Scalable Algorithms for Multi-Instance Learning

Xiu-Shen Wei, Jianxin Wu, *Member, IEEE,* and Zhi-Hua Zhou, *Fellow, IEEE*

***Abstract*—Multi-instance learning (MIL) has been widely applied to diverse applications involving complicated data objects such as images and genes. However, most existing MIL algorithms can only handle small- or moderate-sized data. In order to deal with large scale MIL problems, we propose miVLAD and miFV, two efficient and scalable MIL algorithms. They map the original MIL bags into new vector representations using their corresponding mapping functions. The new feature representations keep essential bag-level information, and at the same time lead to excellent MIL performances even when linear classifiers are used. Thanks to the low computational cost in the mapping step and the scalability of linear classifiers, miVLAD and miFV can handle large scale MIL data efficiently and effectively. Experiments show that miVLAD and miFV not only achieve comparable accuracy rates with state-of-the-art MIL algorithms, but also have hundreds of times faster speed. Moreover, we can regard the new miVLAD and miFV representations as multi-view data, which improves the accuracy rates in most cases. In addition, our algorithms perform well even when they are used without parameter tuning (i.e., adopting the default parameters), which is convenient for practical MIL applications.**

***Index Terms*—Multi-instance learning, Large scale data, Scalability, Efficiency**

I. Introduction

THe multi-instance learning (MIL) framework was proposed during the investigation of drug activity prediction [1], and was naturally applied to this problem. Different from traditional single-instance learning, the input of MIL are a set of bags that are labeled positive or negative, instead of receiving a set of instances who have positive or negative labels. In particular, the instances in MIL have no label information associated with them. The MIL objective is to train a classifier which can label new bags. Multi-instance learning has been widely applied in various applications, such as image categorization or retrieval, face detection, text categorization, computer-aided medical diagnosis, etc. [2].

Over the past years, many effective MIL algorithms [3], [4], [5], [6], [7], [8] have been developed to solve the difficulty caused by the missing of instance labels. For example, miSVM [3] is an SVM based method, which aims at explicitly assigning labels to instances in bags by searching for the max-margin hyperplanes to separate instances. The MIBoosting method [5] assumes that every instance contributes independently and equally to label of the bag which contains it, and proposes a boosting approach to solve the MIL problem. miGraph [6] treats the instances as non-i.i.d. and maps a bag to an undirected graph to solve MIL.

In the classic bag representation of MIL, an object is represented as a set of instances, which can naturally encode the original complex objects. This fact might partly explain the success of existing MIL algorithms, which have achieved decent accuracy rates in different MIL applications. However, the bag representation is complicated, and directly processing and classifying them means that MIL's hypothesis space will become much larger and more complex. This fact leads to an undesired outcome: most MIL algorithms are usually time-consuming and incapable of handling large scale problems. In the real world, however, applications of MIL consistently request scalable learning algorithms to handle millions of complex objects or examples (e.g., images, genes, etc.)

In order to handle large scale MIL datasets, one natural way is to convert the representation of an object from a bag to a simpler one, i.e., a vector. The conversion should be very efficient and it should keep as much information from the bag as possible, in order to achieve an efficient, scalable and accurate multi-instance learning machine. Along this line of research, some methods have been proposed, including the MILES algorithm [7] and the CCE method [9]. However, as we will empirically show in this paper, neither the efficiency nor the accuracy of these methods is mature enough to handle large scale multi-instance learning datasets.

In this paper, we propose two efficient, scalable and accurate MIL algorithms: miVLAD (MIL based on the VLAD representation) and miFV (MIL based on the Fisher Vector representation), respectively. In both methods, bags are mapped by their corresponding mapping functions into new feature vector representations. The major difference between miVLAD/miFV and CCE/MILES is that the proposed methods encode more information into the new vector representation. In addition, they are efficient to compute, and both lead to excellent results using linear classifiers.

Thanks to the low computational cost in the mapping step and the scalability of linear classifiers, miVLAD and miFV can handle large scale MIL problems efficiently. Our experimental results show that on small-scale and medium-scale MIL datasets, both proposed methods achieve comparable accuracy with state-of-the-art MIL algorithms, e.g., MIBoosting [5] and miGraph [6], and outperform other widely used MIL algorithms, e.g., miSVM [3] and EM-DD [4]. It is worth mentioning that miVLAD and miFV have hundreds of, or even up to thousands of times faster speed than these compared MIL algorithms. What's more, on large scale problems the learning of most existing MIL algorithms did not terminate after a few days, while both miVLAD and miFV can finish training within a few hours and achieve good classification accuracy.

This research was supported by the National Natural Science Foundation of China under Grant 61422203 and Grant 61333014, the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Tencent fund.

The authors would like to thank Dan Zhang for supplying the WebKB datasets, Jaume Amores for supplying the Speaker dataset, and thank the anonymous reviewers, whose comments have helped improving this paper.

All authors are with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China. J. Wu is the corresponding author. E-mail: {weixs,wujx,zhouzh}@lamda.nju.edu.cn.

In addition, many problems in machine learning involve data sets with multiple views where observations are represented by multiple sources of features. Similarly, miVLAD and miFV could provide different and complementary information for MIL bags. As shown in our experiments, by treating the new representations of miVLAD and miFV as multi-view data [10], we can improve the classification performance on most data sets. Meanwhile, our proposed algorithms are not sensitive to their corresponding parameters. Even with the default parameters, they can achieve satisfactory performances too.

We organize the rest of this paper as follows. Related works on MIL are introduced in Section II. The proposed miVLAD and miFV algorithms are presented in Section III. We present our experimental results in Section IV. In Section V, we discuss the relationship between the two proposed algorithms, and the relationship between several MIL algorithms and miVLAD/miFV. Finally, discussions on future issues are provided in Section VI to conclude this paper. Preliminary studies on miFV was published in [11].

II. Related Work

During the past decade, many multi-instance learning algorithms have been developed in the literature. Roughly speaking, we can categorize MIL methods in the literature based on their algorithmic styles, including at least density based methods (Diverse Density [12] and EM-DD [4]), k-nearest neighbor based methods (Citation-kNN and Bayesian-kNN [8]), support vector machine based methods (miSVM and MISVM [3]), ensemble based method (MIBoosting [5]), converting MIL into single instance methods (MIWrapper [13] and MILES [7]), distance based method (MIMEL [14]), dictionary learning based method (GD-MIL [15]), kernel based methods (miGraph [6] and MIKernels [16]) and so on [17], [18], [19], [20], [21]. Existing MIL algorithms have achieved satisfactory accuracy rates. However, most of them can only handle small- or moderate-sized data, and only have limited scalability. To the best of our knowledge, existing MIL algorithms cannot deal with large scale MIL data efficiently.

In order to deal with MIL problems more efficiently, some researchers had tried to solve MIL by using the similarity that represents the closeness of the bag to specific target points in the original instance space, e.g., DD-SVM [17] and MILES [7]. Some had tried to treat MIL as a special case of semi-supervised learning, e.g., MissSVM [19]. However, due to the bag representation of MIL, the complexity of the hypothesis space of existing MIL algorithms is still too large to learn very efficiently. And, handling large scale MIL data is not practical yet. For example, the MILES method took 1.2 minutes (i.e., 72 seconds) to learn on the Musk2 data set, which is a small scale problem with only 102 bags [7].[1]

CCE (Constructive Clustering based Ensemble) is a more efficient MIL algorithm [9]. This method could solve MIL problems with lower computational complexity, because it uses a vector representation that is based on constructive clustering. However, as will be discussed in detail in Section V-C, the information CCE extracts from bags is very limited comparing to miVLAD or miFV, which makes its accuracy significantly worse than that of the proposed methods. Another disadvantage of CCE is that it uses an ensemble of classifiers based on multiple clustering results, in order to incorporate more information from the bag representation. This step significantly increases its computational complexity and renders it incapable of handling large scale problems.

In addition, much progress of multi-view learning [10] has been made recently, e.g., [22], [23], [24], [25], [26], [27], [28]. Moreover, the proposed MIL algorithms can provide complementary information of two different views for multi-instance bags. By utilizing different information from miVLAD and miFV as multi-view learning, it is not only efficient and scalable, but also can achieve better classification performance in most cases than single miVLAD/miFV, which is validated in our experiments.

III. The Proposed miFV and miVLAD Algorithms

In this paper, we propose two accurate, efficient and scalable multi-instance learning algorithms, named as miVLAD and miFV. Both methods convert a bag into a vector very efficiently, and keep useful information inside the bag.

The VLAD (Vector of Locally Aggregated Descriptors) representation [29] and the FV (Fisher Vector) representation [30] are two approaches in computer vision. Given an image and a set of descriptors (vectors) extracted inside this image (e.g., SIFT), VLAD or FV encodes them into a high dimensional vector, which is the new image-level signature. After that, we feed these image-level representations into Support Vector Machine (SVM), e.g., [31], [32], to train classifiers. Because of their efficiency and effectiveness, they become state-of-the-art approaches in computer vision and have shown excellent performances in many applications. For example, the applications of FV in large scale image retrieval [33] and image categorization [30], and the applications of VLAD in image classification [29] and action recognition [34], all readily show the efficiency, effectiveness and scalability of VLAD and FV. Moreover, recently some studies have shown their effectiveness and efficiency both theoretically and practically [30].

In the VLAD and FV representation of images, an image is represented as a set of local patches. This set representation is related to, but different from the bag representation in MIL. In a bag in MIL, every instance has a label (although unknown during the learning process), indicating whether it is a positive or negative instance. In a set representation of an image, however, the local patches do not have semantic meanings. For example, in an image labeled as *tall building*, none of the local patches extracted from it can be treated as a "tall building". Instead, they might be a window of a building, partial walls of it, sky, part of a tree, etc. It is difficult to assign either a positive or negative label to one image patch, even by a human. Since both VLAD and FV map a set of items into one single vector, we, however, may borrow ideas from them to implement scalable multi-instance learning.

