32×32	32×32
Training tensor size	$32 \times 32 \times 160$	$32 \times 32 \times 120$	$32 \times 32 \times 150$	$32 \times 32 \times 500$	$32 \times 32 \times 760$
Test tensor size	$32 \times 32 \times 1,280$	$32 \times 32 \times 45$	$32 \times 32 \times 415$	$32 \times 32 \times 6,700$	$32 \times 32 \times 1,654$

Algorithm 2: Rank estimation algorithm [8]

Input: A data tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d \times k}$.
Output: Estimated rank of $\mathcal{X}$.
for $i = 1, 2, \ldots, d$ **do**
  Compute $X_{(i)} X_{(i)}^T = U \Lambda U^T$ (eigen-decomposition);
  Compute λ as sorted $\mathrm{diag}(\Lambda)$ in noninceasing order;
  Compute r_i as the minimum r with $\frac{\sum_{j=1}^{r} \lambda_j}{\sum_{j=1}^{n_i} \lambda_j} \geq 98\%$;
return $\{r_1, r_2, \ldots, r_d\}$.

2) Performance Measurements: We measure the quality of the extracted features by their classification results. Here we use two classical classifiers, k-nearest neighbors (k-NN) [60] and multi-class support vector machine (SVM) [61], as well as an advanced classifier, ReLU neural network [62]. Their parameter settings are introduced as follows. We tune the parameters of the two classical classifiers by a grid-search strategy with a three-fold cross-validation. Then we tune k for the k-NN classifier in the search space for $k = 1, 2, \ldots, 10$ and tune c and γ for the SVM classifier in the same search space, i.e., $\{2^i\}_{i=-7}^{7} \times \{2^i\}_{i=-7}^{7}$. For the ReLU neural network, we set the number of hidden layer as 1, the dimension of the hidden layer as 100, the ℓ_2 regularization parameter as 0.0001, the learning rate as 0.001, the momentum parameter as 0.9, and we use the well-known LBFGS [63] method to train the neural network. The classification accuracy, defined as the fraction of the correct label predictions, is used as an indicator of the classification performance. The higher the classification accuracy, the better the feature extraction performance.

D. Competitor Choices

We remark here that for a fair comparison in the experiments, we mainly focus on comparing with basic competitors instead of the most cutting-edge ones (specific competitors will be introduced in due course). This is due to the following two reasons.

- Because our proposed ℓ_∞ model is only a backbone, we can simply use it to replace the one of any more advanced method having a similar basic architecture. Therefore, if our scheme has a superior performance over basic methods, then hopefully a similar improvement shall also be observed if our backbone is adopted in more advanced methods instead.
- Most recent more advanced methods involve regularizers, but ours do not, and therefore the fairness would be impaired if we forcibly compare with them. Besides, on the other hand, it is also not appropriate to remove their regularizers and then compare, and moreover, in many cases, if we do remove them, the resulting ablated methods would simply degenerate to the ℓ_1 or ℓ_2 model to be compared in the next subsection.

TABLE II: Comparisons on COIL with ℓ_1 and ℓ_2 models

Noise	k-NN ACC			SVM ACC			ReLU-NN ACC		
	ℓ_2	ℓ_1	ℓ_∞	ℓ_2	ℓ_1	ℓ_∞	ℓ_2	ℓ_1	ℓ_∞
original	**0.9208**	0.8906	0.8891	**0.9406**	0.9065	0.9198	**0.8938**	0.8482	0.8576
ms-5-10-20	0.8560	0.8560	**0.8810**	0.9234	0.9242	**0.9388**	0.8516	0.8591	**0.9128**
ms-10-10-10	0.8435	0.8240	**0.8990**	0.9234	0.8758	**0.9500**	0.8641	0.8349	**0.9141**
ms-10-20-30	0.8224	0.7852	**0.8406**	0.9036	0.9133	**0.9466**	0.8836	0.8714	**0.9070**
ms-10-30-50	0.8221	0.8013	**0.8721**	0.9031	0.9107	**0.9339**	0.8484	**0.8583**	**0.8583**
ms-10-50-90	0.8128	0.7888	**0.8664**	0.9133	0.9008	**0.9401**	0.8141	0.8182	**0.8471**
ms-20-20-20	0.8159	0.8073	**0.8698**	0.9112	0.9010	**0.9378**	0.8747	0.8315	**0.8961**
ms-30-30-30	0.7901	0.8180	**0.8768**	0.9174	0.9104	**0.9380**	0.8482	0.8284	**0.9083**
ms-40-40-40	0.8076	0.8193	**0.8964**	0.9190	0.9128	**0.9570**	0.8604	0.8656	**0.8927**
ms-50-50-50	0.8250	0.8206	**0.8680**	0.9216	0.9216	**0.9469**	0.8792	**0.8812**	0.8568
ms-5-10-15-20	0.8285	0.7830	**0.8969**	0.9041	0.8934	**0.9346**	0.8363	0.8816	**0.9131**
ms-10-10-10-10	0.8320	0.8391	**0.8482**	**0.9260**	0.9156	0.9115	0.8432	**0.8682**	0.8490
ms-10-20-30-40	0.7526	0.7049	**0.8534**	0.8945	0.8734	**0.9352**	0.8477	0.8495	**0.8625**
ms-10-30-50-70	0.7940	0.7776	**0.8555**	0.8729	0.9156	**0.9344**	**0.8490**	0.8383	0.8247
ms-20-20-20-20	0.7716	**0.8247**	0.8063	0.9078	0.9086	**0.9245**	0.8576	0.8680	**0.8948**
ms-30-30-30-30	0.7615	0.7880	**0.8010**	0.9125	0.9159	**0.9430**	0.8440	0.8654	**0.8805**
ms-40-40-40-40	0.7596	0.7802	**0.8526**	0.9201	0.9281	**0.9549**	0.8518	0.8448	**0.8724**
ms-50-50-50-50	0.7721	0.7974	**0.8104**	0.9190	0.9143	**0.9385**	0.8378	0.8513	**0.8542**
sp-5-10-20	0.8029	0.8109	**0.8839**	0.9115	0.9026	**0.9383**	0.8753	0.8135	**0.8891**
sp-10-10-10	0.7865	0.7492	**0.8664**	0.9065	0.8841	**0.9320**	0.8604	0.8435	**0.8812**
sp-10-20-30	0.7969	0.7414	**0.8766**	0.8922	0.7859	**0.9336**	0.8367	0.6690	**0.8784**
sp-10-30-50	0.7865	0.8318	**0.8846**	0.8992	0.8901	**0.9164**	**0.8646**	0.8398	0.8417
sp-10-50-90	0.8227	0.6411	**0.8701**	0.8938	0.7299	**0.9107**	0.8234	0.4911	**0.8513**
sp-20-20-20	0.7802	0.8255	**0.8646**	0.9057	0.8896	**0.9313**	0.7901	0.8005	**0.8182**
sp-30-30-30	0.7896	0.7422	**0.8789**	0.8932	0.8401	**0.9089**	0.8177	0.7310	**0.8419**
sp-40-40-40	0.8013	0.8253	**0.8685**	0.8948	0.8690	**0.9242**	0.7818	0.8180	**0.8318**
sp-50-50-50	0.8216	0.7901	**0.8508**	0.8776	**0.8961**	0.8779	0.8063	0.7990	**0.8453**
sp-5-10-15-20	0.7701	0.7482	**0.8453**	0.8849	0.8388	**0.9263**	0.8414	0.7276	**0.8443**
sp-10-10-10-10	0.7464	0.7878	**0.8701**	0.8781	0.8867	**0.9286**	0.7875	0.8339	**0.8513**
sp-10-20-30-40	0.7630	0.6997	**0.8536**	0.8576	0.7443	**0.9065**	**0.8214**	0.6677	0.8180
sp-10-30-50-70	0.7828	0.8180	**0.8464**	0.8888	0.8604	**0.8992**	0.8034	0.7802	**0.8409**
sp-20-20-20-20	0.7456	0.7990	**0.8268**	0.8755	0.8724	**0.8823**	0.8294	0.8409	**0.8417**
sp-30-30-30-30	0.7609	0.6701	**0.8406**	0.8661	0.6927	**0.8688**	**0.8284**	0.5531	0.8120
sp-40-40-40-40	0.7604	0.7865	**0.8372**	0.8534	0.8581	**0.8747**	0.8076	0.8172	**0.8208**
sp-50-50-50-50	0.7576	0.5669	**0.8052**	**0.8612**	0.6635	0.8414	0.7674	0.4971	**0.8161**

