Late Fusion via Subspace Search With Consistency Preservation

Xuanyi Dong, Yan Yan, Mingkui Tan, Yi Yang, and Ivor W. Tsang

***Abstract*— In many real-world applications, data can be represented by multiple ways or multi-view features to describe various characteristics of data. In this sense, the prediction performance can be significantly improved by taking advantages of these features together. Late fusion, which combines the predictions of multiple features, is a commonly used approach to make the final decision for a test instance. However, it is ubiquitous that different features dispute the prediction on the same data with each other, leading to performance degeneration. In this paper, we propose an efficient and effective matrix factorization-based approach to fuse predictions from multiple sources. This approach leverages a hard constraint on the matrix rank to preserve the consistency of predictions by various features, and we thus named it as Hard-rank Constraint Matrix Factorization-based fusion (HCMF). HCMF can avoid the performance degeneration caused by the controversy of multiple features. Extensive experiments demonstrate the efficacy of HCMF for outlier detection and the performance improvement, which outperforms the state-of-the-art late fusion algorithms on many data sets.**

***Index Terms*— Late fusion, matrix factorization, classification.**

I. Introduction

FEATURE representation of objects, which transforms raw data into numerical features, is a prerequisite for most real-world applications. There are often multiple ways to generate numerical features from raw data for visual recognition. For example, for image data, one can construct hand-crafted features such as scale-invariant feature transform (SIFT) feature [1] and Histograms of Gradients (HOG) feature [2], or extract features based on well-trained convolutional neural networks (CNNs) [3]. Moreover, text data can be represented by term frequency-inverse document frequency (tf-idf) and word2vec. Each category of features tends to capture some specific characteristics of data, such as textures of images and frequencies of text data. When building a prediction model, the model trained on one kind of features could be biased and hence may incur unsatisfactory performance. Existing works have investigated the performance improvement by combining multiple kinds of features in real-world applications [4]–[9].

Late fusion is a typical approach to obtaining the final decision on testing data. It aims to fuse the predictions from different features [5]–[7]. A commonly used method is to learn weights for predictions from various features to combine them, which has a number of variations [4], [6], [7]. Among these algorithms, multiple kernel learning (MKL) [10], [11] is often used to train such a weighted combination. However, a single feature may produce corrupted predictions on some data, which likely degenerates performance. To deal with this issue, therefore, some works formulate it as a robust low-rank matrix recovery problem [5], [12] by nuclear norm minimization. Nevertheless, nuclear norm minimization applied in these works requires singular value decomposition (SVD), which would be computationally unaffordable for large matrices.

To avoid the expensive nuclear norm minimization, in this paper, we propose a more efficient matrix factorization (MF) algorithm to recover the underlying low-rank label matrix. Particularly, our method searches the solution in a low-rank subspace under a fixed-rank constraint. This is a hard constraint, and thus different from [5] and [12] which introduce a soft constraint on the low-rank property. On the other hand, to remove the abnormal predictions generated by individual features, we propose to filter out these inconsistent results by using the $\ell_{2,1}$ regularization. Especially, $\ell_{2,1}$ loss can preserve the fidelity within each column by the ℓ_2 loss in the vertical direction. It is also robust to sparse errors across all the columns by ℓ_1 loss in the horizontal direction. Compared to $\ell_{2,1}$ loss, ℓ_2 loss is sensitive to outliers, thus may not produce robust and satisfactory predictions. ℓ_1 loss is not sensitive to outliers, but not capable to detect the abnormal features which generate corrupted results. Besides, most existing MF optimization methods only consider smooth loss functions, rather than the non-smooth loss. Therefore, we, in particular, present an approach based on the augmented Lagrangian multiplier (ALM) to extend MF methods to non-smooth problems, and the framework is illustrated in Figure 1.

Manuscript received December 23, 2017; revised May 25, 2018 and July 22, 2018; accepted August 25, 2018. Date of publication August 30, 2018; date of current version October 1, 2018. This work was supported in part by the Data to Decisions CRC (D2D CRC), in part by ARC under Grant FT130100746 and Grant DP180100106, in part by the National Natural Science Foundation of China under Grant 61602185, and in part by the Recruitment Program for Young Professionals. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Khan M. Iftekharuddin.
(Corresponding author: Yi Yang.)

X. Dong, Y. Yan, and I. W. Tsang are with the Centre for Artificial Intelligence, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: xuanyi.dong@student.uts.edu.au; yan.yan-3@student.uts.edu.au; ivor.tsang@uts.edu.au)

M. Tan is with the School of Software Engineering, South China University of Technology, Guangzhou 510640, China (e-mail: mingkuitan@scut.edu.cn).

Y. Yang is with the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100864, China, and also with the Centre for Artificial Intelligence, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: yi.yang@uts.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2018.2867747

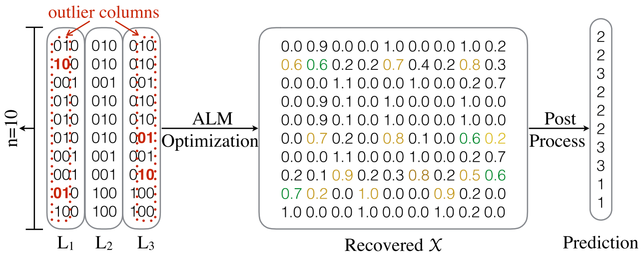


Fig. 1. The framework of HCMF. In this example, there are 10 instances, 3 classifiers, and 3 classes. Predictions of each classifier are converted to a binary indicator matrix, denoted by L_1, L_2, and L_3. Let $L = [L_1, L_2, L_3]$ be the label matrix. HCMF detects and removes the outliers on L and then recovers the underlying low-rank subspace of L as the predictions on the testing data. Lastly, we apply a post process to generate the fusion results.

TABLE I

NOTATION AND SYMBOLS USED IN THIS PAPER

Symbols	Description
n	the number of testing instances
m	the number of classifiers
x_i	the instance
$\mathcal{C}$	the number of classes
$\mathbb{R}$	the set of real numbers
f_i	the i-th classifier
$\mathbf{L}_i$	the indicator matrix for predictions of i-th classifier
$\mathbf{L}$	the horizontal concatenation of all $\mathbf{L}_i$
$\mathbf{X}$	the recovered matrix from $\mathbf{L}$
$\|\|\cdot\|\|_{2,1}$	the $\ell_{2,1}$ norm
$\mathbf{U}$, $\mathbf{V}$	two factor matrices that constrain the rank of $\mathbf{X}$
$\boldsymbol{\lambda}$	the Lagrange multipliers.
μ	the penalty in the augmented Lagrangian function
ρ	the increasing factor for μ

The main contributions are summarized as follows:

(1) We formulate the late fusion problem as a fixed-rank robust matrix recovery problem, which is the first attempt to introduce a hard constraint on the low-rank property to the late fusion problem. HCMF intuitively removes inconsistent predictions caused by abnormal features and preserves the consistency of predictions from different sources.

(2) We propose an ALM-based optimization algorithm to deal with the proposed low-rank-constrained problem efficiently. Two challenges in this problem are the fixed-rank constraint and non-smoothness of $\ell_{2,1}$ loss. Our method decouples the two difficulties and handles them separately.

(3) We provide theoretical convergence analysis for the proposed robust matrix recovery algorithm. Our method has a lower computation complexity than most previous methods.

(4) We empirically demonstrate the efficacy of our method regarding outlier detection and show the better performance achieved by the hard constraint on the matrix rank. Moreover, compared to existing late fusion and feature fusion algorithms, our proposed algorithm shows significant improvements.

II. RELATED WORK

MKL is a typical method to combine different features for the multi-modal fusion. Gehler and Nowozin [4] proposed boosting approaches to ensemble different features by the weighted fusion. Lai *et al.* [7] proposed a sample specific late fusion based on graph Laplacian with RankSVM style constraints. Xu *et al.* [6] proposed an optimization algorithm to learn the weights, thresholding, and smoothing parameters jointly. However, these MKL based fusion methods only learn weights to combine different basic models, and they do not consider the outlier rejection in the fusion process explicitly.

MF has attracted increasing attention recently. Many manifold optimization approaches on MF have been proposed by exploiting the manifold geometry in subspace. Based on the manifold optimization, fixed-rank matrices are supposed to belong to a smooth matrix manifold [13]–[15]. Vandereycken [13] proposed a low-rank geometric conjugate gradient (LRGeomCG) method. Boumal and Absil [14] applied first- and second-order Riemannian trust-region methods to solve the low-rank matrix completion by exploiting the low-rank constraint. Ngo and Saad [15] described gradient methods based on a scaled metric on the Grassmann manifold for low-rank matrix completion. They required the incoherent matrix assumption and fresh measurements at each iteration, and thus their method cannot be guaranteed the global optimality in our work. Besides, the Riemannian manifold method may converge faster than the alternating minimization approach [13]. However, most MF methods only consider the general least square loss, which cannot be directly applied to handle $\ell_{2,1}$ loss.

Feature fusion algorithms [8], [16] combines different models in the feature level, which costs more computational expense than the late fusion. They are proposed for specific tasks. For example, Feichtenhofer *et al.* [8] explored a variety of fusion methods to combine spatial and temporal CNN models, significantly improving the performance of two-stream network [16]. It requires additional effort to extend these methods to general problems. Other researchers leverage the fusion idea into their framework, such as [17]–[23].