Before presenting the details of the proposed methods, we first give a formal definition of multi-instance learning

[1]As a comparison, both proposed methods use less than 0.1 second on this data set, cf. Section IV.

as the following. Let $\mathcal{X}$ denote the instance space. Given a data set $\{(X_1, y_1), \ldots, (X_i, y_i), \ldots, (X_{N_B}, y_{N_B})\}$, where $X_i = \{\boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{ij}, \ldots, \boldsymbol{x}_{i,n_i}\}$ is called a bag and $y_i \in \mathcal{Y} = \{-1, +1\}$ is the label of X_i, the goal is to generate a classifier to classify unseen bags. Here $\boldsymbol{x}_{ij} \in \mathcal{X}$ is an instance $[x_{ij1}, \ldots, x_{ijl}, \ldots, x_{ijd}]^T$, x_{ijl} is the value of $\boldsymbol{x}_{ij}$ at the l-th attribute, N_B is the number of training bags, n_i is the number of instances in the corresponding bag X_i, and d is the number of attributes.

In addition, we denote $N_I = \sum_{i=1}^{N_B} n_i$ as the total number of instances in all bags, $\boldsymbol{x}_{i\cdot}$ as all the instances in the bag X_i, and $\boldsymbol{x}_{\cdot j}$ as all instances from all bags, respectively. Note that, in this paper, we treat multi-instance learning with the standard assumption. A bag is labeled positive if it contains at least one positive instance, otherwise it is labeled as a negative bag. Yet, the labels of instances are unknown in both the training and the testing data.

A. miVLAD

Now we describe the miVLAD (multi-instance learning based on the VLAD representation) algorithm. Pseudo code of the miVLAD algorithm is presented in Algorithm 1.

Algorithm 1 The miVLAD algorithm

```
1:  Input:
2:    Training data $\{(X_1, y_1), \ldots, (X_{N_B}, y_{N_B})\}$
3:  Train:
4:    Learn a codebook $\mathcal{C} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K\}$ based on the set
      of all instances $\boldsymbol{x}_{\cdot j}$, from all training bags
5:    for $i$ = 1 to $N_B$ do
6:      Get the new feature vector via the mapping function
        $\boldsymbol{v}_i \leftarrow \mathcal{M}_v(X_i, \mathcal{C})$
7:      $v_{i\cdot l} \leftarrow \mathrm{sign}(v_{i\cdot l})\sqrt{|v_{i\cdot l}|}$
8:      $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i / \|\boldsymbol{v}_i\|_2$
9:    end for
10:   Use the new training set $\{(\boldsymbol{v}_1, y_1), \ldots, (\boldsymbol{v}_{N_B}, y_{N_B})\}$ to
      learn a classifier $\mathcal{F}$
11: Test:
12:   for all test bags $X_{i'}$ $(i' \in \{1, 2, \ldots, N'_B\})$ do
13:     Get the new feature vector via the mapping function
        $\boldsymbol{v}_{i'} \leftarrow \mathcal{M}_v(X_{i'}, \mathcal{C})$
14:     $v_{i'\cdot l} \leftarrow \mathrm{sign}(v_{i'\cdot l})\sqrt{|v_{i'\cdot l}|}$
15:     $\boldsymbol{v}_{i'} \leftarrow \boldsymbol{v}_{i'} / \|\boldsymbol{v}_{i'}\|_2$
16:   end for
17:   Output the prediction $\mathcal{F}(\boldsymbol{v}_{i'})$
```

The first step is to gather all instances $\boldsymbol{x}_{\cdot j}$ from all training bags. Then, we cluster $\boldsymbol{x}_{\cdot j}$ into K centroids, i.e., $\mathcal{C} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k, \ldots, \boldsymbol{c}_K\}$, with the k-means clustering algorithm, and $\mathcal{C}$ is called the codebook. Each instance $\boldsymbol{x}_{ij}$ is assigned to its nearest centroid $\boldsymbol{c}_k = NN(\boldsymbol{x}_{ij})$, where $NN(\boldsymbol{x})$ is the nearest neighbor of $\boldsymbol{x}$ in the codebook $\mathcal{C}$. After that, as presented in Algorithm 2, a mapping function $\mathcal{M}_v$ maps a bag X_i into a feature vector $\boldsymbol{v}_i$ based on the codebook $\mathcal{C}$. For each bag X_i, the mapping function is to accumulate the total differences $\boldsymbol{x}_{i\cdot} - \boldsymbol{c}_k$, where the instances $\boldsymbol{x}_{i\cdot}$ are from the bag X_i and are assigned to $\boldsymbol{c}_k$. Then, for the K differences, the function $\mathcal{M}_v$ concatenates them into a new feature vector $\boldsymbol{v}_i$ to represent the bag X_i.

Algorithm 2 Mapping function $\mathcal{M}_v$ in the miVLAD algorithm

```
1:  Input:
2:    Instances $\{\boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{ij}, \ldots, \boldsymbol{x}_{i,n_i}\}$ in a bag $X_i$
3:    A pre-learned codebook $\mathcal{C} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k, \ldots, \boldsymbol{c}_K\}$
4:  Output:
5:    The feature vector $\boldsymbol{v}_i$ which represents the bag $X_i$
6:  Procedure:
7:    for $k = 1$ to $K$ do
8:      Compute $\boldsymbol{v}_{ik}$ using equation (1)
9:    end for
10:   Concatenate all $K$ components into one vector $\boldsymbol{v}_i$
```

Note that the dimensionality D of $\boldsymbol{v}_i$ is $D = K \times d$. Hence, a component of $\boldsymbol{v}_i$ can be expressed in the following form:

$$v_{ikl} = \sum_{\boldsymbol{x}_{ij} \in \Omega} x_{ijl} - c_{kl} \,, \tag{1}$$

where $\Omega = \{\boldsymbol{x}_{ij} | NN(\boldsymbol{x}_{ij}) = \boldsymbol{c}_k\}$. v_{ikl} represents the l-th attribute of the k-th component of $\boldsymbol{v}_i$, x_{ijl} and c_{kl} denote the l-th attribute of the instance $\boldsymbol{x}_{ij}$ and of its corresponding centroid $\boldsymbol{c}_k$, respectively. In the following, each element of $\boldsymbol{v}_i$ is sign square rooted by $v_{i\cdot l} \leftarrow \mathrm{sign}(v_{i\cdot l})\sqrt{|v_{i\cdot l}|}$ [30]. Then, the new feature vector $\boldsymbol{v}_i$ is subsequently ℓ_2-normalized by $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i / \|\boldsymbol{v}_i\|_2$. Thus, the bags are turned into corresponding feature vectors.

Finally, we feed $(\boldsymbol{v}_i, y_i)$ to a standard supervised learner, e.g., a support vector machine, to learn a classification model $\mathcal{F}$. A bag $X_{i'}$ in the testing set will be firstly mapped into a new feature vector $\boldsymbol{v}_{i'}$ by $\mathcal{M}_v$. Then, we can get the bag-level prediction via $\mathcal{F}(\boldsymbol{v}_{i'})$.

B. miFV

We first give an introduction to the Fisher Kernel, and then propose the miFV (multi-instance learning based on the Fisher Vector representation) algorithm.

1) Basics of the Fisher Kernel: The Fisher Kernel (FV) combines the strengths of generative and discriminative approaches to pattern classification [35].

Let $S = \{\boldsymbol{s}_t, t = 1, \ldots, T\}$ be a sample of T observations $\boldsymbol{s}_t \in \mathcal{S}$. Let p be a probability density function which models the generative process of elements in $\mathcal{S}$ with parameters λ. Then, the sample S can be described by the gradient vector

$$G_\lambda^S = \nabla_\lambda \log p(S|\lambda) \,. \tag{2}$$

Intuitively, the gradient of the log-likelihood describes how the parameters of the generative model p should be modified to better fit the data S. Note that the dimensionality of G_λ^S only depends on the number of parameters in p, rather than on the sample size T. In other words, it transforms sets with different number of elements into fixed length vectors G_λ^S, which is amenable for the mapping function $\mathcal{M}_f$ in the miFV algorithm to map the bags with different numbers of instances into a fixed-length feature vector.

In [35], the Fisher Kernel (FK) was proposed to measure the similarity between two samples S_1 and S_2:

$$\mathcal{K}_{FK}(S_1, S_2) = G_\lambda^{S_1 \prime} F_\lambda^{-1} G_\lambda^{S_2} \,, \tag{3}$$

where F_λ is the Fisher information matrix of p:

$$F_\lambda = E_{\boldsymbol{s}\sim p}[\nabla_\lambda \log p(\boldsymbol{s}|\lambda) \nabla_\lambda \log p(\boldsymbol{s}|\lambda)'] \,. \tag{4}$$

Since F_λ is symmetric and positive definite, it has a Cholesky decomposition $F_\lambda^{-1} = L_\lambda{}' L_\lambda$. The FK in (3) can be rewritten explicitly as a dot-product:

$$\mathcal{K}_{FK}(S_1, S_2) = \boldsymbol{f}_\lambda^{S_1 \prime} \boldsymbol{f}_\lambda^{S_2} \,, \tag{5}$$

where

$$\boldsymbol{f}_\lambda^S = L_\lambda G_\lambda^S = L_\lambda \nabla_\lambda \log p(S|\lambda) \,. \tag{6}$$

The normalized gradient vector presented in (6) is the Fisher Vector (FV). In consequence, using a non-linear kernel machine with the $\mathcal{K}_{FK}$ kernel is equivalent to using a linear kernel machine with the FV (i.e., $\boldsymbol{f}_\lambda^S$) as feature vectors.