TABLE III: Comparisons on YALE with ℓ_1 and ℓ_2 models

Noise	k-NN ACC			SVM ACC			ReLU-NN ACC		
	ℓ_2	ℓ_1	ℓ_∞	ℓ_2	ℓ_1	ℓ_∞	ℓ_2	ℓ_1	ℓ_∞
original	**0.6370**	0.6222	0.5556	0.7407	**0.7556**	0.7407	**0.7259**	0.6815	0.6667
ms-5-10-20	0.3481	0.5259	**0.6000**	0.7852	0.7556	**0.8148**	**0.6667**	0.6519	0.6444
ms-10-10-10	0.4074	**0.4667**	0.4074	0.7407	**0.7704**	**0.7704**	0.6519	**0.7037**	0.6593
ms-10-20-30	0.3852	0.4815	**0.4963**	0.7556	0.7630	**0.7852**	0.6148	0.6370	**0.6593**
ms-10-30-50	0.4444	0.4889	**0.5259**	0.7704	0.7630	**0.8222**	0.6296	**0.6815**	0.6593
ms-10-50-90	0.3852	**0.5333**	0.4963	0.7333	0.7407	**0.7704**	0.6519	**0.6815**	0.6741
ms-20-20-20	0.2963	0.4296	**0.4593**	0.7778	0.7704	**0.7926**	0.6370	**0.7333**	0.6741
ms-30-30-30	0.4000	**0.4963**	0.4444	0.7778	**0.8296**	0.7778	**0.6741**	0.6593	0.6519
ms-40-40-40	0.4148	**0.5111**	0.4519	**0.8148**	0.8000	0.7111	**0.6370**	**0.6370**	**0.6370**
ms-50-50-50	**0.4370**	**0.4370**	0.3852	0.7259	0.7333	**0.7407**	0.4889	0.4370	**0.5556**
sp-5-10-20	0.3333	0.4519	**0.6296**	0.7111	0.7778	**0.8370**	0.6741	0.6000	**0.7185**
sp-10-10-10	0.3259	**0.4519**	0.4148	0.7407	0.7704	**0.7926**	0.6593	**0.7185**	0.6815
sp-10-20-30	0.4444	**0.4889**	0.4074	0.7111	0.8222	**0.8296**	0.6667	**0.7630**	0.6815
sp-10-30-50	0.4000	0.5185	**0.5333**	0.6519	0.7481	**0.8296**	0.6741	**0.6815**	0.6741
sp-10-50-90	0.4074	**0.5852**	0.4593	0.6519	**0.7704**	0.7481	0.5556	**0.5852**	0.5185
sp-20-20-20	0.3407	**0.5407**	0.4963	0.6370	**0.7556**	0.7333	0.5481	0.6593	**0.7259**
sp-30-30-30	0.4444	0.5259	**0.5704**	0.5630	**0.6593**	**0.6593**	0.6074	**0.6519**	0.6148
sp-40-40-40	0.4074	**0.5926**	0.5704	0.5333	0.5333	**0.5778**	0.5259	**0.7111**	0.6519
sp-50-50-50	0.3778	0.4296	**0.5185**	0.4889	0.5037	**0.5556**	0.5556	0.6074	**0.6222**

E. Experiment: Comparisons with ℓ_1 and ℓ_2 Models

In this part, we evaluate the performance of our proposed ℓ_∞ model by image classification and face recognition tasks under different corruption degrees, in comparison with ℓ_1 and ℓ_2 models. Specifically, we apply the ℓ_∞, ℓ_2 and ℓ_1 models to extract features from all corrupted datasets mentioned in Section VI-B6, following the workflow introduced in Section III-C. Then, we evaluate the quality of the extracted features in terms of classification accuracy. To alleviate the stochastic effects of random initialization, we run each model three times and report the averaged results. Experimental results are shown in Table II to Table VI, where values in bold represent the highest. From these tables, we have the following findings.