III. THE PROPOSED APPROACH

We first elaborate on the basic formulation of HCMF. We then present the details of using ALM extending MF to handle $\ell_{2,1}$ loss. Lastly, we introduce the post-process strategy to convert the recovered label matrix to the final predictions.

A. Problem Formulation

We introduce most notation and symbols in Table I. Suppose there are n testing instances and $\mathcal{C}$ classes. We denote each instance as $x_i \in \mathbb{R}^d$, where $1 \leq i \leq n$ and $\mathbb{R}^d$ denotes the d-dimension feature vector. Assume that we have already trained m classifiers, i.e., $f_1, f_2, \ldots f_m$. Then by applying each single classifier f_j $(1 \leq j \leq m)$ on the entire testing data, one can derive an indicator matrix $\mathbf{L}_j \in \mathbb{R}^{n\times\mathcal{C}}$, where $\mathbf{L}_{j,ic} = 1$ if the i-th instance is assigned to the c class, otherwise $\mathbf{L}_{j,ic} = 0$ (as shown in Figure 1). $\mathbf{L} \in \mathbb{R}^{n\times m\mathcal{C}}$ is the horizontal concatenation of $\mathbf{L}_j$, where $1 \leq j \leq m$. Since the matrix $\mathbf{L}$ may contain outliers, HCMF aims to detect and remove outliers from $\mathbf{L}$ by recovering $\mathbf{L}$ into another matrix $\mathbf{X} \in \mathbb{R}^{n\times m\mathcal{C}}$. On the one hand, to preserve the consistency among m classifiers, we introduce a rank constraint on $\mathbf{L}$. On the other hand, to catch the outlier columns, we apply robust $\ell_{2,1}$ loss, which is computed by $||\mathbf{X}||_{2,1} = \sum_{j=1}^{m\mathcal{C}} \sqrt{\sum_{i=1}^{n} \mathbf{X}_{ij}^2}$. Therefore, a general formulation of recovering the underlying low-rank label matrix can be written as follows:

$$\min_{\mathbf{X}} \; ||\mathbf{L}-\mathbf{X}||_{2,1} \quad \text{s.t. } \operatorname{rank}(\mathbf{X}) = p, \tag{1}$$

where $\mathbf{X}$ is the underlying low-rank label matrix to be recovered, $\operatorname{rank}(\mathbf{X})$ obtains the rank of $\mathbf{X}$, and p is an integer. In our setting, to preserve the consistency among various classifiers, we use the hard constraint on the rank, i.e., $p = \mathcal{C}$.

It is difficult to optimize Problem (1) directly due to the presence of the rank constraint. We hence transform the original problem as an MF formulation as follows:

$$\begin{aligned} &\min_{\mathbf{X}} \; ||\mathbf{L}-\mathbf{X}||_{2,1} \\ &\text{s.t. } \mathbf{X} = \mathbf{U}\mathbf{V}, \quad \mathbf{U} \in \mathbb{R}^{n\times p}, \; \mathbf{V} \in \mathbb{R}^{p\times m\mathcal{C}}, \end{aligned} \tag{2}$$

where $\mathbf{U}$ and $\mathbf{V}$ are two factor matrices. If the $\mathbf{U}$ and $\mathbf{V}$ have the dimension of $n \times p$ and $p \times m\mathcal{C}$, then the rank of matrix $\mathbf{X}$ will be less than p. This is the rank constraint shown in Eq. (1). Most existing MF algorithms focus on smooth loss functions, rather than a non-smooth $\ell_{2,1}$ loss function [13]–[15]. In the following section, we present an ALM optimization framework to solve Problem (2).

B. Optimization Based on ALM

Most MF optimization methods only consider smooth loss functions instead of the non-smooth $\ell_{2,1}$ loss in Problem (2). In this section, we present an ALM-based algorithm to optimize Problem (2). By introducing a new variable $\mathbf{E} = \mathbf{L} - \mathbf{X}$, we can develop Problem (2) as below:

$$\min_{\mathbf{E},\mathbf{X}} \; ||\mathbf{E}||_{2,1} \quad \text{s.t. } \mathbf{E} = \mathbf{L} - \mathbf{X}, \; \operatorname{rank}(\mathbf{X}) = p. \tag{3}$$

The augmented Lagrangian function of Problem (2) is constructed as below:

$$\begin{aligned} \mathcal{L}(\mathbf{X}, \mathbf{E}, \boldsymbol{\lambda}, \mu) = ||\mathbf{E}||_{2,1} + \langle \boldsymbol{\lambda}, \mathbf{L}-\mathbf{X}-\mathbf{E}\rangle \\ + \frac{\mu}{2}||\mathbf{L}-\mathbf{X}-\mathbf{E}||_F^2, \end{aligned} \tag{4}$$

where $\operatorname{rank}(\mathbf{X}) = p$, μ is a scalar, and $\boldsymbol{\lambda} \in \mathbb{R}^{n\times m\mathcal{C}}$ is the Lagrangian multiplier. Here $\langle \mathbf{A}, \mathbf{B}\rangle$ is the inner product of $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{B} \in \mathbb{R}^{m\times n}$, and defined as $\langle \mathbf{A}, \mathbf{B}\rangle = \sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{A}_{ij}\mathbf{B}_{ij}$. Then we can update $\mathbf{X}$ and $\mathbf{E}$ alternatively. We summarize this procedure in Algorithm 1, where ρ is the factor that increases the penalty parameter *mu* per iteration. Algorithm 1 involves two sub-problems, which solve the $\mathbf{X}$ and $\mathbf{E}$, respectively. Next we will introduce these two sub-problems in Sec. III-C and Sec. III-D and then provide convergence analysis of Algorithm 1 in Sec. III-E. Last we present the strategies for post process in Sec. V-F.

Algorithm 1 The ALM Algorithm for Problem (3)

Input: $\mathbf{L} \in \mathbb{R}^{n\times m\mathcal{C}}$, $\operatorname{rank}(\mathbf{X}) = p$
  Initialize $\rho > 1$, $\boldsymbol{\lambda}_0 = 0$, $\mathbf{E}_0 = 0$, and $\mu_0 > 0$.
  for $t = 0$; $||\mathbf{E}_{t+1} - \mathbf{E}_t||_F \geq \epsilon$; t**++** **do**
    while not converge **do**
      1: Obtain $\mathbf{X}_{t+1}$ by solving Problem (6)
      2: Obtain $\mathbf{E}_{t+1}$ by solving Problem (12)
    end while
    3: $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \mu_t(\mathbf{L} - \mathbf{X}_{t+1} - \mathbf{E}_{t+1})$.
    4: $\mu_{t+1} = \rho\mu_t$.
  end for
Output: $\mathbf{X} \in \mathbb{R}^{n\times m\mathcal{C}}$

C. Subproblem Optimization w.r.t. X

To update $\mathbf{X}$, we have the following augmented Lagrangian function w.r.t. $\mathbf{X}$:

$$\begin{aligned} \mathcal{L}(\mathbf{X}) &= \langle \boldsymbol{\lambda}, \mathbf{L}-\mathbf{X}-\mathbf{E}\rangle + \frac{\mu}{2}||\mathbf{L}-\mathbf{X}-\mathbf{E}||_F^2 \\ &= \frac{\mu}{2}||\mathbf{M}-\mathbf{X}||_F^2 - \frac{1}{2\mu}||\boldsymbol{\lambda}||_F^2, \end{aligned} \tag{5}$$

where $\mathbf{M} = \mathbf{L} - \mathbf{E} + \frac{\boldsymbol{\lambda}}{\mu}$. Eq. (5) is similar to Eq. (4) but fixes $\mathbf{E}$, λ, and μ. Since we fix $\mathbf{E}$, the $||\mathbf{E}||_{2,1}$ term in Eq. (4) becomes the constant value, and we thus omit $||\mathbf{E}||_{2,1}$ in Eq. (5). Hence, we derive the following subproblem w.r.t. $\mathbf{X}$:

$$\min_{\mathbf{X}} \; ||\mathbf{M}-\mathbf{X}||_F^2, \quad \text{s.t. } \operatorname{rank}(\mathbf{X}) = p. \tag{6}$$

Problem (6) contains a smooth loss function of $\mathbf{X}$, and we thus propose to solve this subproblem by exploiting Riemannian geometry of smooth fixed-rank matrices, which is briefly presented as below.