2) Representing Bags with Fisher Vectors: Here we treat a bag X_i as a sample S mentioned above. Recalling that, in the traditional MIL assumption, the instances in bags are *independently and identically distributed.* In the same way, the $\boldsymbol{s}_t$'s from $\mathcal{S}$ are generated independently by p. Thus, a natural choice of p is a Gaussian Mixture Model (GMM) as one can approximate with arbitrary precision any continuous distribution with a GMM. We can estimate the parameters of the GMM p on the training bags using Maximum Likelihood Estimation (MLE). Pseudo code of the miFV algorithm is shown in Algorithm 4.

We denote the parameters of the K-component GMM by $\lambda = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, k = 1, \ldots, K\}$, where ω_k, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are respectively the mixture weight, mean vector and covariance matrix of the k-th Gaussian. In what follows, for a bag $X_i = \{\boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{ij}, \ldots, \boldsymbol{x}_{i,n_i}\}$, let $\mathcal{L}(X_i|\lambda) = \log p(X_i|\lambda)$. Because of the independence assumption, we can rewrite this equation as follows:

$$\mathcal{L}(X_i|\lambda) = \sum_{j=1}^{n_i} \log p(\boldsymbol{x}_{ij}|\lambda) \,, \tag{7}$$

and the GMM can be presented as:

$$p(\boldsymbol{x}_{ij}|\lambda) = \sum_{k=1}^{K} \omega_k p_k(\boldsymbol{x}_{ij}|\lambda) \,, \tag{8}$$

where the component p_k denotes the k-th Gaussian:

$$p_k(\boldsymbol{x}_{ij}|\lambda) = \frac{\exp\{-\frac{1}{2}(\boldsymbol{x}_{ij} - \boldsymbol{\mu}_k)' \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_{ij} - \boldsymbol{\mu}_k)\}}{(2\pi)^{D/2} |\boldsymbol{\Sigma}_k|^{1/2}} . \tag{9}$$

The mixture weights are subject to the constraint:

$$\sum_{k=1}^{K} \omega_k = 1, \forall\, k : \omega_k \geq 0 \,, \tag{10}$$

to ensure that $p(\boldsymbol{x}_{ij}|\lambda)$ is a valid distribution. The covariance matrices are assumed to be diagonal and the diagonal entries form a vector $\boldsymbol{\sigma}_k^2$ [36],

Now, we will show in the miFV algorithm how the mapping function $\mathcal{M}_f$ maps a bag into a vector, which is shown as the pseudo code in Algorithm 3. The gradients of a single instance $\boldsymbol{x}_{ij}$ w.r.t. the parameters of the GMM model, $\lambda = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, k = 1, \ldots, K\}$, can be presented as

$$\nabla_{\omega_k} \log p(\boldsymbol{x}_{ij}|\lambda) = \gamma_j(k) - \omega_k \,, \tag{11}$$

$$\nabla_{\boldsymbol{\mu}_k} \log p(\boldsymbol{x}_{ij}|\lambda) = \gamma_j(k) \left(\frac{\boldsymbol{x}_{ij} - \boldsymbol{\mu}_k}{\boldsymbol{\sigma}_k^2} \right) , \tag{12}$$

$$\nabla_{\boldsymbol{\sigma}_k} \log p(\boldsymbol{x}_{ij}|\lambda) = \gamma_j(k) \left[\frac{(\boldsymbol{x}_{ij} - \boldsymbol{\mu}_k)^2}{\boldsymbol{\sigma}_k^3} - \frac{1}{\boldsymbol{\sigma}_k} \right] , \tag{13}$$

where $\gamma_j(k)$ is the soft assignment of $\boldsymbol{x}_{ij}$ to Gaussian k, which is also the probability of $\boldsymbol{x}_{ij}$ to be generated by the k-th Gaussian:

$$\gamma_j(k) = p(k|\boldsymbol{x}_{ij}, \lambda) = \frac{\omega_k p_k(\boldsymbol{x}_{ij}|\lambda)}{\sum_{t=1}^{K} \omega_t p_t(\boldsymbol{x}_{ij}|\lambda)} \,. \tag{14}$$

Note that, the division and exponentiation of vectors should be understood as term-by-term operations in the above equations.

Algorithm 3 Mapping function $\mathcal{M}_f$ in the miFV algorithm

```
1:  Input:
2:    Instances $\{\boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{ij}, \ldots, \boldsymbol{x}_{i,n_i}\}$ in a bag $X_i$
3:    GMM parameters $\lambda = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, k = 1, \ldots, K\}$
4:  Output:
5:    The FV $\boldsymbol{f}_\lambda^{X_i}$ which describes the bag $X_i$
6:  Procedure:
7:    for $j = 1$ to $n_i$ do
8:      Compute $\gamma_j(k)$ using equation (14)
9:    end for
10:   for $k = 1$ to $K$ do
11:     Compute $f_{\omega_k}^{X_i}$ using equation (15)
12:     Compute $\boldsymbol{f}_{\boldsymbol{\mu}_k}^{X_i}$ using equation (16)
13:     Compute $\boldsymbol{f}_{\boldsymbol{\sigma}_k}^{X_i}$ using equation (17)
14:   end for
15:   Concatenate all FV components into one vector $\boldsymbol{f}_\lambda^{X_i}$
```

After obtaining the gradients, the next step is to compute L_λ. The methods in [36] and [30] supply us a closed form of L_λ, which can also be solved efficiently. The normalized gradients are [36] [30]:

$$f_{\omega_k}^{X_i} = \frac{1}{\sqrt{\omega_k}} \sum_{j=1}^{n_i} \left(\gamma_j(k) - \omega_k \right) \,, \tag{15}$$

$$\boldsymbol{f}_{\boldsymbol{\mu}_k}^{X_i} = \frac{1}{\sqrt{\omega_k}} \sum_{j=1}^{n_i} \gamma_j(k) \left(\frac{\boldsymbol{x}_{ij} - \boldsymbol{\mu}_k}{\boldsymbol{\sigma}_k} \right) , \tag{16}$$

$$\boldsymbol{f}_{\boldsymbol{\sigma}_k}^{X_i} = \frac{1}{\sqrt{\omega_k}} \sum_{j=1}^{n_i} \gamma_j(k) \frac{1}{\sqrt{2}} \left[\frac{(\boldsymbol{x}_{ij} - \boldsymbol{\mu}_k)^2}{\boldsymbol{\sigma}_k^2} - 1 \right] . \tag{17}$$

Recall that the dimension of the instances in bag X_i is d. It is easy to see that $f_{\omega_k}^{X_i}$ in (15) is a scalar, while $\boldsymbol{f}_{\boldsymbol{\mu}_k}^{X_i}$ and $\boldsymbol{f}_{\boldsymbol{\sigma}_k}^{X_i}$ are d-dimensional vectors. Thus, the FV $\boldsymbol{f}_\lambda^{X_i}$ which can describe a bag X_i should be concatenated by $f_{\omega_k}^{X_i}$, $\boldsymbol{f}_{\boldsymbol{\mu}_k}^{X_i}$ and $\boldsymbol{f}_{\boldsymbol{\sigma}_k}^{X_i}$, for all $k = 1, \ldots, K$ Gaussian components. Then, $\boldsymbol{f}_\lambda^{X_i}$ is normalized by the square-root and L_2-normalization which is similar to that in miVLAD. Finally, the mapping function $\mathcal{M}_f$ maps the bag X_i into a $(2d+1)K$-dimensional normalized FV, i.e., $\boldsymbol{f}_\lambda^{X_i}$.

The rest training and testing procedures are similar to those in miVLAD, as detailed in Line 12-16 in Algorithm 4.

Algorithm 4 The miFV algorithm

```
1:  Input:
2:    Training data $\{(X_1, y_1), \ldots, (X_{N_B}, y_{N_B})\}$
3:  Train:
4:    Estimate the parameters $\lambda = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ of the GMM
      $p$ on the training bags with MLE
5:    for $i = 1$ to $N_B$ do
6:      Map the bag $X_i$ into a FV $\boldsymbol{f}_\lambda^{X_i} \leftarrow \mathcal{M}_f(X_i, p)$
7:      $[\boldsymbol{f}_\lambda^{X_i}]_j \leftarrow \mathrm{sign}([\boldsymbol{f}_\lambda^{X_i}]_j)\sqrt{|[\boldsymbol{f}_\lambda^{X_i}]_j|}$
8:      $\boldsymbol{f}_\lambda^{X_i} \leftarrow \boldsymbol{f}_\lambda^{X_i} / \|\boldsymbol{f}_\lambda^{X_i}\|_2$
9:    end for
10:   Use the new training set $\{(\boldsymbol{f}_\lambda^{X_1}, y_1), \ldots, (\boldsymbol{f}_\lambda^{X_{N_B}}, y_{N_B})\}$
      to learn a classifier $\mathcal{F}$
11: Test:
12:   for all test bags $X_{i'}$ $(i' \in \{1, 2, \ldots, N'_B\})$ do
13:     Map the bag $X_{i'}$ into a FV $\boldsymbol{f}_\lambda^{X_{i'}} \leftarrow \mathcal{M}_f(X_{i'}, p)$
14:     $[\boldsymbol{f}_\lambda^{X_{i'}}]_j \leftarrow \mathrm{sign}([\boldsymbol{f}_\lambda^{X_{i'}}]_j)\sqrt{|[\boldsymbol{f}_\lambda^{X_{i'}}]_j|}$
15:     $\boldsymbol{f}_\lambda^{X_{i'}} \leftarrow \boldsymbol{f}_\lambda^{X_{i'}} / \|\boldsymbol{f}_\lambda^{X_{i'}}\|_2$
16:   end for
17:   Output the prediction $\mathcal{F}(\boldsymbol{f}_\lambda^{X_{i'}})$
```

C. Efficiency and Scalability of miVLAD and miFV

As aforementioned, FV is efficient to compute. In addition, we do not need to use costly nonlinear kernels to map these very high-dimensional gradient vectors implicitly into a still higher dimensional space [36], [30]. Hence, it leads to excellent accuracy rates even with linear classifiers. Thus, miFV is efficient to compute. Even for problems with huge number of instances and feature dimensions, linear SVM and SGD (stochastic gradient descent) methods can solve them very efficiently. In consequence, both miVLAD and miFV can easily handle large scale MIL problems, e.g., the video annotation task which is shown in our experiments.