- **Our ℓ_∞ model is effective.** As observed, in more than

TABLE IV: Comparisons on UMIST with ℓ_1 and ℓ_2 models

Noise	k-NN ACC			SVM ACC			ReLU-NN ACC		
	ℓ_2	ℓ_1	ℓ_∞	ℓ_2	ℓ_1	ℓ_∞	ℓ_2	ℓ_1	ℓ_∞
original	0.8281	0.8466	**0.8546**	0.8755	0.9133	**0.9181**	0.8024	0.8209	**0.8683**
ms-5-10-20	0.7912	0.8193	**0.8482**	0.8554	0.8851	**0.8900**	0.7671	**0.8129**	0.8000
ms-10-10-10	0.7727	0.8104	**0.8217**	0.8586	0.9036	**0.9341**	0.8112	0.7639	**0.8506**
ms-10-20-30	0.7534	0.7590	**0.7928**	0.8378	0.8635	**0.9253**	0.7655	0.7888	**0.8426**
ms-10-30-50	0.7823	0.7454	**0.7888**	0.8281	0.8643	**0.8980**	0.7502	0.7783	**0.7815**
ms-10-50-90	0.7028	0.7293	**0.8008**	0.8217	0.8699	**0.8803**	0.6699	0.7261	**0.7494**
ms-20-20-20	0.7365	0.7486	**0.8080**	0.8514	0.8787	**0.9341**	0.7446	**0.8177**	0.8161
ms-30-30-30	0.7157	0.7028	**0.8000**	0.8434	0.8795	**0.9036**	0.7213	0.7157	**0.7944**
ms-40-40-40	0.7261	0.7269	**0.8080**	0.8538	0.8699	**0.9124**	0.7566	0.7357	**0.8064**
ms-50-50-50	0.7277	0.7084	**0.8048**	0.8120	0.8265	**0.8675**	0.6353	0.6570	**0.7092**
sp-5-10-20	0.7446	0.7293	**0.8313**	0.8643	0.8795	**0.9357**	0.7510	0.8120	**0.8313**
sp-10-10-10	0.6867	0.6876	**0.7823**	0.7920	0.8072	**0.8787**	0.7920	0.7743	**0.8032**
sp-10-20-30	0.6546	0.6506	**0.7679**	0.7550	0.7719	**0.8771**	0.6940	0.7133	**0.7687**
sp-10-30-50	0.6900	0.6739	**0.8209**	0.8297	0.8233	**0.9221**	0.7173	0.7221	**0.7992**
sp-10-50-90	0.7261	0.7044	**0.8281**	0.7622	0.8000	**0.8795**	0.7398	0.7470	**0.8104**
sp-20-20-20	0.6554	0.6731	**0.7944**	0.7815	0.7960	**0.8916**	0.6490	0.6747	**0.8080**
sp-30-30-30	0.6908	0.6803	**0.7855**	0.7871	0.7888	**0.9076**	0.6972	0.7373	**0.8201**
sp-40-40-40	0.7454	0.7550	**0.8225**	0.8129	0.8498	**0.9020**	0.6916	0.7695	**0.8241**
sp-50-50-50	0.7606	0.6659	**0.8586**	0.8514	0.6514	**0.9293**	0.7221	0.5197	**0.8305**

TABLE V: Comparisons on COIL-100 with ℓ_1 and ℓ_2 models

Noise	k-NN ACC			SVM ACC			ReLU-NN ACC		
	ℓ_2	ℓ_1	ℓ_∞	ℓ_2	ℓ_1	ℓ_∞	ℓ_2	ℓ_1	ℓ_∞
original	0.4299	0.4371	**0.4488**	0.4659	0.4815	**0.4864**	0.4029	**0.4086**	0.4081
ms-5-10-20	0.3775	0.4012	**0.4436**	0.4385	0.4724	**0.4855**	0.3531	**0.4055**	0.3896
ms-10-10-10	0.3427	0.3124	**0.3546**	0.4215	0.4719	**0.4848**	0.3076	0.3340	**0.3660**
ms-10-20-30	0.3701	0.4224	**0.4325**	0.4370	**0.4930**	0.4896	0.3945	**0.4082**	0.3999
ms-10-30-50	0.3604	0.4139	**0.4382**	0.4433	0.4909	**0.4979**	0.3927	0.4245	**0.4567**
ms-10-50-90	0.3733	0.4072	**0.4384**	0.4606	0.4931	**0.5049**	0.3764	0.3722	**0.3993**
ms-20-20-20	0.3687	**0.4140**	0.3385	0.4436	**0.4903**	0.4881	0.3325	**0.3719**	0.3222
ms-30-30-30	0.3873	0.3294	**0.4110**	0.4527	**0.4885**	0.4769	0.2846	**0.3755**	0.3294
ms-40-40-40	0.3845	0.4160	**0.4231**	0.4533	**0.4903**	0.4815	0.2685	**0.3740**	0.3399
ms-50-50-50	0.3933	0.4285	**0.4387**	0.4818	**0.5031**	0.4930	**0.4049**	0.3790	0.3818
sp-5-10-20	0.3858	0.4450	**0.4741**	0.4405	0.4789	**0.4995**	0.3342	0.3960	**0.4223**
sp-10-10-10	0.3940	0.4188	**0.4450**	0.4383	0.4554	**0.4691**	0.3701	0.3314	**0.4047**
sp-10-20-30	0.4302	0.4561	**0.4822**	0.4666	0.4753	**0.4992**	**0.3859**	0.3618	0.3787
sp-10-30-50	0.4467	0.4630	**0.4822**	0.4777	0.4822	**0.4943**	0.3850	**0.4270**	0.3944
sp-10-50-90	0.4428	0.4661	**0.4889**	0.4736	0.4818	**0.5019**	**0.3884**	0.3666	0.3687
sp-20-20-20	0.4138	0.4507	**0.4651**	0.4608	0.4676	**0.4825**	**0.3853**	0.3635	0.3701
sp-30-30-30	0.4418	0.4553	**0.4776**	0.4620	0.4678	**0.4863**	0.3740	0.3682	**0.3904**
sp-40-40-40	0.4557	0.4723	**0.4836**	0.4786	0.4779	**0.4929**	0.3817	**0.3957**	0.3748
sp-50-50-50	0.4548	0.4781	**0.4846**	0.4705	0.4847	**0.4937**	0.3803	**0.3882**	0.3701