The key idea of manifold optimization is that the update of $\mathbf{X}$ is always performed on the same manifold, which ensures the rank of $\mathbf{X}$ unchanged as the optimization iterates. In other words, we search the solution for $\mathbf{X}$ in a low-rank subspace. A smooth manifold of fixed-rank-p matrices is defined as [13]:

$$\begin{aligned} \mathcal{M}_p &= \{\mathbf{X} \in \mathbb{R}^{n\times m\mathcal{C}} : \operatorname{rank}(\mathbf{X}) = p\} \\ &= \{\mathbf{U}\operatorname{diag}(\boldsymbol{\sigma})\mathbf{V}^{\mathsf{T}} : \mathbf{U} \in \mathrm{St}_p^n, \mathbf{V} \in \mathrm{St}_p^{m\mathcal{C}}, ||\boldsymbol{\sigma}||_0 = p\} \end{aligned}$$

where $\mathrm{St}_p^n = \{\mathbf{U} \in \mathbb{R}^{n\times p} : \mathbf{U}^{\mathsf{T}}\mathbf{U} = \mathbf{I}\}$ denotes the Stiefel manifold of $n \times p$ real and orthonormal matrices. We denote the tangent space by $T_{\mathbf{X}}\mathcal{M}_p$ of $\mathcal{M}_p$ at $\mathbf{X} = \mathbf{U}\operatorname{diag}(\boldsymbol{\sigma})\mathbf{V}^{\mathsf{T}} \in \mathbb{R}^{n\times m\mathcal{C}}$, which is obtained as below:

$$\begin{aligned} &T_{\mathbf{X}}\mathcal{M}_p \\ &= \{\mathbf{U}\mathbf{M}\mathbf{V}^{\mathsf{T}} + \mathbf{U}_r\mathbf{V}^{\mathsf{T}} + \mathbf{U}\mathbf{V}_r^{\mathsf{T}} : \mathbf{M} \in \mathbb{R}^{p\times p}, \\ &\qquad \mathbf{U}_r \in \mathbb{R}^{n\times p}, \mathbf{U}_r^{\mathsf{T}}\mathbf{U} = \mathbf{0}, \mathbf{V}_r \in \mathbb{R}^{m\mathcal{C}\times p}, \mathbf{V}_r^{\mathsf{T}}\mathbf{V} = \mathbf{0}\}. \end{aligned} \tag{7}$$

One can define a metric $g_{\mathbf{X}}(\mathbf{A}, \mathbf{B}) = \langle \mathbf{A}, \mathbf{B}\rangle$ on $\mathcal{M}_p$, with $\mathbf{X} \in \mathcal{M}_p$ and $\mathbf{A}, \mathbf{B} \in T_{\mathbf{X}}\mathcal{M}_p$, then $\mathcal{M}_p$ becomes a Riemannian manifold by restricting $\langle \mathbf{A}, \mathbf{B}\rangle$ to the *tangent bundle*, defined as the disjoint union of all tangent spaces: $T\mathcal{M}_p = \bigcup_{\mathbf{X}\in\mathcal{M}_p}\{\mathbf{X}\} \times T_{\mathbf{X}}\mathcal{M}_p = \{(\mathbf{X}, \mathbf{P}) \in \mathbb{R}^{n\times m\mathcal{C}} \times \mathbb{R}^{n\times m\mathcal{C}} : \mathbf{X} \in \mathcal{M}_p, \mathbf{P} \in T_{\mathbf{X}}\mathcal{M}_p\}$.

Let $f(\mathbf{X}) = ||\mathbf{M} - \mathbf{X}||_F^2$. Suppose that $\mathbf{G} = \nabla f(\mathbf{X})$ in Euclidean space is at $\mathbf{X} = \mathbf{U}\operatorname{diag}(\boldsymbol{\sigma})\mathbf{V}^{\mathsf{T}}$. Then, according

Algorithm 2 Computation of Grad $f(\mathbf{X})$ (Algorithm 2 in [13])

Input: $\mathbf{X} = \mathbf{U}\mathrm{diag}(\sigma)\mathbf{V}^\mathsf{T} \in \mathcal{M}_r$, $\mathbf{G}$.
1. $\mathbf{R}_u \leftarrow \mathbf{G}^\mathsf{T}\mathbf{U}$, $\mathbf{R}_v \leftarrow \mathbf{G}\mathbf{V}$.
2. $\mathbf{M} \leftarrow \mathbf{U}^\mathsf{T}\mathbf{R}_v$.
3. $\mathbf{U}_r \leftarrow \mathbf{R}_v - \mathbf{U}\mathbf{M}$, $\mathbf{V}_r \leftarrow \mathbf{R}_u - \mathbf{V}\mathbf{M}^\mathsf{T}$.

Output: $\mathrm{grad} f(\mathbf{X}) = \mathbf{U}\mathbf{M}\mathbf{V}^\mathsf{T} + \mathbf{U}_r\mathbf{V}^\mathsf{T} + \mathbf{U}\mathbf{V}_r^\mathsf{T} \in T_\mathbf{X}\mathcal{M}_p$.

Algorithm 3 LRGeomCG (Algorithm 1 in [13])

Input: Initial $\mathbf{X}_1 \in \mathcal{M}_p$, tangent vector $\boldsymbol{\eta}_0 = \mathbf{0}$, $k = 1$.
while not converge **do**
1. Compute Riemannian gradient $\mathbf{P}_k = \mathrm{grad} f(\mathbf{X}_k)$ by Algorithm 2.
2. Compute a conjugate direction with PR+: $\boldsymbol{\eta}_k = -\mathbf{P}_k + \beta_k \mathcal{T}_{\mathbf{X}_{k-1}\to\mathbf{X}_k}(\boldsymbol{\eta}_{k-1}) \in T\mathcal{M}_p$.
3. Find a step size $t_k = \min_t f(\mathbf{X} + t\boldsymbol{\eta}_k)$
4. Update $\mathbf{X}_{k+1} = R_{\mathbf{X}_k}(t_k\boldsymbol{\eta}_k)$.
5. $k = k + 1$.

end while
Output: $\mathbf{X} \in \mathcal{M}_p$.

to [13], the Riemannian gradient of $f(\mathbf{X})$ on $\mathcal{M}_p$ is given as the orthogonal projection of $\mathbf{G}$ onto the tangent space at $\mathbf{X}$:

$$\mathrm{grad} f(\mathbf{X}) = P_{T_\mathbf{X}\mathcal{M}_p}(\mathbf{G}), \tag{8}$$

where $P_{T_\mathbf{X}\mathcal{M}_p}(\mathbf{Z}) : \mathbf{Z} \mapsto P_U\mathbf{Z}P_V + P_U^\perp\mathbf{Z}P_V + P_U\mathbf{Z}P_V^\perp$ denotes the orthogonal projection of any $\mathbf{Z} \in \mathbb{R}^{n\times m\mathcal{C}}$ onto the tangent space at $\mathbf{X} = \mathbf{U}\mathrm{diag}(\boldsymbol{\sigma})\mathbf{V}^\mathsf{T}$. Here $P_U = \mathbf{U}\mathbf{U}^\mathsf{T}$ and $P_U^\perp = \mathbf{I} - \mathbf{U}\mathbf{U}^\mathsf{T}$ for any $\mathbf{U} \in \mathrm{St}_p^n$.

Lemma 1: Assuming that $\mathbf{U}_r$, $\mathbf{V}_r$ *and* $\mathbf{M}$ *are computed by Algorithm 2, we can obtain* $\mathrm{grad} f(\mathbf{X}) = \mathbf{U}\mathbf{M}\mathbf{V} + \mathbf{U}_r\mathbf{V}^\mathsf{T} + \mathbf{U}\mathbf{V}_r^\mathsf{T}$, *which is equivalent to the definition of grad* $f(\mathbf{X})$ *in* (8).

Proof:

$$\begin{aligned}
&\mathrm{grad} f(\mathbf{X}) \\
&\quad = \mathbf{U}\mathbf{M}\mathbf{V}^\mathsf{T} + \mathbf{U}_r\mathbf{V}^\mathsf{T} + \mathbf{U}\mathbf{V}_r^\mathsf{T} \\
&\quad = \mathbf{U}(\mathbf{U}^\mathsf{T}\mathbf{G}\mathbf{V})\mathbf{V}^\mathsf{T} + (\mathbf{R}_v - \mathbf{U}\mathbf{M})\mathbf{V}^\mathsf{T} + \mathbf{U}(\mathbf{R}_u - \mathbf{V}\mathbf{M}^\mathsf{T})^\mathsf{T} \\
&\quad = P_\mathbf{U}\mathbf{G}P_V + \mathbf{G}\mathbf{V}\mathbf{V}^\mathsf{T} - \mathbf{U}\mathbf{M}\mathbf{V}^\mathsf{T} + \mathbf{U}\mathbf{U}^\mathsf{T}\mathbf{G} - \mathbf{U}\mathbf{M}\mathbf{V}^\mathsf{T} \\
&\quad = P_\mathbf{U}\mathbf{G}P_V + \mathbf{G}P_V - 2\mathbf{U}\mathbf{M}\mathbf{V}^\mathsf{T} + P_U\mathbf{G} \\
&\quad = P_\mathbf{U}\mathbf{G}P_V + \mathbf{G}P_V - 2\mathbf{U}(\mathbf{U}^\mathsf{T}\mathbf{R}_v)\mathbf{V}^\mathsf{T} + P_U\mathbf{G} \\
&\quad = P_\mathbf{U}\mathbf{G}P_V + \mathbf{G}P_V - 2\mathbf{U}(\mathbf{U}^\mathsf{T}\mathbf{G}\mathbf{V})\mathbf{V}^\mathsf{T} + P_U\mathbf{G} \\
&\quad = P_\mathbf{U}\mathbf{G}P_V + (\mathbf{I} - P_U)\mathbf{G}P_V + P_U\mathbf{G}(\mathbf{I} - P_V) \\
&\quad = P_\mathbf{U}\mathbf{G}P_V + P_U^\perp\mathbf{G}P_V + P_U\mathbf{G}P_V^\perp.
\end{aligned}$$