For miVLAD, because the VLAD representation can be viewed as a simplification of the FV representation [29], miVLAD is even more efficient than miFV. In addition, we can directly concatenate the VLAD and FV representation of a bag to form a new representation. This new representation can be viewed as formed by two channels, i.e., VLAD and FV, which can be treated as multi-view data [10]. The rest training and testing procedures are similar to those in miVLAD/miFV. We denote this method as *miV&F*. Because VLAD and FV are both efficient to obtain, miV&F is also very efficient and scalable. Meanwhile it can achieve better accuracy rates in most cases than miVLAD/miFV.

Regarding the computational cost of the proposed methods, it is dominated by two parts: one part is building the codebook $\mathcal{C}$, and the other part is the mapping process, i.e., $\mathcal{M}_v$ or $\mathcal{M}_f$. For the first one, the typical cost to cluster N_I instances in training bags into K centroids $\mathcal{C}$ is $\mathcal{O}(N_I \times K \times d \times n)$, where n is the number of iterations for k-means in miVLAD or GMM in miFV. After that, the second part will map each bag into a new representation. As detailed in Algorithm 1 and 4, for every bag X_i, we compute the distance from every instance in X_i to every centroid in the codebook, which is the same as computing the distance from each instance in training bags to each centroid in $\mathcal{C}$. Therefore, the cost of that is $\mathcal{O}(N_I \times K \times d)$. In short, the computational cost of the proposed algorithms is $\mathcal{O}((n+1) \times N_I \times K \times d)$ for training (excluding the SVM learning cost). However, because it is difficult to make formal complexity analysis of other MIL algorithms, we empirically validate the efficiency and scalability of our proposed methods in the experiments part.

D. Practical Issues

Principal component analysis (PCA) is a preprocessing technique that uses orthogonal projections to convert a long vector into a shorter one, whose components are linearly uncorrelated with each other. PCA is useful in reducing computational complexities induced by the long vector representations, and sometimes can increase the system accuracy by removing noise contained in the original vectors. In practice, before using the mapping functions in miVLAD and miFV, we can run PCA to reduce noise within the original instances in bags, or reduce their dimensionality. Similarly, PCA is also essential for the success of VLAD and Fisher Vector in practical applications [30].

Hence, there are only two parameters in miVLAD and miFV, i.e., the *number of centers* and the *PCA energy*. The number of centers represents the number of centroids in miVLAD and the number of Gaussian components in miFV, respectively. The PCA energy parameter reflects how much information is left after using PCA. For example, when PCA energy is 1.0, that means we do not use PCA; when PCA energy is 0.9, that means the remaining feature can reconstruct the original one with 90% variations retained. The results of parameter analysis will be shown in the experiments section, which illustrates that miVLAD and miFV can achieve satisfactory results on different data sets even with the corresponding default parameters.

IV. Experiments

In this section, we first describe the experimental setup and the data sets used in our experiments. Then we present the experimental results mainly in four aspects, i.e., *accuracy comparison*, *efficiency comparison*, *scalability* and *parameter analysis*.

A. Data Sets and Experimental Setup

Experiments are performed on five MIL benchmark data sets, four moderate-sized data sets for image classification and document classification, and finally three large scale data sets for the audio classification task, the text categorization task and the video annotation task. On these data sets, we compare the proposed miVLAD and miFV algorithms with six state-of-the-art MIL algorithms: *MIWrapper* [13], *CCE* [9],

Table I
COREL IMAGES.

ID	Category Name	ID	Category Name
1	African people and villages	11	Dogs
2	Beach	12	Lizards
3	Historical building	13	Fashion models
4	Buses	14	Sunset scenes
5	Dinosaurs	15	Cars
6	Elephants	16	Waterfalls
7	Flowers	17	Antique furniture
8	Horses	18	Battle ships
9	Mountains and glaciers	19	Skiing
10	Food	20	Deserts

EM-DD [4], *miSVM* [3], *MIBoosting* [5] and *miGraph* [6]. Empirical results of miV&F are also reported. For miVLAD, miFV and miV&F, we take LIBLINEAR [31] as the final linear classifier presented in Line 10 in Algorithm 1 and Algorithm 4 for the first two methods, respectively. In addition, the *Simple-MI* method is also a baseline method [2], which stands for representing one bag with the mean vector of all the instances in that bag, and associating the mean vector with bag-level label to build a classifier.

We first evaluate the proposed methods on five benchmark data sets popularly used in the studies of MIL, including *Musk1*, *Musk2*, *Elephant*, *Fox* and *Tiger*. In addition, two famous categories, i.e., *Course* and *Faculty*, in WebKB which is a benchmark data set for document classification are used in experiments.[2] One webpage from one of these two categories is taken as a positive bag, and different fixed-length paragraphs are treated as instances. The same number of the negative bags are randomly sampled from the remaining six categories in WebKB. More details of these data sets can be found in [37]. On each of the seven data sets, we run ten times 10-fold cross validation and report the average results.

Because image categorization is one of the most successful applications of MIL [38], [39], two image data sets (*1000-Image* and *2000-Image*) for classification are also used in our experiments. These two data sets contain 10 and 20 categories from *COREL* images, respectively.[3] Images are in the JPEG format of 384×256 or 256×384 image resolution. The category names are listed in Table I along with the identifiers for these 20 categories. We treat each image as a bag and employ the *SBN* bag generator [40] with the default patch size (2×2) to extract instances/patches from each bag/image. For the *1000-Image* and *2000-Image* data sets, we use the same experimental routine as described in [6]. The experiment is repeated five times with different training/test data splittings, and the average results are reported.

Finally, we use three large scale data sets, i.e., *Speaker*, *Process*, and *TRECVID 2005*, to evaluate the scalability of the proposed methods. The Speaker data set [2] was created by Amores based on an audio database. In this case, the task is to identify the gender of the speaker, using as input an audio recording of a sentence spoken by the person. The audio recording is represented as a bag of feature vectors, using a standard representation in the audio processing community.

Table II
DETAILED CHARACTERISTICS OF THE DATA SETS. NOTE THAT, BECAUSE WE USE THE ONE-AGAINST-ONE STRATEGY FOR THE *1000-Image* AND *2000-Image* DATA SETS, "♯ POSITIVE" ("♯ NEGATIVE") FOR THEM PRESENTS THE NUMBER OF POSITIVE (NEGATIVE) BAGS USED IN EACH ROUND. FOR *TRECVID 2005*, BECAUSE IT CONTAINS 39 SUB-DATASETS, WE MERELY PRESENT THE TOTAL NUMBER OF BAGS IN IT.

Data set	♯ attribute	♯ bag			♯ instance
		positive	negative	total	
Musk1	166	47	45	92	476
Musk2	166	39	63	102	6,598
Elephant	230	100	100	200	1,220
Fox	230	100	100	200	1,320
Tiger	230	100	100	200	1,391
Course	320	674	674	1,348	3,528
Faculty	361	795	795	1,590	4,248
1000-Image	121	100	100	1,000	3,000
2000-Image	121	100	100	2,000	3,000
Speaker	20	190	240	430	583,600
Process	200	757	10,961	11,718	118,417
TRECVID	1,000	–	–	61,901	680,911

The large scale text categorization data set, i.e., Process [41], was obtained as part of Task 2 of the BioCreative Text Mining Challenge. Given a name of a human protein and a full-text journal article, the task is to determine whether this protein-article pair can be annotated with a particular gene ontology (GO) term. For the MIL setting, each article is represented as a bag. An instance in a bag refers to a paragraph in an article. Each paragraph is described by a set of word count features, along with a set of numerical features that capture some aspects of the protein-GO code interaction.

The third and last large scale data set is the development (DEV) set of *TRECVID 2005* for the video annotation task. The original data set consists of 80 hours of TV news videos from 13 different programs in English, Arabic and Chinese, and contains 43,907 shots. Some shots have been segmented further into sub-shots. The final DEV set contains 61,901 sub-shots.[4] For each sub-shot, the keyframes have been extracted to represent the sub-shot, and each of them has been associated with one or more concepts in the 39 concept set [42]. In this case, we treat each sub-shot as a bag and each frame in one sub-shot as an instance. Each bag/sub-shot obtains 11 instances/keyframes of 1,000-dimension.

These large scale multiple instance data sets are characterized by different numbers of features ranging from 20 to 1,000, and different numbers of instances ranging from 118,417 to 680,911. The Speaker data set has a small number of bags, but a large number of instances. Process has a large number of bags with a small number of instances. Moreover, TRECVID 2005 has both large numbers of bags and instances. Especially, please note that the numbers of instances per bag are 1,357, 10 and 11 for Speaker, Process and TRECVID 2005, respectively, which will lead to interesting observations in Section IV-E. Detailed characteristics of these 12 data sets are summarized in Table II.