TABLE VI: Comparisons on YALE-B with ℓ_1 and ℓ_2 models

Noise	k-NN ACC			SVM ACC			ReLU-NN ACC		
	ℓ_2	ℓ_1	ℓ_∞	ℓ_2	ℓ_1	ℓ_∞	ℓ_2	ℓ_1	ℓ_∞
original	0.4081	**0.4196**	0.4069	**0.4359**	0.3881	0.4268	0.2993	0.2963	**0.3174**
ms-5-10-20	0.3759	**0.4127**	0.3881	0.4272	**0.4538**	0.3950	**0.3140**	0.3041	0.2386
ms-10-10-10	0.3235	**0.4131**	0.3174	0.3916	**0.4254**	0.3797	**0.2481**	0.2465	0.2320
ms-10-20-30	0.2997	0.3668	**0.3738**	0.3934	**0.4393**	0.3924	**0.2584**	0.1945	0.2191
ms-10-30-50	0.2898	0.3396	**0.3640**	0.3738	**0.4045**	0.4037	**0.2797**	0.2735	0.2166
ms-10-50-90	0.3130	**0.3773**	0.3688	0.3329	**0.3414**	0.2723	**0.2064**	0.2033	0.1812
ms-20-20-20	0.2031	**0.3345**	0.2846	0.3305	**0.4188**	0.3861	**0.2181**	0.2011	0.1834
ms-30-30-30	0.1636	**0.2680**	0.2287	0.3180	**0.3827**	0.3537	**0.1915**	0.1528	0.1896
ms-40-40-40	0.1467	**0.1943**	0.1784	0.2868	0.3497	**0.3601**	0.1675	**0.1792**	0.1467
ms-50-50-50	0.1590	**0.1836**	0.1753	0.2868	**0.3428**	0.3374	0.0983	**0.1173**	0.1135
sp-5-10-20	0.3283	0.3593	**0.3748**	0.3730	0.4381	**0.4442**	**0.3353**	0.2733	0.2646
sp-10-10-10	0.2499	**0.3722**	0.3410	0.4041	**0.4438**	0.4196	**0.3412**	0.3200	0.3158
sp-10-20-30	0.2459	**0.3577**	0.3484	0.3615	**0.4146**	0.3992	**0.3085**	0.2753	0.2541
sp-10-30-50	0.2755	**0.3527**	0.3394	0.3668	0.3974	**0.3976**	**0.2576**	0.2235	0.2221
sp-10-50-90	0.2765	**0.3430**	0.3275	0.3547	0.3827	**0.3835**	**0.2753**	0.1832	0.2320
sp-20-20-20	0.1749	**0.2747**	0.2640	0.3517	**0.3805**	0.3783	**0.3174**	0.2306	0.2257
sp-30-30-30	0.1824	0.1937	**0.2011**	0.3247	**0.3601**	0.3537	0.2537	**0.2547**	0.1927
sp-40-40-40	**0.2215**	0.1528	0.1733	0.3370	0.3380	**0.3418**	**0.2090**	0.1493	0.1640
sp-50-50-50	**0.1971**	0.1572	0.1719	0.3021	0.3118	**0.3281**	**0.2241**	0.2076	0.2005

half of cases, the performance of the proposed ℓ_∞ model shows substantial improvement over both the ℓ_2 and ℓ_1 models. For instance, on the COIL (sp-10-20-30-40) dataset in terms of k-NN ACC, the ℓ_∞ model shows over 10% and 20% improvement over the ℓ_2 and ℓ_1 models, respectively. Besides, although the ℓ_∞ model shows inferior performance than the ℓ_1 model in several cases on the YALE and YALE-B datasets, its overall performance is still good, and is much better than the ℓ_2 model in many cases. These evidences have fully illustrated the effectiveness of the proposed ℓ_∞ model.

- **Our ℓ_∞ model best suits for noisy scenarios.** As observed, on the five original datasets (without corruption), the proposed ℓ_∞ model performs not well sometimes. This is because the ℓ_∞ model aims at optimizing the maximum fitting error, which is an inappropriate choice in noise-free scenarios. When the data contains uncertainty, the ℓ_∞ model should be a good choice.
- **There is no free lunch** [64]. As observed, although our proposed ℓ_∞ performs excellently on the COIL, UMIST and COIL-100 datasets, its performance is not as good as expected on the YALE and YALE-B datasets, where the ℓ_1 model performs relatively well. We think this is partly due to the fact that the YALE and YALE-B datasets only contain the frontal faces of the individuals under different conditions, but the others cover a series rotations/poses of the objects/individuals, and thus the importance distributions of different samples on these two datasets become relatively more even, and hence the ℓ_1 model may intuitively be more competitive. This shows that different models may have their own preferred datasets.