□

Based on the above discussion, the solution of $\mathbf{X}$ can be solved by Algorithm 3. Particularly, in Step 2 of Algorithm 3, β_k is determined by a Polak-Ribière (PR+) rule [13]:

$$\beta_k = \frac{\mathrm{grad} f(\mathbf{X}_k)^\mathsf{T}(\mathrm{grad} f(\mathbf{X}_k) - \mathrm{grad} f(\mathbf{X}_{k-1}))}{\langle \mathrm{grad} f(\mathbf{X}_{k-1}), \mathrm{grad} f(\mathbf{X}_{k-1})\rangle}. \tag{9}$$

The step size t_k in Step 3, given a search direction $\boldsymbol{\eta}_k \in T_{\mathbf{X}_k}\mathcal{M}_p$, is chosen such that

$$f(R_{\mathbf{X}_k}(t_k\boldsymbol{\eta}_k)) \le f(\mathbf{X}_k) + c_1 t_k \langle \mathrm{grad} f(\mathbf{X}_k), \boldsymbol{\eta}_k\rangle, \tag{10}$$

where $0 \le c_1 \le \frac{1}{2}$. The notation $\mathcal{T}_{\mathbf{X}_{k-1}\to\mathbf{X}_k}(\boldsymbol{\eta}_{k-1})$ in Step 2 and $R_{\mathbf{X}_k}(t_k\boldsymbol{\eta}_k)$ denote *vector transport* and *retraction*, respectively. For more details of above two operators and the geometry of $\mathcal{M}_p$, see [13] and the references therein.

D. Subproblem Optimization w.r.t. E

Similarly, to update $\mathbf{E}$, we have the following problem:

$$\min_{\mathbf{E}} \ ||\mathbf{E}||_{2,1} + ||\mathbf{N} - \mathbf{E}||_F^2, \tag{11}$$

where $\mathbf{N} = \mathbf{L} - \mathbf{X} + \frac{\boldsymbol{\lambda}}{\mu}$. The above problem can be efficiently solved by the column-wise soft-thresholding operator [24]:

$$\mathbf{E}_i = \mathcal{S}_\alpha(\mathbf{N}_i) = \begin{cases} \mathbf{0}, & \text{if } ||\mathbf{N}_i||_2 \le \alpha \\ \mathbf{N}_i - \dfrac{\alpha\mathbf{N}_i}{||\mathbf{N}_i||_2}, & \text{otherwise,} \end{cases} \tag{12}$$

where $\mathbf{E}_i$ and $\mathbf{N}_i$ denote the i-th column of $\mathbf{E}$ and $\mathbf{N}$, and $\alpha = \frac{1}{2}$.

E. Convergence Analysis

The following lemma and theorem provide the theoretical convergence guarantee of the proposed algorithm. We emphasize that our problem is a non-convex problem with a rank constraint, which is rarely studied in the previous literature.

Lemma 2: Given that $\rho > 1$, *the sequence* $\{\mu_t\}$ *is always increasing,* $\mathbf{E}_0 = 0$, $\boldsymbol{\lambda}_0 = 0$ *and* $\mu_0 > 0$, *suppose that there exists* $D \ge 0$ *such that* $||\mathbf{v}||_F \le D$ *for all* $\mathbf{v} \in \frac{\partial||\mathbf{E}||_{2,1}}{\partial\mathbf{E}}$. *Then the sequence* $\{\boldsymbol{\lambda}_t\}$ *computed by Algorithm 1 is bounded, i.e.,* $||\boldsymbol{\lambda}_t||_F \le D$ *for any* $t \ge 0$.

Proof: By the optimality of $\mathbf{E}_{t+1}$ in Step 2 in Algorithm 1, we have the sub-differential of $\mathcal{L}$ w.r.t. $\mathbf{E}$ as zeros:

$$\begin{aligned}
0 \ &\in \ \frac{\mathcal{L}(\mathbf{X}_{t+1}, \mathbf{E}_{t+1}, \boldsymbol{\lambda}_t, \mu_t)}{\mathbf{E}_{t+1}} \\
&= \partial(||\mathbf{E}_{t+1}||_{2,1}) + \boldsymbol{\lambda}_t + \mu_t(\mathbf{L} - \mathbf{X}_{t+1} - \mathbf{E}_{t+1}) \\
&= \partial(||\mathbf{E}_{t+1}||_{2,1}) + \boldsymbol{\lambda}_{t+1} \\
&\Rightarrow -\boldsymbol{\lambda}_{t+1} \in \partial(||\mathbf{E}_{t+1}||_{2,1}).
\end{aligned} \tag{13}$$

The second equation is due to Step 3 of Algorithm 1. Consider the computation of $\mathbf{E}_{t+1}$ in Eq. (12), which is associated with $\mathbf{L}, \mathbf{X}_{t+1}$, $\boldsymbol{\lambda}_t$, and μ_t. Then we will analyze those variables respectively.

$\mathbf{L}$ is the observation and thus can be bounded. $\mathbf{X}_{t+1}$ is bounded due to the convergence guarantee of Step 1 in Algorithm 1 [13].

$\{\boldsymbol{\lambda}_t\}$ is initialized by setting $\boldsymbol{\lambda}_0 = 0$ and $\{\mu_t\}$ is non-decreasing. Therefore, as Algorithm 1 runs, all the elements of $\mathbf{E}_{t+1}$, $\partial(\mathbf{E}_{t+1})$ and $\boldsymbol{\lambda}_t$ are bounded accordingly as t increases. This makes some non-negative D exist such that $\frac{\partial||\mathbf{E}_{t+1}||_{2,1}}{\partial\mathbf{E}_{t+1}} \le D$. Furthermore, we have $||\boldsymbol{\lambda}_t||_F \le D$ for any $t > 0$. This completes the proof. □

Theorem 1: Suppose the sequences $\{\mathbf{X}_t\}_{t=1}^\infty$, $\{\mathbf{E}_t\}_{t=1}^\infty$ *and* $\{\boldsymbol{\lambda}_t\}_{t=1}^\infty$ *are generated by Algorithm 1. As* $t \to \infty$, *the gradients of* $\mathcal{L}$ *w.r.t.* $\mathbf{X}$ *and* $\mathbf{E}$ *vanish, thus any accumulation point* $(\mathbf{X}^*, \mathbf{E}^*)$ *is a stationary point.*

Proof: Starting with Step 3 in Algorithm 1, we have

$$\begin{aligned}\mathbf{L}-\mathbf{X}_{t+1}-\mathbf{E}_{t+1} &= \frac{1}{\mu_t}(\boldsymbol{\lambda}_{t+1}-\boldsymbol{\lambda}_t)\\ &\Rightarrow ||\mathbf{L}-\mathbf{X}_{t+1}-\mathbf{E}_{t+1}||_F=\frac{1}{\mu_t}||\boldsymbol{\lambda}_{t+1}-\boldsymbol{\lambda}_t||_F.\end{aligned}$$

Since μ_t always increases as t increases, consider that when $\mu_t \to +\infty$, we have

$$||\mathbf{L}-\mathbf{X}_{t+1}-\mathbf{E}_{t+1}||_F \le \frac{1}{\mu_t}(||\boldsymbol{\lambda}_{t+1}||_F+||\boldsymbol{\lambda}_t||_F) \to 0,$$

which implies that all the elements in $\mathbf{L}-\mathbf{X}_{t+1}-\mathbf{E}_{t+1}$ is approaching to 0. Here we employ the result of Lemma 2.

By employing the result in [13, Proposition 4.2] with a suitable choice of regularization, for $t \to \infty$, we have the following result after running Step 1 of Algorithm 1:

$$||P_{T_{\mathbf{X}_{t+1}}\mathcal{M}_p}(\mathbf{L}-\mathbf{X}_{t+1}-\mathbf{E}_t+\frac{\boldsymbol{\lambda}_t}{\mu_t})||_F \to 0,$$

which indicates that the Riemannian gradient of $\mathbf{X}_{t+1}$ is approaching to 0.

Recall the definition of the orthogonal projection in Eq. (8), i.e., $P_{T_{\mathbf{X}}\mathcal{M}_p}(\mathbf{Z}) : \mathbf{Z} \mapsto P_U\mathbf{Z}P_V + P_U^{\perp}\mathbf{Z}P_V + P_U\mathbf{Z}P_V^{\perp}$, we have

$$\begin{aligned}&||P_{T_{\mathbf{X}_{t+1}}\mathcal{M}_p}(\mathbf{L}-\mathbf{X}_{t+1}-\mathbf{E}_t+\frac{\boldsymbol{\lambda}_t}{\mu_t})||_F\\ &= ||P_{T_{\mathbf{X}_{t+1}}\mathcal{M}_p}(\mathbf{L}-\mathbf{X}_{t+1}-\mathbf{E}_{t+1}+\frac{\boldsymbol{\lambda}_t}{\mu_t})\\ &\quad +P_{T_{\mathbf{X}_{t+1}}\mathcal{M}_p}(\mathbf{E}_{t+1}-\mathbf{E}_t)||_F \to 0.\end{aligned}$$

Considering that all the elements in $\mathbf{L}-\mathbf{X}_{t+1}-\mathbf{E}_{t+1}$ is approaching to 0, we can also derive that all the elements in $P_{T_{\mathbf{X}_{t+1}}\mathcal{M}_p}(\mathbf{E}_{t+1}-\mathbf{E}_t)$ is approaching to 0, and additionally $||P_{T_{\mathbf{X}_{t+1}}\mathcal{M}_p}(\mathbf{E}_{t+1}-\mathbf{E}_t)||_F \to 0$.