[2]http://www.cs.cmu.edu/~webkb/
[3]http://www.cs.olemiss.edu/~ychen/ddsvm.html
[4]http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.118.6497

B. Accuracy Comparison

We first report the comparison results on the nine MIL data sets (i.e., all data sets except for the three large scale ones) in Table III. Note that, because the number of positive bags is equal to the one of the negative bags in these data sets, the impact of class imbalance can be ignored. So in the experiments of these nine data sets, we use *accuracy* as the evaluation criterion. As shown in the table, miV&F achieves three times the best performance and three times the second best performance in all the nine cases. miFV gets four times the second best performance. miVLAD has comparable performances with miFV, and especially on two document classification data sets, its performances are ideal and satisfactory. In addition, we also compare *miVLAD_def*, *miFV_def* and *miV&F_def* with miVLAD, miFV and miV&F, respectively. miVLAD_def (miFV_def or miV&F_def) indicates miVLAD (miFV or miV&F) runs with its corresponding default parameters. The parameter analysis of the proposed algorithms can be found in Section IV-E.

When comparing different algorithms on several datasets, the Friedman test [43] can be used to compare their overall performance. As shown in Figure 1, the average rank of miFV, miV&F and miVLAD on these data sets are 1st, 2nd and 4th, respectively. Note that, although miV&F achieves not only the best performance but the second best one in six data sets of the nine, the performance of other cases is not satisfactory enough, which pulls down its overall average rank. Moreover, miFV performs significantly better than Simple-MI, MIWrapper, CCE, EM-DD and miSVM. Meanwhile, it can achieve comparable performances with two state-of-the-art MIL algorithms, i.e., MIBoosting and miGraph. In addition, miVLAD also performs well although its average performance is slightly lower than that of miGraph.

C. Efficiency Comparison

Because our goal is to handle large scale MIL problems, it is crucial to study the efficiency of the proposed algorithms. We report the time cost of training time and test time in Figure 2 and Figure 3, respectively. The time costs of training and testing include both the processes of transforming features and training linear classifiers. Note that in these figures the vertical axes are shown in *log-scale*. All the experiments are performed on a machine with a 3.10 GHz CPUs and 16GB main memory.

Obviously, except for Simple-MI, miVLAD and miFV are the most efficient ones on all the data sets. EM-DD is the most time-consuming one, followed by miSVM. For CCE, because it trains an ensemble of classifiers based on multiple clusterings, it is not efficient enough and even worse than EM-DD and miGraph on the four moderate-sized data sets. Compared with the accuracy-wise comparable algorithms, i.e., MIBoosting and miGraph, the proposed algorithms have hundreds of times faster speed. Especially for the four moderate-sized data sets, i.e., *Course*, *Faculty*, *1000-Image* and *2000-Image*, the efficiency of the proposed algorithms is more prominent than other state-of-the-art MIL algorithms.

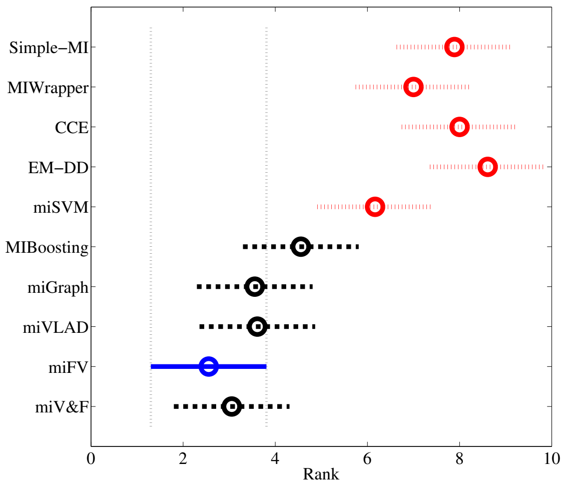

Figure 1. Friedman test results for comparing the proposed methods with other MIL algorithms on nine MIL data sets. The vertical axis indicates the MIL algorithms, and the horizontal axis indicates the rank values. The circle is the average rank for each algorithm and the bar indicates the critical values for a two-tailed test at 95% significance level. When two algorithms having non-overlapping bars, it indicates that they are significantly different. Significantly worse results are presented in dotted bars (colored in red) located on the right-hand-side of the diagram. The best results are presented in solid bars (colored in blue) on the leftmost side in the diagram.

D. Scalability

Now we present the results of the proposed algorithms on the three large scale data set, i.e., *Speaker*, *Process*, and *TRECVID 2005*. For the Process data set, we split the data set into three parts, i.e., training data, validation data and test data. Because this data set has different numbers of positive and negative bags, the training part has 400 positive and 400 negative bags, the validation part has 157 positive and 157 negative ones, and the test part contains 200 positive and 10,404 negative ones. For verifying the effectiveness of our proposed algorithms on the unbalanced testing data set, we employ *F1-score* as the criterion:

$$F_1 = 2 \cdot \frac{PR}{P+R}, \tag{18}$$

where P and R are the precision and recall on the testing data set, respectively. We do the experiments on Process for five times with different data splittings, and report the average F1-score on the test part in Figure 4. The classification performance is consistent with the small-sized MIL data sets. The proposed algorithm (miVLAD, miFV, and miV&F) achieves top F1-score.

We perform similar splitting process to the Speaker and TRECVID 2005 data sets. The training part of Speaker has 120 positive and 120 negative bags, and the validation part has 30 positive and 30 negative ones, and the test part has 40 positive and 90 negative ones. The F1-score on the test part of Speaker is reported in Figure 5.

For TRECVID 2005, these three parts have 40,616, 9,331 and 11,954 bags, respectively. For each concept of the data set, if we treat it as the positive label, then the other concepts

Table III

Comparison results (mean±std.) on 9 data sets. The highest average accuracy of each column is marked in bold and with ●; the second highest one of each column is just in bold. Note that miVLAD_def (miFV_def or miV&F_def) indicates the miVLAD (miFV or miV&F) algorithm runs with its corresponding default parameters.

	Musk1	Musk2	Elephant	Fox	Tiger	Course	Faculty	1000-Image	2000-Image
Simple-MI	.832±.123	.853±.111	.801±.088	.546±.092	.778±.092	.896±.007	.910±.008	.844±.090	.818±.099
MIWrapper	.849±.106	.796±.106	.827±.088	.582±.102	.770±.092	.929±.009	.906±.004	.847±.086	.831±.092
CCE	.831±.027	.713±.024	.793±.021	.599±.027	.760±.012	.936±.006	.934±.006	.805±.102	.801±.095
EM-DD	.849±.098	.869±.108	.771±.098	.609±.101	.730±.096	.538±.120	.410±.008	.741±.145	.739±.139
miSVM	.874±.120	.836±.088	.822±.073	.582±.102	.789±.089	.915±.010	.915±.014	.854±.148	.849±.139
MIBoosting	.837±.120	.790±.088	.827±.073	**.638±.102** ●	.784±.089	.938±.019	.941±.017	**.910±.060** ●	**.898±.063** ●
miGraph	.889±.073	**.903±.086** ●	**.869±.078**	.616±.079	.801±.083	**.980±.001** ●	.854±.006	.896±.070	**.896±.072**
miVLAD	.871±.097	.872±.095	.850±.080	.620±.098	.811±.087	.971±.015	**.969±.007** ●	.878±.081	.868±.079
miFV	**.909±.089**	**.884±.094**	.852±.081	**.621±.109**	**.813±.083**	.968±.009	.961±.007	.899±.070	.882±.070
miV&F	**.915±.083** ●	.881±.087	**.871±.073** ●	.620±.096	**.823±.084** ●	**.976±.007**	**.965±.007**	**.901±.079**	.879±.075
miVLAD_def	.865±.111	.872±.095	.844±.085	.587±.096	.795±.092	.955±.028	.928±.022	.878±.081	.868±.079
miFV_ def	.875±.106	.861±.106	.852±.081	.560±.099	.789±.091	.943±.008	.930±.010	.879±.075	.875±.072
miV&F_def	.898±.089	.877±.102	.868±.071	.573±.108	.789±.094	.960±.004	.949±.007	.879±.078	.876±.083

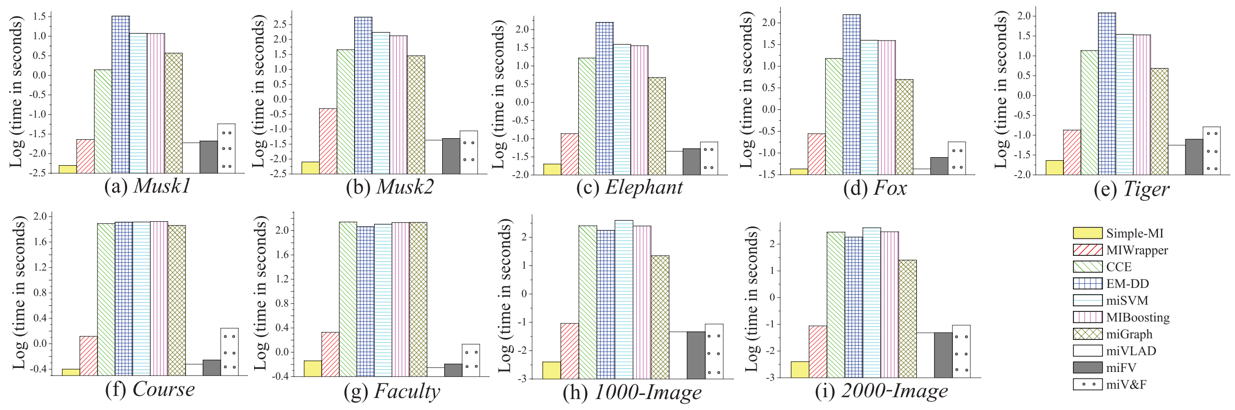

Figure 2. Comparison of mean time cost of **training time** on nine data sets. Note that for the first seven data sets, these results are the average time cost of ten times 10-fold cross validation. For the latter two image data sets, they are the average time cost of the rounds for one-against-one strategy. The vertical axes are shown in *log-scale*. This figure is best viewed in color.