F. Experiment: Comparisons with Classical and SOTA Methods

To comprehensively verify the effectiveness of the ℓ_∞ method, we further compare it with six classical feature extraction methods, namely probabilistic PCA (ProbPCA) [65], factor analysis (FA) [66], isometric mapping (IsoMap) [67], locally linear embedding (LLE) [68], Laplacian eigenmaps (LapE) [26] and autoencoder (AE) [69], as well as three SOTA feature extraction methods, namely UMAP[7] [70], TriMAP[8] [71] and PaCMAP[9] [72]. For the six classical methods, we directly use their implementations provided by the MATLAB Dimensionality Reduction Toolbox [73], while for the three SOTA methods, we use their open-sourced implementations. Since the ℓ_∞ model is dedicated for handling data uncertainty, all the experiments here are conducted on the five corrupted datasets: COIL (sp-10-30-50), YALE (sp-10-30-50), UMIST (sp-10-30-50), COIL-100 (sp-10-30-50) and YALE-B (sp-10-30-50). Since the nine comparative methods are matrix-based, we first vectorize the images in the five datasets and arrange the resultants in order to form a data matrix, and then use Algorithm 2 again to estimate the rank of the data matrix in the implementation. For IsoMap, LLE and LapE, we set $k = 20$ since they adopt a k-NN graph. Moreover, we use the Nyström approximation [74] to extract features from the test data as these three methods do not support exact out-of-sample mapping. For AE, we set the layer size as $f_0 \to \lceil 1.2 f_0 \rceil + 5 \to \lceil f_0/4 \rceil + 3 \to \lceil f_0/10 \rceil \to f$ where f_0 denotes the number of original features (i.e., $32 \times 32 = 1{,}024$ for all datasets) and $\lceil \cdot \rceil$ is the ceiling function. For the three SOTA methods, we use their default parameter settings as their numbers of parameters are too large to be tuned. The evaluating procedures are the same as the previous experiment.

We list the experimental results in Table VII where figures in bold represent the highest and figures in shade are the top-two. As observed, in terms of both k-NN ACC, SVM ACC and ReLU-NN ACC, the ℓ_∞ method consistently achieves the top-two performance on these five corrupted datasets. Besides, for 10 out of 15 cases, our ℓ_∞ model performs the best over all competitors. This fully illustrates the effectiveness of the ℓ_∞ model which is mainly attributed to its consideration of maintaining the worst-case model performance.

[7]Open sourced at https://github.com/lmcinnes/umap.
[8]Open sourced at https://github.com/eamid/trimap.
[9]Open sourced at https://github.com/YingfanWang/PaCMAP.

TABLE VII: Comparisons with nine classical and SOTA methods

Model	k-NN ACC					SVM ACC					ReLU-NN ACC				
	COIL	YALE	UMIST	COIL-100	YALE-B	COIL	YALE	UMIST	COIL-100	YALE-B	COIL	YALE	UMIST	COIL-100	YALE-B
ProbPCA	0.8318	0.4667	0.7012	0.1955	0.2135	0.8648	0.5481	0.7671	0.1844	0.2062	0.8164	0.5778	0.6434	0.3085	0.0387
FA	0.7102	0.4889	0.5631	0.2087	0.2292	0.7247	0.5852	0.6892	0.2175	0.2360	0.7789	0.4889	0.6602	0.3204	0.0472
IsoMap	0.7354	0.4667	0.7325	0.1646	0.1890	0.7578	0.6000	0.7759	0.1644	0.1889	0.7258	0.4222	0.7253	0.3194	0.0526
LLE	0.7130	0.4000	0.7301	0.1769	0.2163	0.8148	0.4000	0.7205	0.1764	0.2139	0.7391	0.5111	0.6651	0.2585	0.0629
LapE	0.2599	0.0741	0.2313	0.0114	0.0276	0.3734	0.1333	0.2048	0.0246	0.0442	0.3398	0.2444	0.1855	0.0385	0.0441
AE	0.3615	0.4296	0.5888	0.2162	0.2366	0.6117	0.5333	0.6867	0.1932	0.2174	0.7875	0.1333	0.6313	0.2610	0.0381
UMAP	0.6969	0.4148	0.5614	0.2880	0.0907	0.7049	0.4444	0.6000	0.2799	0.1133	0.5182	0.3704	0.4418	0.0566	0.0437
TriMAP	0.8466	0.3704	0.7365	**0.6836**	0.0977	0.8672	0.3704	0.7542	**0.6417**	0.0854	**0.8609**	0.3259	**0.8008**	**0.5941**	0.0792
PaCMAP	0.4883	0.1481	0.3815	0.2650	0.0707	0.4987	0.2000	0.3791	0.2677	0.0879	0.4721	0.1556	0.3767	0.2174	0.0792
ℓ_∞	**0.8846**	**0.5333**	**0.8209**	0.4822	**0.3394**	**0.9164**	**0.8296**	**0.9221**	0.4943	**0.3976**	0.8417	**0.6741**	0.7992	0.3944	**0.2221**

(1) ℓ_2 model (2) ℓ_1 model (3) ℓ_∞ model

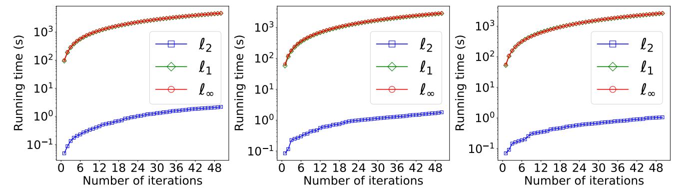
Fig. 2: Visualization of the extracted features by the three models on the UMIST (sp-10-30-50) dataset.