When μ_t is very large, we have

$$\begin{aligned}&\mathbf{L}-\mathbf{X}_{t+1}-\mathbf{E}_{t+1}+\frac{\boldsymbol{\lambda}_t}{\mu_t}\\ &\approx \mathbf{L}-\mathbf{X}_{t+1}-\mathbf{E}_{t+1}+\frac{\boldsymbol{\lambda}_{t+1}}{\mu_{t+1}}\\ &\approx \mathbf{L}-\mathbf{X}_{t+1}-\mathbf{E}_{t+1} \to 0,\end{aligned}$$

which implies that

$$||P_{T_{\mathbf{X}_{t+1}}\mathcal{M}_p}(\mathbf{L}-\mathbf{X}_{t+1}-\mathbf{E}_{t+1}+\frac{\boldsymbol{\lambda}_{t+1}}{\mu_{t+1}})||_F \to 0.$$

This furthermore indicates that, before any update in the $(t+2)$-th iteration of Algorithm 1, $\mathbf{X}_{t+1}$ has already made Riemannian gradient ($\mathrm{grad} f(\mathbf{X}_{t+1})$) approach to 0. Therefore, we can observe that Step 1 of Algorithm 1 in the $(t+2)$-th iteration will not change $\mathbf{X}_{t+1}$ very much to make $\mathrm{grad} f(\mathbf{X})$ *vanish*, i.e., $||P_{T_{\mathbf{X}_{t+2}}\mathcal{M}_p}(\mathbf{L}-\mathbf{X}_{t+2}-\mathbf{E}_{t+1}+\frac{\boldsymbol{\lambda}_{t+1}}{\mu_{t+1}})||_F \to 0$. That is $\mathbf{X}_{t+2} \to \mathbf{X}_{t+1}$.

On the other hand, due to the dependence of $\mathbf{E}_{t+2}$ on $\mathbf{X}_{t+2}$, $\mathbf{E}_{t+2}$ will also not change very much from $\mathbf{E}_{t+1}$, which means $\mathbf{E}_{t+2} \to \mathbf{E}_{t+1}$. This proves that $(\mathbf{X}_t, \mathbf{E}_t)$ is a stationary point when $t \to +\infty$. □

F. Post Process

After we recover the low-rank matrix $\mathbf{X}$ from the label assignment matrix $\mathbf{L}$, it requires post-processing to generate the (hard) prediction labels for testing data from $\mathbf{X}$. There can be a large number of approaches to generate the hard labels. In this paper, intuitively, we consider the following three strategies:

[AVE] A prediction score matrix $\mathbf{X}^* \in \mathbb{R}^{n\times\mathcal{C}}$ is obtained by $\mathbf{X}^* = 1/m\sum_{j=1}^m \mathbf{X}_j$, where $\mathbf{X}_j$ is the recovered result for $\mathbf{L}_j$. The indicator of the highest value in each row of $\mathbf{X}^*$ is regarded as the predicted label for the corresponding instance.

[VOTE] For $\mathbf{X}_j$ where $1 \le j \le m$, we set $\mathbf{I}_{ij}$ as the index of the highest value in the i-th row of $\mathbf{X}_j$. Each $\mathbf{I}_{ij}$ contributes one point to the i-th data with label $\mathbf{I}_{ij}$. The label with the most votes is the final prediction result.

[GATE] For $\mathbf{X}_j$ where $1 \le j \le m$ is transformed through the softmax function as a gate of the corresponding confidence score matrix, which is normalized by L_2 norm. As $\mathbf{X}_j$ removes outliers from $\mathbf{L}_j$, we produce the $\mathbf{X}_j$ by the corresponding score matrix after softmax and normalization. The similar process as the first strategy is applied to predict the final labels.

According to the experiments, the last strategy "GATE" usually outperforms best (shown later). We thus choose this strategy as the post process for all experiments by default except for the post process analysis section.

IV. Algorithm Analysis

A. Computation Complexity

1) The Overall Computational Complexity of HCMF: Assume that S and T are the numbers of iterations of Algorithm 1 and Algorithm 3, respectively. In practical experiments, we emphasize that we only perform the inner loop in Line 1 and 2 in Algorithm 1 for one time, which is known as inexact ALM [25]. The per-iteration complexity of updating of $\mathbf{X}$ is $O((n+m\mathcal{C})\mathcal{C}^2)$. According to [13], T is less than 100 for most situations. S, in our experiments, is usually small (10 to 50). The per-iteration complexity of updating $\mathbf{E}$ and post-processing can be easily calculated as $O(nm\mathcal{C})$ and $O(nm\mathcal{C})$. The total per-iteration complexity is thus $O(ST((n+m\mathcal{C})\mathcal{C}^2+O(nm\mathcal{C}))$, which is linear by growing of n and m, with the cubic complexity of $\mathcal{C}$.

2) Computational Complexity of SVD in HCMF: HCMF requires SVD in Algorithm 3 when it performs retraction. However, the computational complexity is $\mathcal{C}^3$ in each iteration of our method, where $\mathcal{C}$ is the estimated rank. We would like to emphasize that in the low-rank matrix recovery scenario, $\mathcal{C}$ is much smaller than n, which the number of rows in the matrix $\mathbf{X}$. Compared to the SVD performed on the matrix $\mathbf{X}$, the computational complexity $\mathcal{C}^3$ is always computationally affordable in the low-rank setting. Most nuclear norm-based methods [5], [12] have the cubic computational complexity on m or n due to the presence of SVDs. Compared with them, our algorithm is more efficient for the large-scale problems. For example, Gao *et al.* [12] reported their entire complexity is $(m\mathcal{C})^3+n(m\mathcal{C})^2$ per iteration. With the increase in the number of classifiers, the computation costs too much to be affordable due to the cubic complexity of m, but ours can still solve the solution.

B. Hard Constraint vs. *Soft Constraint*

The objective function in Eq. (1) is similar to [12]. However, they apply a soft low-rank constraint, $\text{rank}(\mathbf{X}) \leq p$, on the $\ell_{2,1}$ loss for the clustering problem [12]. The soft low-rank constraint is not applicable to the object recognition task, which will be discussed later. Therefore, we leverage the hard low-rank constraint in our proposed algorithm.

C. Limitations and Failure Cases

1) One Testing Sample: Some real-time applications might require the algorithm to handle one image per frame, i.e., $n = 1$ for our algorithm. If $n = 1$, the consistency between different classifiers cannot be estimated correctly due to the lack of data. In this case, our algorithm fails. However, in a typical machine learning task, testing data usually comes in bulk, rather than a data sequence. If a new test sample comes, re-running on the entire updated data is common for fusion methods. It may not be surprising to see accuracy improvement for the fusion algorithm. Our goal is to design a fusion method which improves performance as much as possible given the ideal base predictions. Our empirical results show better performance of our algorithm than other baselines.

2) Poor Classifiers: HCMF aims to find the consistency property among the different prediction results $\mathbf{X}$. If most classifiers perform well, we can easily find the outliers and remove the abnormal predictions from $\mathbf{X}$. In contrast, if most classifiers perform bad, the dominant of $\mathbf{X}$ is the wrong predictions. In this case, the correct predictions may be the outliers, and the fusion result can become worse than a single model. However, such a case is rare in the supervised classification problems.

V. Experimental Studies

A. Synthetic Experiments

This synthetic experiment aims to study the ability of our method to detect the outliers. We first generate a ground-truth label matrix $G \in \{0, 1\}^{n \times \mathcal{C}}$. We set $n = 250$ and $\mathcal{C} = 20$, where, as aforementioned, n is the number of instances and $\mathcal{C}$ is the number of classes. We then create m ($m = 15$) copies of the ground-truth matrix. We randomly permute a certain proportion of their labels, denoted by the noise ratio. In this way, we construct a contaminated matrix $\mathbf{X}$ with the size of $n \times m\mathcal{C}$, and the underlying matrix is a rank-$\mathcal{C}$ matrix.

1) Comparison With the Hard Rank Constraint Algorithm: We use HCMF (hard low-rank constraint) to recover the low-rank matrix $\mathbf{X}$ and compare our result with RCEC [12] (soft low-rank constraint). Figure 2 illustrates the ratio of each eigenvalue from the recovered matrix by principal components analysis (PCA). The recovered matrix represents the prediction based on the results from $m = 15$ different classifiers. To preserve the consistency, the rank of this recovered matrix should be as close as to $\mathcal{C} = 20$. From Figure 2, we can see that the recovered matrix of ours well preserve the low-rank property. However, the soft constraint [12] cannot recover the ratio of each eigenvalue compared to the ground-truth matrix. Therefore, they are not able to recover the rank of $\mathbf{X}$, i.e., preserve the consistency among different classifiers.