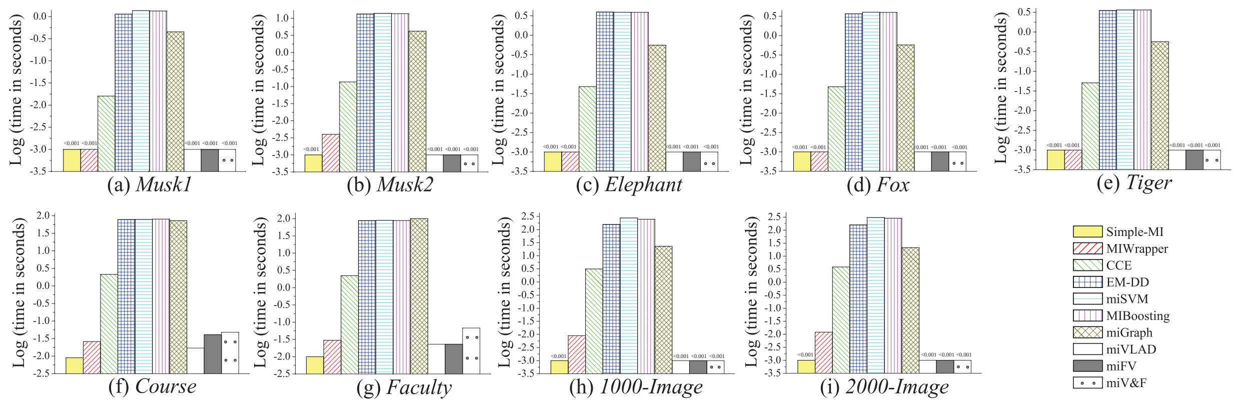

Figure 3. Comparison of mean time cost of **test time** on nine data sets. Note that for the first seven data sets, these results are the average time cost of ten times 10-fold cross validation. For the latter two image data sets, they are the average time cost of the rounds for one-against-one strategy. The vertical axes are shown in *log-scale*. "<0.001" indicates the time cost is less than 0.001 second. This figure is best viewed in color.

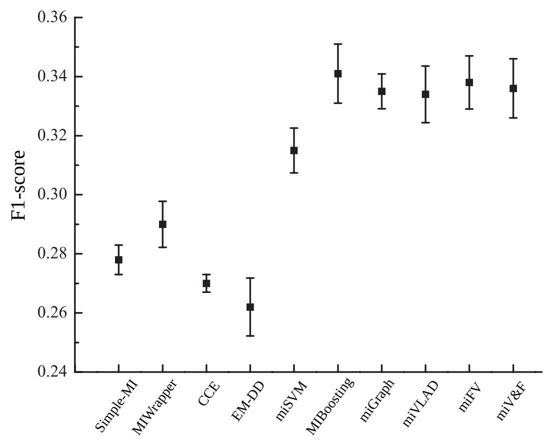

Figure 4. Classification results of the Process data set. The black block represents the average F1-score for each MIL algorithm, and the vertical bar is the standard deviation.

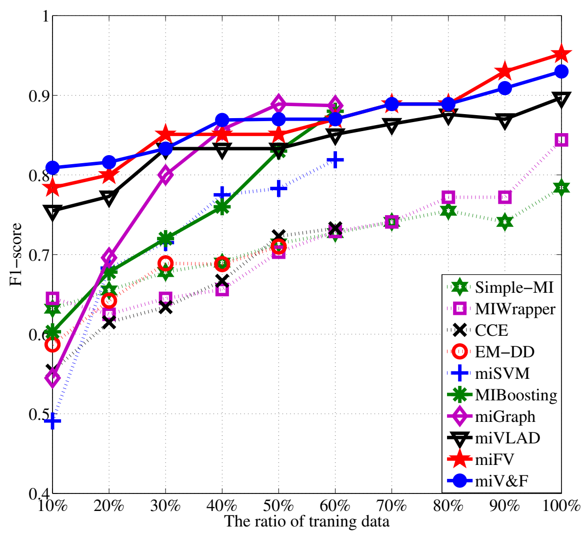

Figure 5. Classification results of the Speaker data set. Note that, when the ratio of training data is bigger than 50%, EM-DD could not return results in 48 hours. And CCE, miSVM, MIBoosting and miGraph return no results in 48h when the ratio is bigger than 60%. This figure is best viewed in color.

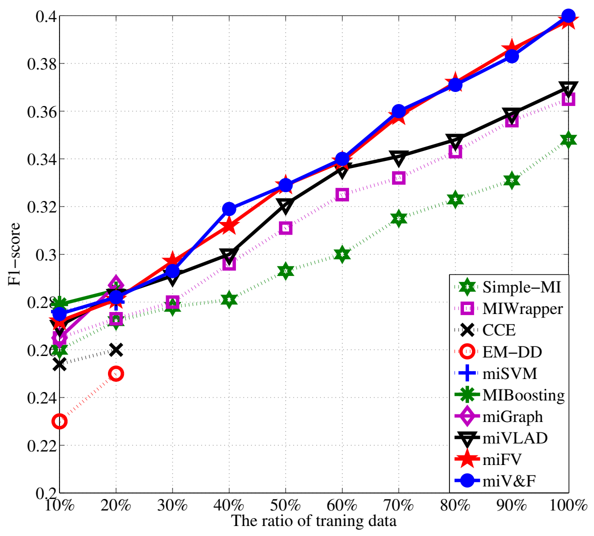

Figure 6. Micro-averaged F-score results of TRECVID 2005. Note that, when the ratio of training data is bigger than 20%, CCE, EM-DD, miSVM, MIBoosting and miGraph can not return results in 48 hours. This figure is best viewed in color.

become the negative ones. Thus, we can get 39 MIL data sets, which are sub-problems of the original data set. Note that, due to the class imbalance phenomenon, for each sub-dataset/concept, we build a binary classifier five times with five random balanced under-samplings to solve this problem. Because this data set contains 39 sub-datasets, we evaluate the general performances of the compared approaches on two commonly used criteria: *micro-averaged F-score* (mi.f.), and *macro-averaged F-score* (ma.f.), which is shown as follows:

$$\text{micro-averaged F-score} = 2 \cdot \frac{P_{all}R_{all}}{P_{all}+R_{all}}, \tag{19}$$

$$\text{macro-averaged F-score} = \frac{\sum_{i=1}^{M} F_i}{M}, \tag{20}$$

where M indicates the number of classes. In micro-averaged F-score, it is computed globally over all category predictions, and P_{all} and R_{all} are obtained by summing over all individual predictions. While macro-averaged F-score is computed locally over each category first and then the average over all categories is taken. $F_i = 2P_iR_i/(P_i+R_i)$ is the corresponding F-score of the i-th class.

However, due to their high computational complexity, the other five MIL algorithms, i.e., CCE, EM-DD, miSVM, MIBoosting and miGraph, could not return results of these two large scale data sets in 48 hours. We sample different ratio of the original training data ranging from 10% to 100% for all the MIL algorithms. The results of Speaker is shown in Figure 5, and the TRECVID 2005's results of mi.f. and ma.f. are presented in Figure 6 and Figure 7, respectively. As shown in these figures, the results of CCE, EM-DD, miSVM, MIBoosting and miGraph are not complete. That is because these complex MIL algorithms can not return results in 48h, even for a small sampling of the original training data. In contrast, our proposed algorithms can return results in less than 0.5 hour for Speaker and several hours for TRECVID 2005 when we use the whole training data.

The curves of the proposed algorithms (miF&V, miFV and miVLAD) clearly are on top of other curves. In other words, when the sampling ratio is small and results for other data sets are available, the proposed algorithms still exhibits clear advantages in terms of various accuracy evaluation metrics.

E. Parameter Analysis

As aforementioned, for miVLAD and miFV, the two most important parameters are the number of centers and the PCA energy. Note that, the parameters of LIBLINEAR is fixed when we change the value of parameters in both miVLAD and miFV. In this section, we first report the results of the proposed algorithms with different parameter values on small-

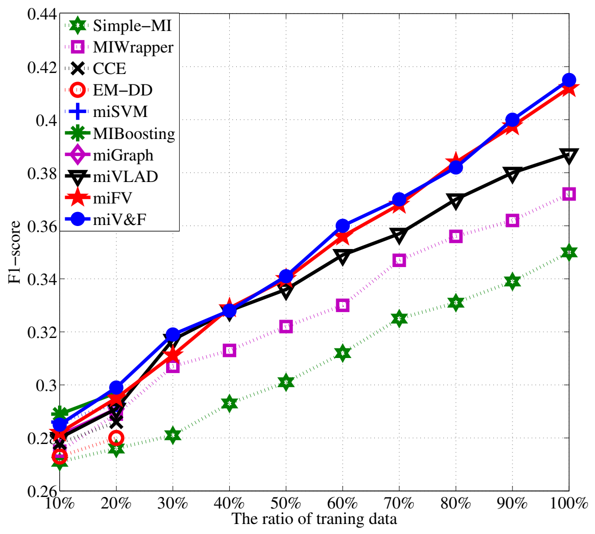

Figure 7. Macro-averaged F-score results of TRECVID 2005. Note that, when the ratio of training data is bigger than 20%, CCE, EM-DD, miSVM, MIBoosting and miGraph can not return results in 48 hours. This figure is best viewed in color.

and moderate-sized data sets. Then, we perform the parameter analysis on two large scale ones, i.e., Speaker and Process.