G. Experiment: Visualization of the Extracted Features

In this part, we perform a visualization experiment to qualitatively compare the ℓ_2, ℓ_1 and ℓ_∞ models and deliver some intuitive insights. Specifically, we apply the well-known t-distributed stochastic neighbor embedding (t-SNE) [75] to embed the extracted features by these three models from the test set of the UMIST (sp-10-30-50) dataset into $\mathbb{R}^2$ and then plot the distribution of the dimension-reduced features in this low-dimensional space. The experimental results are illustrated in Figure 2 where different colors represent different classes. We can clearly observe from the figures that the features extracted by the ℓ_∞ model show a better separability and features within the same class stay close for almost every class, while the features extracted by the other two models admit some entanglements and features within some classes can spread out. This phenomenon intuitively demonstrates the superiority of the features extracted by the ℓ_∞ model over the other two and explains the reason of the better performance, which is mainly attributed to the robustness considered in the ℓ_∞ model.

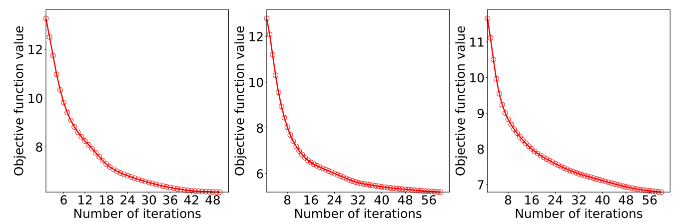
H. Experiments: Running Time and Convergence Analyses

In this part, we first analyze the running time of the proposed ℓ_∞ model. Specifically, we run the ℓ_2, ℓ_1 and ℓ_∞ models on the COIL (sp-10-30-50), YALE (sp-10-30-50) and UMIST (sp-10-30-50) datasets, and record their accumulated time consumption within 50 iterations, reported in Figure 3. Similar running time trends can be observed on other corruption conditions, therefore we omit them here for the interest of space. As observed, the running time trends of the ℓ_2, ℓ_1 and ℓ_∞ models are very similar across the three datasets. Besides, for each of the three datasets, although the ℓ_∞ model is much slower than the ℓ_2 model, it costs nearly the same time as the ℓ_1 model, which shows its efficiency to some extent. Furthermore, as previous experimental results convey, the ℓ_∞ model shows great performance improvement over the ℓ_2 and ℓ_1 models. Therefore, the ℓ_∞ model strikes a good balance between effectiveness and efficiency.



(1) COIL (sp-10-30-50) (2) YALE (sp-10-30-50) (3) UMIST (sp-10-30-50)

Fig. 3: Running time analysis on the COIL (sp-10-30-50), YALE (sp-10-30-50) and UMIST (sp-10-30-50) datasets.



(1) COIL (sp-10-30-50) (2) YALE (sp-10-30-50) (3) UMIST (sp-10-30-50)

Fig. 4: Convergence analysis on the COIL (sp-10-30-50), YALE (sp-10-30-50) and UMIST (sp-10-30-50) datasets.

In Section V-B, we proved the convergence of Algorithm 1 theoretically. We now empirically study the convergence behavior of Algorithm 1. Specifically, we run the ℓ_∞ model on the COIL (sp-10-30-50), YALE (sp-10-30-50) and UMIST (sp-10-30-50) datasets again, and record its objective function value in each iteration on each dataset. The curves for objective function values are shown in Figure 4. Because similar convergence trends can be observed on other corruption conditions, we omit them here again for the interest of space. As observed, on each of the three datasets, the objective function value decreases monotonically, which validates our convergence theory. Besides, although the computational complexity for each iteration of Algorithm 1 is relatively high (see Theorem V.2), the algorithm converges very quickly in practice. As observed, Algorithm 1 can converge within only dozens of iterations on each of the three datasets, showing its efficiency to some extent. This fast convergence of Algorithm 1 guarantees the efficiency of the whole feature extraction process.

VII. Conclusion

In this paper, we proposed a novel robust nonnegative Tucker decomposition model, the ℓ_∞ model, for feature extraction from uncertain data. Inspired by the idea of maintaining the worst-case model performance from robust optimization, the ℓ_∞ model aims to minimize the maximum fitting error to tackle the data uncertainty. To solve the proposed model, we developed an effective algorithm based on alternating update with theoretically guaranteed convergence. We performed extensive experiments on five real-world benchmark datasets under a variety of noisy conditions. The experimental results showed excellent performance of the ℓ_∞ model compared to many others. In the future, we plan to investigate how to incorporate prior data knowledge into the ℓ_∞ model.

References

[1] Y. Chen, T.-Z. Huang, and X.-L. Zhao, "Destriping of multispectral remote sensing image using low-rank tensor decomposition," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 12, pp. 4950–4967, 2018.

[2] G. Jiang, S. Liu, M. Yu, F. Shao, Z. Peng, and F. Chen, "No reference stereo video quality assessment based on motion feature in tensor decomposition domain," *Journal of Visual Communication and Image Representation*, vol. 50, pp. 247–262, 2018.

[3] S. Fernandes, H. Fanaee-T, and J. Gama, "Tensor decomposition for analysing time-evolving social networks: An overview," *Artificial Intelligence Review*, vol. 54, no. 4, pp. 2891–2916, 2021.

[4] F. Cong, Q.-H. Lin, L.-D. Kuang, X.-F. Gong, P. Astikainen, and T. Ristaniemi, "Tensor decomposition of EEG signals: A brief review," *Journal of Neuroscience Methods*, vol. 248, pp. 59–69, 2015.

[5] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[6] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "MPCA: Multilinear principal component analysis of tensor objects," *IEEE Transactions on Neural Networks*, vol. 19, no. 1, pp. 18–39, 2008.

[7] S. Yan, D. Xu, Q. Yang, L. Zhang, X. Tang, and H.-J. Zhang, "Discriminant analysis with tensor representation," in *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 526–532.

[8] A. H. Phan and A. Cichocki, "Tensor decompositions for feature extraction and classification of high dimensional datasets," *Nonlinear Theory and Its Applications*, vol. 1, no. 1, pp. 37–68, 2010.