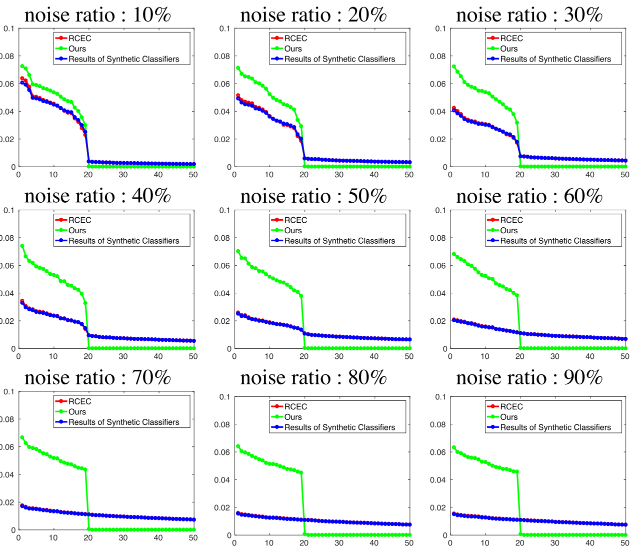


Fig. 2. Comparison of the hard-rank constraint (HCMF) and the soft-rank constraint (RCEC) on the synthetic data. We show the ratio of each eigenvalue to the sum of eigenvalues (y-axis). The x-axis indicates indexes of eigenvalue, sorted from large to small. HCMF preserves the low-rank property. However, RCEC can hardly recover the matrix into the original rank.

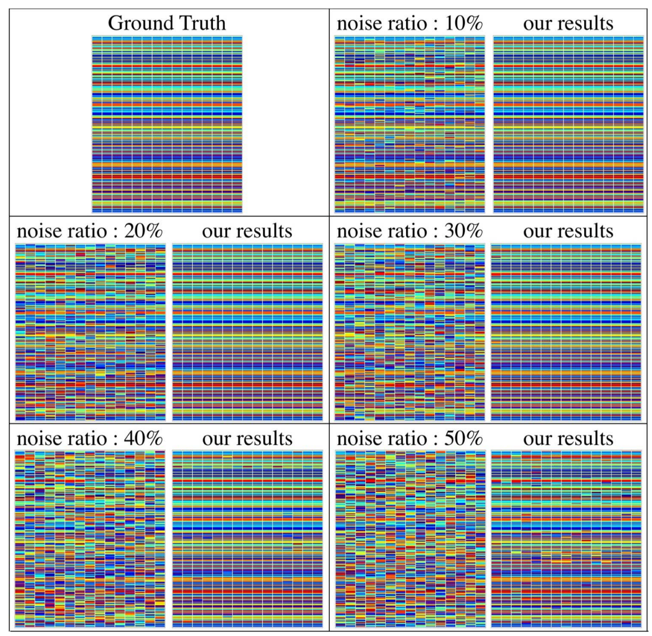


Fig. 3. Visualization results of HCMF on the synthetic dataset. Each sub-figure has 15 columns, where each of them represents the result of one classifier. In each sub-figure, each row corresponds to one instance, where different colors represent different class labels. The "Ground Truth" figure shows the visualized ground truth data. In each block (apart from the first block), we visualize the randomly contaminated results of 15 classifiers, and the recovered results from our algorithm. Most outlier elements are corrected, even if there is 50% random noise.

2) Visualization on the Synthetic Data: To better demonstrate the efficacy, we visualize the recovered result of $\mathbf{X}$ by our algorithm. From Figure 3, we can observe that HCMF

TABLE II

COMPARISON ON THE SYNTHETIC DATA. THERE ARE SIX FUSED CLASSIFIERS. THREE OF THEM HAVE THE ACCURACY OF 80%, AND ANOTHER THREE OF THEM HAVE THE ACCURACY OF 60%

Best-Single	[20]	Weighting	[AVE]	[VOTE]	[GATE]
80.0%	96.8%	97.2%	97.9%	97.8%	**98.2%**

detect the outliers accurately. Furthermore, most outlier entries are corrected to align with the ground-truth predictions, even if there is 50% random noise. By restricting the recovered matrix as low-rank, HCMF can preserve the consistency of all classifiers to generate the final prediction.

3) The Effect of Unbalanced Instances: We set $n = 500$, $\mathcal{C} = 20$, and $m = 6$ to generate the ground-truth label matrix. We first randomly select 50% instances and label them as the first class, and we then randomly assign the classes of 2$\sim$6 for the remaining 50% instances. Therefore, the instances of the first class dominant the whole data. Similar to the above experiment, we create the predictions of $m = 6$ classifiers with the noise ratio of 30%. By using HCMF with GATE, we obtain the average accuracy of 97.0% by ten runs. If we simply randomly assign the ground truth label, we can obtain the average accuracy of 97.2%. The accuracy of the unbalance case is similar to that of the balance case. Therefore, unbalance instances will not substantially affect our HCMF.

4) The Effect of Combining Good and Bad Classifiers: Similar to the experiment in Figure 3, we set $n = 1000$, $\mathcal{C} = 20$, and $m = 6$. We then randomly generate the predictions of three "good" classifiers, where the accuracy of each classifier is 80%. We also randomly generate the predictions of three "bad" classifiers, where the accuracy of each classifier is 60%. The first 500 instances are regarded as the training set, and the last 500 instances are viewed as the testing set. We use the neural network to learn m weight values to do the weighted average on the predictions of m classifiers on the training set. We denote this method as "Weighting" in Table II. The weighting method achieves the accuracy of 97.2%. Our HCMF can effectively utilize the information of bad classifiers and achieve 97.8% by the VOTE post strategy. By using the GATE strategy, HCMF obtains a better accuracy of 98.2%.

5) Comparison With the Smooth Loss: The method of [13] is originally designed for the smooth loss function, e.g., the Frobenius norm. However, to make the results robust (detecting the outliers), we need to use the $\ell_{2,1}$ norm, which is non-smooth. Therefore, [13] cannot handle such non-smooth problem considered in this paper. Instead, we propose to use an ALM-based method to solve the non-smooth $\ell_{2,1}$ norm. To demonstrate the non-smooth $\ell_{2,1}$ norm is superior to the smooth ℓ_2 norm in our problem, we make the following experiment. We replace the $\ell_{2,1}$ norm in Eq. (1) by the ℓ_2 norm, and use the method of [13] to solve the $\min_{\mathbf{X}} ||\mathbf{L} - \mathbf{X}||_2$ with the low-rank constraint. We use [GATE] as the post-processing procedure for [13] and obtain the accuracy of 98.2%. From Table II, [13] only obtains 96.8% accuracy, which is worse than ours.

B. Real-World Experiment Setup

To investigate the performance of HCMF on real-world data and compare with other state-of-the-art algorithms, we perform the experiments on eight publicly available real-world datasets.

We used the best-performing CNN models, including GoogleNet [26], ResNet [27], ResNeXt [28], Pre-ResNet [29], WRN [30], VGG [31], NIN [32], and etc. These features are fairly state-of-the-art. They have been widely used in various recent research papers and have been verified the most effective features by recent internationally recognized competitions such as the ImageNet competition [3].

On the UCF-101 dataset, five different features are extracted, i.e., "fc6" of C3D [33], "pool5/7x7_s1" from GoogleNet, "pool5" of ResNet-152, and two "fc6"s of Two-Stream [16]. Given a video, we extract the above five features following the same process described in [16].

On the CIFAR-10 dataset, we extract fourteen different kinds of CNN features. These features are extracted from the last pooling layer of ResNet-20/32/44/56/110, Pre-ResNet-20/32/44/56/110/164, CIFAR-Full, modified VGG16, and modified GoogleNet models, respectively.

On the CIFAR-100 dataset, we use the last pooling layer features of the ResNet-20/32/44/56/68/110 and the Pre-ResNet-20/32/44/56/110/164 models (twelve features in total).

For other datasets, i.e., Oxford-IIIT-Pet, PASCAL VOC 2007, Oxford Flower 17, Pascal Sentence and Wikipedia, we extract eight different kinds of features to train SVM classifiers. Four of them are from the "fc6" layer of AlexNet, VGG16/19, CaffeNet [3]. One of them is from the "pool5/7x7_s1" layer of GoogleNet. Three of them are from the "pool5" layer of the ResNet-50/101/152 models.

The code and model configuration are public available at GitHub https://github.com/D-X-Y/HCMF.

C. Comparison With Late Fusion Methods

We compare HCMF to five late fusion methods as follows:

Average Score Fusion (ASF): we directly average the results from multiple classifiers, then the L_2 norm is applied on each classifier's results for normalization.

Multiple Kernel Learning (MKL): MKL learns a weight coefficient w for each classifier, and the final scores are obtained from function $f(s) = w^T s, \ \sum w = 1$.

Robust Convex Ensemble Clustering (RCEC) [12]: We apply RCEC to recover $\mathbf{X}$ from L in Eq. (2). RCEC employs the normalize cuts method to generate final clustering results. In this paper, we instead use the introduced three post strategies to generate the fusion labels for a fair comparison.

Feature Weighting via Optimal Thresholding (FWOT) [6]: they propose to learn thresholding, smoothing parameters and weights in a joint framework to combine multiple prediction results.

LPBoost [4]: A variant linear combination is applied to multiple classifiers to boost performance.