1) Results of Small- and Moderate-sized Data Sets: The parameter analysis results of the proposed algorithms on the nine small- and moderate-sized data sets are presented in Table IV and Table V, respectively. We first fix the value of PCA energy to 1.0 (that is, not using PCA). As shown in Table IV, miVLAD can achieve the best performance in most cases when the number of centroids is 2. For the two image data sets, miVLAD achieves the best performance when we do not use PCA. However it can get the best performance when PCA is used on the two document classification data sets. That may be related to the noises in the document data sets, which can be overcome by using PCA to get better performance. But, miVLAD can achieve the best performance in most cases when PCA is not used. In addition, similar phenomena also exist in miFV. In consequence, we take the number of centroids of 2 and PCA energy of 1.0 as the default parameters in miVLAD; and the number of Gaussian components of 1 and PCA energy of 1.0 as the default parameters in miFV, respectively. In addition, the default parameters in miV&F is the same as the corresponding ones in miVLAD and miFV.

As shown in Table III, the performances of the proposed algorithms with default parameters are comparable with the ones with the well tuned parameters. That illustrates another advantage of the proposed algorithms: it is convenient to employ miVLAD or miFV. Even with the default parameters, it can also achieve a satisfactory result efficiently.

2) Results of Large Scale Data Sets: Now we do the parameter analysis experiments on the Speaker and Process data set. In the former results of parameter analysis, we can find that the optimum parameters of numbers of centers is 1 or 2 for miVLAD or miFV, respectively. However, during the experiments on Speaker, we find that when we set the numbers of centers as 1 or 2 for the proposed methods, the F1-score is not satisfactory. Thus, we change the numbers of centers into a bigger one, e.g., 64 or 128. As a consequence, we find that the F1-score gets a significant improvement. But the Process data set has similar results of those small- and moderate-sized data sets. The parameter analysis results of these two large scale data sets is reported in Table VI and Table VII, respectively.

As shown in these tables, the proposed algorithms achieve the best performance on the Speaker data set when the number of centers is 256, while the optimal number is 2 on the Process data set. We can explain this difference by the average number of instances per bag. Recall that these two large scale data sets have significant different numbers of instances per bag. On average the bags in Speaker has 1357.2 instances, while the number is only 10.1 for Process. Hence, we reach the conclusion that: when each bag in a data set contains a large number of instances, e.g., thousands of instances, it is better to use a big number of centers (e.g., 128 or 256) for miVLAD/miFV; when the number of instances in each bag is small, a small number of centers (e.g., 1 or 2) is better. This observation is consistent with the performances of VLAD and FV in the computer vision tasks, in which there are thousands or tens of thousands of local descriptors in one image, and the number of centers are usually 64, 128 or 256.

V. Discussions

In this section, we discuss the relationship between Simple-MI, CCE and the proposed algorithms.

A. Relationship between miVLAD and miFV

We first discuss the relationship between the two proposed algorithms. For miFV, it first approximates the distribution of instances in bags with a GMM, which can be treated as a soft assignment method for instances. After that, different higher-order statistics shown in equations (15)-(17) are computed to represent the original bag. For miVLAD, the mapping function $\mathcal{M}_v$ consists of assigning each instance $\boldsymbol{x}_{ij}$ in bag X_i to its closest centroid $\boldsymbol{c}_k$ and accumulating the differences $\boldsymbol{x}_{ij} - \boldsymbol{c}_k$. Hence, miVLAD is a simplified version of miFV under the following approximations: 1) the soft assignment is replaced by a hard assignment and 2) only the gradient with respect to the mean is considered without other higher-order statistics [29], [30]. In consequence, it is better to use the mapping function $\mathcal{M}_f$ to map bags into FVs, because some information might be lost by using the mapping function $\mathcal{M}_v$ in obtaining the new feature vectors. As shown in Table III, miFV outperforms miVLAD in almost all cases. However, miVLAD is more efficient than miFV in most cases. In addition, we can also employ miV&F to get better accuracy rates, while it will roughly double the time of miVLAD/miFV.

B. Simple-MI vs. miVLAD/miFV

Then we compare Simple-MI with the proposed algorithms. We find that Simple-MI only computes the mean vector for each bag. In order to compare these three algorithms in an experimental setting that is fair to all algorithms, we set

Table IV
PARAMETER ANALYSIS RESULTS (MEAN±STD.) OF **MIVLAD** ON 9 DATA SETS. THE HIGHEST AVERAGE ACCURACY OF EACH COLUMN IS MARKED IN BOLD. NOTE THAT WHEN THE NUMBER OF CENTROIDS IS TUNED FROM 1 TO 6, THE PCA ENERGY IS FIXED ON THE VALUE OF 1.0, AND WHEN THE PCA ENERGY IS TUNED, THE NUMBER OF CENTROIDS IS FIXED ON THE VALUE OF 2.

Data set / $\sharp$ of centers	Musk1	Musk2	Elephant	Fox	Tiger	Course	Faculty	1000-Image	2000-Image
1	.836±.106	.857±.106	.829±.091	.589±.105	**.809±.088**	**.968±.005**	**.961±.009**	.869±.079	.854±.081
2	**.865±.111**	**.872±.095**	**.844±.085**	**.611±.092**	.795±.092	.955±.028	.928±.022	**.878±.081**	**.868±.079**
3	.849±.110	.871±.098	.825±.097	.585±.103	.788±.091	.921±.026	.918±.023	.871±.079	.863±.077
4	.831±.099	.865±.098	.818±.097	.579±.103	.781±.097	.925±.028	.928±.020	.866±.081	.854±.078
5	.813±.107	.854±.107	.813±.097	.578±.108	.767±.097	.923±.032	.923±.023	.862±.083	.854±.081
6	.812±.112	.853±.105	.812±.091	.576±.105	.756±.099	.923±.038	.927±.018	.862±.086	.857±.079

Data set / PCA energy	Musk1	Musk2	Elephant	Fox	Tiger	Course	Faculty	1000-Image	2000-Image
1.0	**.865±.111**	**.872±.095**	.844±.085	.611±.092	.795±.092	.955±.028	.928±.022	**.878±.081**	**.868±.079**
0.9	.861±.094	.853±.118	.829±.092	**.620±.098**	**.811±.084**	**.971±.015**	.942±.021	.822±.114	.812±.102
0.8	.766±.117	.748±.126	**.848±.080**	.610±.108	**.811±.087**	.942±.028	**.949±.018**	.772±.104	.771±.104

Table V
PARAMETER ANALYSIS RESULTS (MEAN±STD.) OF **MIFV** ON 9 DATA SETS. THE HIGHEST AVERAGE ACCURACY OF EACH COLUMN IS MARKED IN BOLD. NOTE THAT WHEN THE NUMBER OF GAUSSIAN COMPONENTS IS TUNED FROM 1 TO 6, THE PCA ENERGY IS FIXED ON THE VALUE OF 1.0, AND WHEN THE PCA ENERGY IS TUNED, THE NUMBER OF CENTROIDS IS FIXED ON THE VALUE OF 2.

Data set / $\sharp$ of centers	Musk1	Musk2	Elephant	Fox	Tiger	Course	Faculty	1000-Image	2000-Image
1	.875±.106	.861±.106	**.852±.081**	**.560±.099**	**.789±.091**	**.943±.008**	**.930±.010**	.879±.075	.875±.072
2	**.909±.089**	**.864±.096**	.829±.091	.542±.096	.765±.097	.932±.008	.923±.013	**.899±.070**	**.882±.070**
3	.888±.098	.844±.123	.806±.093	.538±.128	.712±.107	.932±.009	.919±.009	.881±.070	.879±.073
4	.889±.897	.835±.113	.781±.096	.554±.113	.708±.115	.932±.008	.921±.016	.882±.068	.877±.073
5	.864±.104	.831±.131	.764±.109	.531±.122	.686±.107	.930±.006	.922±.012	.881±.073	.878±.070
6	.836±.121	.827±.118	.754±.094	.539±.112	.686±.108	.930±.007	.923±.014	.880±.074	.878±.072

Data set / PCA energy	Musk1	Musk2	Elephant	Fox	Tiger	Course	Faculty	1000-Image	2000-Image
1.0	**.909±.089**	**.864±.096**	.829±.091	.542±.096	.765±.097	.932±.008	.923±.013	**.899±.070**	**.882±.070**
0.9	.840±.116	.843±.116	**.851±.079**	.595±.103	.795±.084	**.968±.009**	**.960±.008**	.836±.107	.833±.097
0.8	.763±.117	.752±.126	.836±.087	**.621±.109**	**.813±.086**	.964±.010	.959±.005	.793±.094	.798±.095

Table VI
PARAMETER ANALYSIS RESULTS OF MIVLAD AND MIFV ON THE **SPEAKER** DATA SET. THE HIGHEST F1-SCORE OF EACH ROW IS MARKED IN BOLD. NOTE THAT PCA IS NOT USED IN THESE EXPERIMENTS.

$\sharp$ of centers	2	20	64	128	256	512
miVLAD	0.702	0.769	0.798	0.833	**0.897**	0.887
miFV	0.714	0.760	0.813	0.899	**0.952**	0.913

Table VII
PARAMETER ANALYSIS RESULTS OF MIVLAD AND MIFV ON THE **PROCESS** DATA SET. THE HIGHEST F1-SCORE OF EACH ROW IS MARKED IN BOLD. NOTE THAT IN THESE EXPERIMENTS, PCA ENERGY IS FIXED ON THE VALUE OF 0.9.