[9] Y. Fu, Q. Ruan, Z. Luo, Y. Jin, G. An, and J. Wan, "FERLrTc: 2D+3D facial expression recognition via low-rank tensor completion," *Signal Processing*, vol. 161, pp. 74–88, 2019.

[10] J. Zhang, X. Li, P. Jing, J. Liu, and Y. Su, "Low-rank regularized heterogeneous tensor decomposition for subspace clustering," *IEEE Signal Processing Letters*, vol. 25, no. 3, pp. 333–337, 2017.

[11] Z. Huang, S. Li, L. Fang, H. Li, and J. A. Benediktsson, "Hyperspectral image denoising with group sparse and low-rank tensor decomposition," *IEEE Access*, vol. 6, pp. 1380–1390, 2017.

[12] B. Jiang, C. Ding, J. Tang, and B. Luo, "Image representation and learning with graph-Laplacian Tucker tensor decomposition," *IEEE Transactions on Cybernetics*, vol. 49, no. 4, pp. 1417–1426, 2018.

[13] G. An, S. Liu, and Q. Ruan, "A sparse neighborhood preserving non-negative tensor factorization algorithm for facial expression recognition," *Pattern Analysis and Applications*, vol. 20, no. 2, pp. 453–471, 2017.

[14] X. Li, M. K. Ng, G. Cong, Y. Ye, and Q. Wu, "MR-NTD: Manifold regularization nonnegative Tucker decomposition for tensor data dimension reduction and representation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 8, pp. 1787–1800, 2016.

[15] W. Yin and Z. Ma, "LE & LLE regularized nonnegative Tucker decomposition for clustering of high dimensional datasets," *Neurocomputing*, vol. 364, pp. 77–94, 2019.

[16] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust optimization*. New Jersey, the U.K.: Princeton university press, 2009.

[17] X. Cao, X. Wei, Y. Han, and D. Lin, "Robust face clustering via tensor decomposition," *IEEE Transactions on Cybernetics*, vol. 45, no. 11, pp. 2546–2557, 2014.

[18] Y. Pang, X. Li, and Y. Yuan, "Robust tensor analysis with ℓ_1-norm," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 2, pp. 172–178, 2009.

[19] P. P. Markopoulos, D. G. Chachlakis, and E. E. Papalexakis, "The exact solution to rank-1 ℓ_1-norm Tucker2 decomposition," *IEEE Signal Processing Letters*, vol. 25, no. 4, pp. 511–515, 2018.

[20] D. G. Chachlakis, A. Prater-Bennette, and P. P. Markopoulos, "ℓ_1-norm Tucker tensor decomposition," *IEEE Access*, vol. 7, pp. 178 454–178 465, 2019.

[21] D. Chachlakis, A. Prater-Bennette, and P. Markopoulos, "ℓ_1-norm higher-order orthogonal iterations for robust tensor analysis," in *Proceedings of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 4826–4830.

[22] P. P. Markopoulos, D. G. Chachlakis, and A. Prater-Bennette, "ℓ_1-norm higher-order singular-value decomposition," in *Proceedings of the 2018 IEEE Global Conference on Signal and Information Processing*, 2018, pp. 1353–1357.

[23] A. Ben-Tal and A. Nemirovski, "Robust optimization—methodology and applications," *Mathematical Programming*, vol. 92, no. 3, pp. 453–480, 2002.

[24] M. J. Idaji, M. B. Shamsollahi, and S. H. Sardouie, "Higher order spectral regression discriminant analysis (HOSRDA): A tensor feature reduction method for ERP detection," *Pattern Recognition*, vol. 70, pp. 152–162, 2017.

[25] A. Jukić and M. Filipović, "Supervised feature extraction for tensor objects based on maximization of mutual information," *Pattern Recognition Letters*, vol. 34, no. 13, pp. 1476–1484, 2013.

[26] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[27] Q. Shi, Y.-M. Cheung, Q. Zhao, and H. Lu, "Feature extraction for incomplete data via low-rank tensor decomposition with feature regularization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 6, pp. 1803–1817, 2018.

[28] B. Sun, J. Li, M. Shao, and Y. Fu, "LRPRNet: Lightweight deep network by low-rank pointwise residual convolution," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[29] B. Zhou, F. Zhang, and L. Peng, "Compact representation for dynamic texture video coding using tensor method," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 2, pp. 280–288, 2012.

[30] M. R. Khokher, A. Bouzerdoum, and S. L. Phung, "A super descriptor tensor decomposition for dynamic scene recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 4, pp. 1063–1076, 2018.

[31] Y. Liu, Z. Long, H. Huang, and C. Zhu, "Low CP rank and Tucker rank tensor completion for estimating missing components in image data," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 4, pp. 944–954, 2019.

[32] Y. Xu, Z. Wu, J. Chanussot, and Z. Wei, "Hyperspectral computational imaging via collaborative Tucker3 tensor decomposition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 1, pp. 98–111, 2020.

[33] Y. He and G. K. Atia, "Patch tracking-based streaming tensor ring completion for visual data recovery," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 12, pp. 8312–8326, 2022.

[34] J. Tang, X. Shu, G.-J. Qi, Z. Li, M. Wang, S. Yan, and R. Jain, "Tri-clustered tensor completion for social-aware image tag refinement," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 8, pp. 1662–1674, 2016.

[35] J. Tang, X. Shu, Z. Li, Y.-G. Jiang, and Q. Tian, "Social anchor-unit graph regularized tensor completion for large-scale image retagging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 2027–2034, 2019.

[36] V. Lebedev, Y. Ganin, M. Rakhuba, I. V. Oseledets, and V. S. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," in *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[37] M. Zhang and C. Ding, "Robust Tucker tensor decomposition for effective image representation," in *Proceedings of the 2013 IEEE International Conference on Computer Vision*, 2013, pp. 2448–2455.