1) Experiment Settings: The parameter μ is selected from $\{0.1, 1, 5\}$, ρ is select from $\{1.01, 1.05, 1.1\}$ and ϵ is select from $\{0.01, 0.001, 0.0001\}$ in all our experiments. Ω is always the full indexes in our experiments. Among different

TABLE III

THE MEAN ACCURACY ON REAL-WORLD DATASETS. FWOT COSTS EXCESSIVE TIME FOR TRAINING UCF-101, CIFAR-10 , CIFAR-100 (OVER 24 HOURS), AND WE THUS OMIT THESE RESULTS INDICATED BY "-". WE USE THE GATE STRATEGY FOR THE POST PROCESS

Method	UCF-101	CIFAR-10	CIFAR-100	Oxford-IIIT-Pet	PASCAL VOC 2007
ASF	86.78%	94.21%	76.91%	92.46%	90.51%
MKL	85.38%	94.15%	72.71%	87.13%	85.82%
RCEC	86.60%	94.99%	77.13%	92.11%	90.92%
FWOT	-	-	-	**92.90%**	91.20%
LPBoost	86.24%	94.88%	76.59%	92.73%	**91.42%**
HCMF	**89.03%**	**95.11%**	**77.72%**	**92.90%**	90.98%

TABLE IV

COMPUTATIONAL TIME ON REAL-WORLD DATASETS. COMPUTATIONAL TIME IS RECORDED IN SECONDS. FWOT COSTS EXCESSIVE TRAINING TIME ON SOME DATASETS (OVER 24 HOURS), SO WE OMIT THESE RESULTS INDICATED BY "∞". MKL IS AN EARLY FUSION METHOD, AND IT THUS CANNOT BE DIRECTLY COMPARED WITH LATE FUSION METHOD REGARDING THE COMPUTATIONAL TIME

Method	UCF-101	CIFAR-10	CIFAR-100	Oxford-IIIT-Pet	PASCAL VOC 2007
ASF	0.01	0.01	0.01	0.01	0.01
RCEC	86.60	81.41	255.51	18.82	16.45
FWOT	∞	∞	∞	61697.45	19674.01
LPBoost	110.81	72.45	4750.90	43.67	7.46
HCMF	421.23	5.384	417.26	76.16	15.21

parameters, there are only small differences in the prediction performance. LIBLINEAR [34] is adopted for our basic classification toolbox. The parameters are set as the default in LIBLINEAR and liblinear-mkl, unless otherwise specified. For RCEC method, we use 5-fold cross validation to select the best parameter β from $\{0.01, 1, 2, 4, 6, ..., 20\}$ and set $\lambda = 0.1, \gamma = 0.01$, which are suggested by Yi *et al.* [35]. For FWOT, the smoothing parameter candidates are $\{0.5, 0.6, ..., 0.9\}$ and the parameter C in is selected from $\{10^{-4}, 10^{-2}, ..., 10^{4}\}$ according to cross-validation, which is same in [6]. For LPBoost, we use cross-validation to pick up the best ν from $\{0.5, 0.6, 0.7, 0.8, 0.9\}$ suggested in [6].

2) Datasets: We use five datasets for the late fusion comparison. UCF-101 is commonly used for action recognition, which has 9573 training and 3783 testing videos, categorized into 101 human action classes. CIFAR-10 contains 60000 pictures with 10 classes. CIFAR-100 similar with CIFAR-10, while has 100 classes. Oxford-IIIT-Pet contains 7349 images covering 37 different breeds of cats and dogs. PASCAL VOC 2007 is a dataset for object detection and image classification. We only use the image with a single class, which includes 6,146 images. For UCF 101, we use the official split-1 set for training and testing. For Oxford-IIIT-Pet and PASCAL VOC 2007, we randomly sample fifty percent pictures as our training data and the others as testing data.

3) Quantitative Comparison: Table III shows the mean accuracy results for the five datasets, and our proposed algorithm provides a significant improvement compared with other effective late fusion methods on most datasets. Additionally, our algorithm is more efficient than other late fusion methods in most situations. As we can see from Table IV, some methods are not capable of dealing with large-scale datasets. For example, FWOT is too slow to train; thus we do not show its result for UCF-101 and CIFAR (more than one day). The time consumption of ours is acceptable in all experiments and quite fast on CIFAR-10/VOC2007. Combined with the fusion performance listed in Table III, we would like to show that our method achieves superior performance with acceptable computational time. Our method and RCEC have the similar range of running time. Nevertheless, our method outperforms RCEC regarding accuracy for all the five datasets. In conclusion, our method can handle the large-scale problem efficiently and yield stable performance.

4) Sensitivity to the Number of Testing Data n: We investigate the sensitivity of our method to the number of the testing data. We test our method on 1K, 2K, 4K, 6K, 8K, and 10K instances as the input of late fusion. We use the same experiment settings as described above. Experiments with these six different numbers of instance obtain similar results, where the variance is less than 0.1%. We observe that our method is not sensitive to the number of instances. Experiments demonstrate the robustness and stability of our method.

5) Discussion: To be noticed, we use the SVM predictions rather than the direct CNN outputs for the experiments in Table III and Table IV, because FWOT and MKL can only leverage SVM predictions. Therefore, by using the SVM prediction, we can investigate the fusion performance of ours and five compared algorithms by a fair comparison. We can usually obtain superior fusion results by taking the CNN predictions as inputs compared to SVM predictions. We will compare results that use the CNN predictions later.

D. Comparison With Feature Fusion Methods

Feature fusion aims to combine the different classifiers' predictions by the feature-level fusion. To fuse multiple models in the feature level, one should load these models in GPU instead of CPU to improve the efficiency. For most current GPU devices, there is only less than 12 GB memory. Given such limited memory, it might be unaffordable to run CNN-based feature fusion for many different models.

TABLE V

THE MEAN ACCURACY OF FEATURE FUSION METHODS ON CIFAR-10

Bayesian	Pooling Layer	Prob Layer	RCEC	**HCMF**
94.10%	95.21%	95.32%	95.56%	**95.93%**

Comparison and Discussion: We compare the following three feature fusion methods, i.e., global-pooling feature fusion and probability prediction fusion. We choose the global-pooling layer feature of all residual-based networks used in Table III. For these two feature fusion methods, we directly concatenate the global-pooling layers (and the probability prediction layers), and then use a fully connected layer with 10 output to combine the concatenation feature to the final prediction for CIFAR-10. For our fusion algorithm, we use the CNN predictions rather than the SVM predictions used in Table III. In Table V, HCMF achieves the best performance among these feature fusion methods. Besides, we also compare ours with RCEC by using the CNN predictions as inputs, where we achieve about 9% relative accuracy improvement.

E. Comparison With State-of-the-Art Single Models

This experiment section investigates how much the proposed method improves the performance, compared with the state-of-the-art single models on CIFAR-10 and CIFAR-100. In this section, we directly fuse multiple state-of-the-art CNN predictions, instead of SVM predictions in the previous section, to have a fair comparison with those CNN models.

1) Model Details: We summarize the fused models in Table VI. Table VII reports the test errors of the baselines and our methods. For our method, we have two fusion approaches (with/without ResNeXt-29-16x64). The first one fuses all the models, the other one does not use the ResNeXt-29-16x64d.

2) Analysis: We carefully cite the best performances reported in their papers. As can be observed, our fusion algorithm is capable of further improving the performance of single models significantly. We emphasize that ResNeXt is a very superior performer in single models. HCMF (w/o ResNeXt) fuses three kinds of models which are all worse than ResNeXt, but outperforms ResNeXt. When we include ResNeXt to our fusion algorithm, our method further decreases the test error. This experiment demonstrates the benefit of our late fusion method. Moreover, a single model has the satisfactory performance with deeper depth or wider channel. However, we can still obtain the performance improvement by leveraging the fusion algorithm when a single model achieves its limits.

3) Test the Robustness of Hcmf: To Test The Robustness Of HCMF, we include one bad classifier into the fused models. We design a simplified ResNet. It removes one convolution layer in the basic residual block. Compared to ResNet-20, there is only one simplified residual block in each residual stage. Since this simplified ResNet only have five layers, we name it as S-ResNet-5 and use it as the bad classifier. By using S-ResNet-5, we achieve 21.18% error on CIFAR-10 and 51.67% error on CIFAR-100. The results of S-ResNet-5 are much worse than the results shown in Table VII. By including S-ResNet-5 into "HCMF (w ResNeXt)", we obtain the error of 3.21 on CIFAR-10 and 16.93 on CIFAR-100. The performance of including S-ResNet-5 is similar to not including that, and HCMF is thus robust to bad classifiers.