[38] T. Wu, "Online tensor low-rank representation for streaming data clustering," *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

[39] M. E. Kilmer, K. Braman, N. Hao, and R. C. Hoover, "Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 1, pp. 148–172, 2013.

[40] 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 2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5249–5257.

[41] P. Zhou and J. Feng, "Outlier-robust tensor PCA," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2263–2271.
[42] C. Jia, Z. Ding, Y. Kong, and Y. Fu, "Semi-supervised cross-modality action recognition by latent tensor transfer learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 9, pp. 2801–2814, 2019.
[43] Y. Chen, X. Xiao, C. Peng, G. Lu, and Y. Zhou, "Low-rank tensor graph learning for multi-view subspace clustering," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 1, pp. 92–104, 2021.
[44] Y. Jia, H. Liu, J. Hou, S. Kwong, and Q. Zhang, "Multi-view spectral clustering tailored tensor low-rank representation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 12, pp. 4784–4797, 2021.
[45] Y. Jia, G. Lu, H. Liu, and J. Hou, "Semi-supervised subspace clustering via tensor low-rank representation," *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.
[46] Q. Liu, X. Li, H. Cao, and Y. Wu, "From simulated to visual data: A robust low-rank tensor completion approach using ℓ_p-regression for outlier resistance," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 6, pp. 3462–3474, 2021.
[47] B. Zhang, T. Cai, Z. Lu, D. He, and L. Wang, "Towards certifying ℓ_∞ robustness using neural networks with ℓ_∞-dist neurons," in *Proceedings of the 2021 International Conference on Machine Learning*, 2021, pp. 12 368–12 379.
[48] B. Zhang, D. Jiang, D. He, and L. Wang, "Boosting the certified robustness of ℓ_∞ distance nets," *arXiv preprint arXiv:2110.06850*, 2021.
[49] ——, "Rethinking lipschitz neural networks for certified ℓ_∞ robustness," *arXiv preprint arXiv:2210.01787*, 2022.
[50] C. Qin, J. Martens, S. Gowal, D. Krishnan, K. Dvijotham, A. Fawzi, S. De, R. Stanforth, and P. Kohli, "Adversarial robustness through local linearization," *Proceedings of the 2019 International Conference on Neural Information Processing Systems*, vol. 32, 2019.
[51] B. A. Francis, *A course in H_∞ control theory*. Springer, 1987.
[52] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.
[53] Y.-D. Kim and S. Choi, "Nonnegative Tucker decomposition," in *Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.
[54] A. H. Phan and A. Cichocki, "Tensor decompositions for feature extraction and classification of high dimensional datasets," *Nonlinear Theory and Its Applications*, vol. 1, no. 1, pp. 37–68, 2010.
[55] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019. [Online]. Available: http://docs.mosek.com/9.0/toolbox/index.html
[56] D. D. Lee and H. S. Seung, "Algorithms for nonnegative matrix factorization," in *Proceedings of the 2000 International Conference on Neural Information Processing Systems*, 2000, pp. 535–541.
[57] E. D. Andersen, C. Roos, and T. Terlaky, "On implementing a primal-dual interior-point method for conic quadratic optimization," *Mathematical Programming*, vol. 95, no. 2, pp. 249–277, 2003.
[58] E. D. Andersen, "Complexity of solving conic quadratic problems," 2013. [Online]. Available: https://erlingdandersen.blogspot.com/2013/11/complexity-of-solving-conic-quadratic.html
[59] B. W. Bader and T. G. Kolda, "MATLAB tensor toolbox version 2.6," 2015. [Online]. Available: http://www.sandia.gov/~tgkolda/TensorToolbox/
[60] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. New Yord, the U.S.: John Wiley & Sons, 2006.
[61] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, 2011.
[62] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
[63] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.
[64] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.
[65] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B*, vol. 61, no. 3, pp. 611–622, 1999.
[66] D. Child, *The essentials of factor analysis*. New York, the U.S.: Cassell Educational, 1990.
[67] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
[68] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
[69] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. New York, the U.S.: MIT press Cambridge, 2016.
[70] L. McInnes, J. Healy, N. Saul, and L. Großberger, "UMAP: Uniform manifold approximation and projection," *Journal of Open Source Software*, vol. 3, no. 29, 2018.
[71] E. Amid and M. K. Warmuth, "TriMap: Large-scale dimensionality reduction using triplets," *arXiv preprint arXiv:1910.00204*, 2019.
[72] Y. Wang, H. Huang, C. Rudin, and Y. Shaposhnik, "Understanding how dimension reduction tools work: An empirical approach to deciphering t-SNE, UMAP, TriMAP, and PaCMAP for data visualization," *Journal of Machine Learning Research*, vol. 22, no. 1, pp. 9129–9201, 2021.
[73] L. Van der Maaten, E. O. Postma, and H. J. van den Herik, "MATLAB toolbox for dimensionality reduction," MICC, Maastricht University, Tech. Rep., 2007.
[74] A. Farahat, A. Ghodsi, and M. Kamel, "A novel greedy algorithm for Nyström approximation," in *Proceedings of the 2011 International Conference on Artificial Intelligence and Statistics*, 2011, pp. 269–277.
[75] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE." *Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2579–2605, 2008.

BILIAN CHEN received her Ph.D. degree from The Chinese University of Hong Kong in 2012. Now she is an associate professor in Xiamen University. Her research interests include machine learning, optimization theory and recommendation system. Her publications appear in IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Neural Networks and Learning Systems, SIAM Journal on Optimization, and so on.

JIEWEN GUAN received his M.Eng. degree from Xiamen University in 2022.

ZHENING LI is currently with the School of Mathematics and Physics and the Centre for Operational Research and Logistics in the University of Portsmouth, UK.

ZHEHAO ZHOU received his M.Eng. degree from Xiamen University in 2021. His research interests include machine learning and tensor optimization.