TABLE VI

MODELS USED ON CIFAR-10 AND CIFAR-100. "*" REPRESENTS USING THE BOTTLENECK BLOCK, OTHERWISE WE USE THE BASIC BLOCK

	CIFAR-10	CIFAR-100
ResNet	110, 134, 164 164*, 200*	56, 110, 164 110*, 164*
WRN	16-10, 22-8, 28-10 40-4, 40-8	16-10, 22-8, 28-10 40-4
NIN	9	9
ResNeXt	29-16x64d	29-16x64d

TABLE VII

THE TEST ERRORS ON CIFAR-10 AND CIFAR-100

Methods	CIFAR-10	CIFAR-100
Pre-ResNet (1001 layers)	4.92	22.71
WRN-28-10	4.17	20.50
Dual-Path-Network [44]	3.65	-
ResNeXt-29-16x64d	3.58	17.31
HCMF (w/o ResNeXt)	3.47	17.24
HCMF (w ResNeXt)	**3.19**	**16.91**

TABLE VIII

THE MEAN ACCURACY USING DIFFERENT POST PROCESS STRATEGIES

Strategy	Wikipedia	Flowers 17	Pascal Sentence
[AVE]	47.78%	**96.77%**	66.40%
[VOTE]	47.85%	96.61%	66.20%
[GATE]	**49.03%**	**96.77%**	**67.20%**

F. Post Strategy Selection

In this section, we compare three post strategies, described in Sec.III-F. We additionally use three different datasets, i.e., Oxford Flower 17 [36], Pascal Sentence [37] and Wikipedia [38]. Wikipedia contains paragraphs of text for each picture, and there are 2,866 image-text pairs categorized into 10 class. Oxford Flowers 17 has 17 flower classes with 80 images for each class. Pascal Sentence contains 1,000 images annotated by caption information and categorized into 20 classes. We use the same experiment settings as the experiments in Table III. 50% of data are sampled as the training set and the others as the testing set.

Table VIII lists the mean accuracy of three post processing strategies on three datasets. The mean accuracy of the *GATE* strategy is the highest among all three competing strategies. The outlier detection of HCMF might cause this improvement, since the recovered matrix provides more consistency benefiting the confidence gate operation.

VI. CONCLUSION

Late fusion aims to improve the prediction performance by taking advantages of multiple features. In practice, the controversy over different features on prediction is ubiquitous and may degenerate the performance. To deal with this issue, we propose HCMF, a late fusion algorithm, which formulates this task as a robust matrix recovery problem with a hard constraint on the matrix rank to preserve the consistency among various features. Nevertheless, most MF literature only focus on smooth loss, making it unclear how to extend them to handle non-smooth robust loss, e.g., $\ell_{2,1}$ loss. We thus propose an ALM-based algorithm to cope with the non-smooth loss and provide theoretical convergence guarantee. Moreover, we demonstrate the hard constraint is more suitable than the soft constraint when the consistency across all base learners of the final predictions is required. Empirical studies on the large-scale real-world datasets demonstrate the improvement of our method compared to the state-of-the-art fusion methods.

REFERENCES

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, pp. 886–893.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[4] P. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sep. 2009, pp. 221–228.

[5] G. Ye, D. Liu, I.-H. Jhuo, and S.-F. Chang, "Robust late fusion with rank minimization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3021–3028.

[6] Z. Xu, Y. Yang, I. Tsang, N. Sebe, and A. G. Hauptmann, "Feature weighting via optimal thresholding for video analysis," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 3440–3447.

[7] K.-T. Lai, D. Liu, S.-F. Chang, and M.-S. Chen, "Learning sample specific weights for late fusion," *IEEE Trans. Image Process.*, vol. 24, no. 9, pp. 2772–2783, Sep. 2015.

[8] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1933–1941.

[9] C. Q. Gao, D. Meng, Y. Yang, Y. Wang, X. Zhou, and A. G. Hauptmann, "Infrared patch-image model for small target detection in a single image," *IEEE Trans. Image Process.*, vol. 22, no. 12, pp. 4996–5009, Dec. 2013.

[10] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, Jan. 2004.

[11] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *J. Mach. Learn. Res.*, vol. 9, pp. 2491–2521, Nov. 2008.

[12] J. Gao, M. Yamada, S. Kaski, H. Mamitsuka, and S. Zhu, "A robust convex formulation for ensemble clustering," in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, pp. 1476–1482.

[13] B. Vandereycken, "Low-rank matrix completion by Riemannian optimization," *SIAM J. Optim.*, vol. 23, pp. 1214–1236, Jun. 2013.

[14] N. Boumal and P.-A. Absil, "RTRMC: A Riemannian trust-region method for low-rank matrix completion," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 406–414.

[15] T. Ngo and Y. Saad, "Scaled gradients on Grassmann manifolds for matrix completion," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1412–1420.

[16] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 568–576.

[17] X. Dong, L. Zheng, F. Ma, Y. Yang, and D. Meng, "Few-example object detection with model communication," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published, doi: 10.1109/TPAMI.2018.2844853.

[18] Z. Ma, X. Chang, Y. Yang, N. Sebe, and A. G. Hauptmann, "The many shades of negativity," *IEEE Trans. Multimedia*, vol. 19, no. 7, pp. 1558–1568, Jul. 2017.

[19] K. R. Jerripothula, J. Cai, and J. Yuan, "Image co-segmentation via saliency co-fusion," *IEEE Trans. Multimedia*, vol. 18, no. 9, pp. 1896–1909, Sep. 2016.

[20] Y. Yan, F. Nie, W. Li, C. Gao, Y. Yang, and D. Xu, "Image classification by cross-media active learning with privileged information," *IEEE Trans. Multimedia*, vol. 18, no. 12, pp. 2494–2502, Dec. 2016.

[21] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang, "Image clustering using local discriminant models and global integration," *IEEE Trans. Image Process.*, vol. 19, no. 10, pp. 2761–2773, Oct. 2010.

[22] L. Zheng, S. Wang, J. Wang, and Q. Tian, "Accurate image search with multi-scale contextual evidences," *Int. J. Comput. Vis.*, vol. 120, no. 1, pp. 1–13, 2016.

[23] L. Zheng, Y. Yang, and Q. Tian, "Sift meets CNN: A decade survey of instance retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 5, pp. 1224–1244, May 2018.

[24] X. Shijie, L. Wen, X. Dong, and T. Dacheng, "FaLRR: A fast low rank representation solver," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 763–773.

[25] Z. Lin, M. Chen, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," UIUC, Champaign, IL, USA, Tech. Rep. #UILU-ENG-09-2215, 2010.

[26] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[28] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2017, pp. 5987–5995.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 630–645.

[30] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf.*, 2016, pp. 87.1–87.12.

[31] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *Proc. Brit. Mach. Vis. Conf.*, 2014.

[32] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proc. Int. Conf. Learn. Represent.*, 2014, doi: 10.5244/C.28.6.

[33] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 4489–4497.

[34] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.

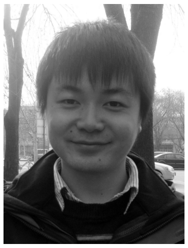
[35] J. Yi, T. Yang, R. Jin, A. K. Jain, and M. Mahdavi, "Robust ensemble clustering by matrix completion," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2012, pp. 1176–1181.

[36] M.-E. Nilsback and A. Zisserman, "A visual vocabulary for flower classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2006, pp. 1447–1454.

[37] C. Rashtchian, P. Young, M. Hodosh, and J. Hockenmaier, "Collecting image annotations using Amazon's mechanical turk," in *Proc. NAACL HLT Workshop Creating Speech Lang. Data Amazon's Mech. Turk*, 2010, pp. 139–147.

[38] N. Rasiwasia *et al.*, "A new approach to cross-modal multimedia retrieval," in *Proc. ACM Multimedia Conf.*, 2010, pp. 251–260.

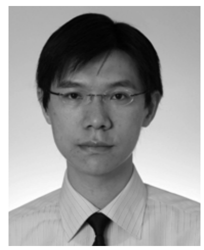
Xuanyi Dong received the B.Eng. degree in computer science and technology from Beihang University, Beijing, China, in 2016. He is currently pursuing the Ph.D. degree with the Center of Artificial Intelligence, University of Technology Sydney, Australia. His research interests include deep learning and its applications to computer vision.



Yan Yan received the B.E. degree in computer science from Tianjin University, Tianjin, China, in 2013. He is currently pursuing the Ph.D. degree with the Centre for Artificial Intelligence, University of Technology Sydney, Australia. His current research interest includes machine learning and computer vision.

Mingkui Tan received the bachelor's degree in environmental science and engineering and the master's degree in control science and engineering from Hunan University, Changsha, China, in 2006 and 2009, respectively, and the Ph.D. degree in computer science from Nanyang Technological University, Singapore, in 2014. From 2014 to 2016, he was a Senior Research Associate in computer vision with the School of Computer Science, The University of Adelaide, Australia. Since 2016, he has been with the School of Software Engineering, South China University of Technology, China, where he is currently a Professor. His research interests include machine learning, sparse analysis, deep learning, and large-scale optimization.

Yi Yang received the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2010. He was a Post-Doctoral Research with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. He is currently a Professor with the University of Technology Sydney, Australia. His current research interest includes machine learning and its applications to multimedia content analysis and computer vision.



Ivor W. Tsang received the Ph.D. degree in computer science from The Hong Kong University of Science and Technology in 2007. He is currently an ARC Future Fellow and a Professor with the University of Technology Sydney. He is also the Research Director of the UTS Priority Research Centre for Artificial Intelligence. In 2009, he was conferred the 2008 Natural Science Award (Class II) by the Ministry of Education, China. He received the prestigious IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding 2004 Paper Award in 2007, the 2014 IEEE TRANSACTIONS ON MULTIMEDIA PRIZE PAPER AWARD, and the Best Student Paper Award at CVPR 2010.