$\sharp$ of centers	1	2	3	4	5	6
miVLAD	0.270	**0.334**	0.320	0.313	0.305	0.300
miFV	0.259	**0.337**	0.325	0.316	0.314	0.282

the number of centroids in miVLAD and the number of Gaussian components in miFV both to 1. The results on nine MIL data sets are presented in Table VIII. In this table, miVLAD outperforms other algorithms on four data sets, while miFV beats other ones on five data sets. However, Simple-MI performs worst on all the nine MIL data sets. For Simple-MI, it only contains the average information of instances in bags. So, it works well when the average of positive and negative bags is quite different; while it will have poor accuracy rates when the average of two bags is similar [2]. However, for miVLAD and miFV, as shown in Equation (1), (16) and (17), they can be considered as using the mapping functions to model how two "distributions" (one is the MIL bag; the other is the whole codebook in miVLAD/miFV) is different. Therefore, even though the number of centers are set to 1, the proposed algorithms can still capture more discriminative information than Simple-MI. Because Simple-MI merely has the average information of instances and contains no higher-order statistics, it is not a surprise that its accuracy rates is the lowest one among these three algorithms.

C. CCE vs. miVLAD/miFV

Finally, we compare CCE with miVLAD and miFV. In CCE [9], it firstly collects the instances in all the bags together, and then clusters the instances into c groups. Each bag is then re-represented by c binary features, where the value of the i-th feature is set to one if the concerned bag has instances falling into the i-th group and zero otherwise. Hence, a standard supervised learner can handle it. In addition, in order to improve the robustness, it produces many classifiers based on different clustering results and then combines their predictions. As aforementioned, miFV considers higher-order statistics, while miVLAD is only with the first-order statistics. Here, CCE just takes binary codes mentioned above as feature vectors, whose vectors contain much less information than that of either miVLAD or miFV. Thus, possibly owing to

Table VIII
COMPARISON RESULTS (MEAN±STD.) OF SIMPLE-MI, MIVLAD AND MIFV ON 9 DATA SETS. THE HIGHEST AVERAGE ACCURACY OF EACH COLUMN IS SHOWN IN BOLD. NOTE THAT, THE NUMBER OF CENTROIDS IN MIVLAD AND THE NUMBER OF GAUSSIAN COMPONENTS IN MIFV ARE BOTH SET TO 1.

Algorithm \ Data set	Musk1	Musk2	Elephant	Fox	Tiger	Course	Faculty	1000-Image	2000-Image
Simple-MI	.832±.123	.853±.111	.801±.088	.546±.092	.778±.092	.896±.007	.910±.008	.844±.090	.818±.099
miVLAD	.836±.106	.857±.106	.829±.091	**.589±.105**	**.809±.088**	**.968±.005**	**.961±.009**	.869±.079	.854±.081
miFV	**.875±.106**	**.861±.106**	**.852±.081**	.560±.099	.789±.091	.943±.008	.930±.010	**.879±.075**	**.875±.072**

the difficulty of setting an adequate number of clusters, the accuracy of CCE is inferior to that of miVLAD or miFV though it uses an additional ensemble step (cf. Figure 1).

VI. CONCLUSION

In domains where complex objects such as images or genes are involved, multi-instance learning has achieved great success. However, one common drawback of existing MIL algorithms is that they are time-consuming and cannot deal with large scale problems. We extend our preliminary study [11] in this paper and propose two efficient and scalable MIL algorithms, i.e., miVLAD and miFV, which transform the bag form of MIL into new feature vector representations. On one hand, the proposed algorithms have hundreds of, even thousands of times faster speed than state-of-the-art MIL algorithms; on the other hand, the proposed algorithms can achieve comparable performances with other MIL algorithms, e.g., MIBoosting and miGraph, and they perform significantly better than miSVM, EM-DD, MIWrapper and CCE. In addition, the proposed algorithms can also perform well conveniently when they are deployed with the default parameters.

The future research direction is related to the observations in Table VIII. As we have discussed in Section V, we can sort the following algorithms in the increasing order of how much information in the original bag is encoded in the final vector representation: Simple-MI $\approx$ CCE $<$ miVLAD $<$ miFV. And, from Figure 3, we can conclude that their accuracy follows the same order in our experiments. This observation should not be a pure coincidence, but hints that we need to find new vector representations that keeps more information from the bags. In addition, it is possible to extend our proposed algorithms to deal with the multi-instance multi-class problems, especially those problems with a great number of classes.

REFERENCES

[1] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial Intelligence*, vol. 89, no. 1-2, pp. 31–71, 1997.

[2] J. Amores, "Multiple instance classification: Review, taxonomy and comparative study," *Artificial Intelligence*, vol. 201, pp. 81–105, 2013.

[3] S. Andrews, I. Tsochantaridis, and T. Hofmann, "Support vector machines for multiple-instance learning," in *Advances in Neural Information Processing Systems 15*, Vancouver, Canada, Dec. 2002, pp. 561–568.

[4] Q. Zhang and S. A. Goldman, "EM-DD: An Improved Multiple-Instance Learning Technique," in *Advances in Neural Information Processing Systems 14*, British Columbia, Canada, Dec. 2001, pp. 1073–1080.

[5] X. Xu and E. Frank, "Logistic Regression and Boosting for Labeled Bags of Instances," in *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Sydney, Australia, May. 2004, pp. 272–281.

[6] Z.-H. Zhou, Y.-Y. Sun, and Y.-F. Li, "Multi-Instance Learning by Treating Instances As Non-I.I.D. Samples," in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, Jun. 2009, pp. 1249–1256.

[7] Y. Chen, J. Bi, and J.-Z. Wang, "MILES: Multiple-instance learning via embedded instance selection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1931–1947, Dec. 2006.

[8] J. Wang and J.-D. Zucker, "Solving Multiple-Instance Problem: A Lazy Learning Approach," in *Proceedings of the 17th International Conference on Machine Learning*, San Francisco, CA, Jun. 2000, pp. 1119–1125.

[9] Z.-H. Zhou and M.-L. Zhang, "Solving Multi-Instance Problems with Classifier Ensemble based on Constructive Clustering," *Knowledge and Information Systems*, vol. 11, pp. 155–170, 2007.

[10] C. Xu, D. Tao, and C. Xu, "A survey on multi-view learning," *http://arxiv.org/pdf/1304.5634v1.pdf*.

[11] X.-S. Wei, J. Wu, and Z.-H. Zhou, "Scalable Multi-Instance Learning," in *Proceedings of the 14th IEEE International Conference on Data Mining*. Shenzhen, China, Dec. 2014, pp. 1037–1042.

[12] O. Maron and T. Lozano-Pérez, "A framework for multiple-instance learning," in *Advances in Neural Information Processing Systems 10*, Denver, CO, Dec. 1997, pp. 570–576.

[13] E. T. Frank and X. Xu, "Applying propositional learning algorithms to multi-Instance data," University of Waikato, Hamilton, NZ, Tech. Rep., Jun. 2003.

[14] Y. Xu and W. Ping, "Multi-instance metric learning," in *Proceedings of the 11st IEEE International Conference on Data Mining*, Vancouver, Canada, Dec. 2011, pp. 874–883.

[15] A. Shrivastava, V. M. Patel, J. K. Pillai, and R. Chellappa, "Generalized dictionaries for multiple instance learning," *International Journal of Computer Vision*, vol. 114, no. 2, pp. 288–305, 2015.

[16] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola, "Multi-instance kernels," in *Proceedings of the 19th International Conference on Machine Learning*, Sydney, Australia, Jul. 2002, pp. 179–186.

[17] Y.-X. Chen and J. Z. Wang, "Image categorization by learning and reasoning with regions," *Journal of Machine Learning Research*, vol. 5, pp. 913–939, 2004.

[18] H. Blockeel, D. Page, and A. Srinivasan, "Multi-instance tree learning," in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, Aug. 2005, pp. 57–64.

[19] Z.-H. Zhou and J.-M. Xu, "On the Relation between Multi-Instance Learning and Semi-Supervised Learning," in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, Jun. 2007, pp. 1167–1174.

[20] Z. Fu, G. Lu, K.-M. Ting, and D. Zhang, "Learning sparse kernel classifiers for multi-instance classification," *IEEE Trans. Neural Networks and Learning Systems*, vol. 15, pp. 1377–1389, Sep. 2013.

[21] J. Wu, X. Zhu, C. Zhang, and Z. Cai, "Multi-instance learning from positive and unlabeled bags," in *Proceedings of the 18th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Tainan, Taiwan, May. 2014, pp. 237–248.

[22] C. Xu, D. Tao, and C. Xu, "Multi-view intact space learning," *IEEE Trans. Pattern Analysis and Machine Intelligence*, to appear.

[23] J. Yu, Y. Rui, and B. Chen, "Exploiting click constraints and multi-view features for image re-ranking," *IEEE Trans. Multimedia*, vol. 16, pp. 159–168, Jan. 2014.

[24] J. Yu, Y. Rui, Y.-Y. Tang, and D. Tao, "High-order distance-based multiview stochastic learning in image classification," *IEEE Trans. Cybernetics*, vol. 44, pp. 2431–2442, Dec. 2014.

[25] J. Yu, Y. Rui, and D. Tao, "Click predictions for web image reranking using multimodal sparse coding," *IEEE Trans. Image Processing*, vol. 23, pp. 2019–2032, May. 2014.

[26] J. Liu, Y. Jiang, Z. Li, Z.-H. Zhou, and H. Lu, "Partially shared latent factor learning with multiview data," *IEEE Trans. Neural Networks and Learning Systems*, vol. 26, pp. 1233–1246, Jun. 2015.

[27] S.-Y. Li, Y. Jiang, and Z.-H. Zhou, "Partial multi-view clustering," in *Proceedings of the 28th AAAI Conference on Artifical Intelligence*. Quèbec, Canada, Jul. 2014, pp. 1968–1974.
[28] W. Wang and Z.-H. Zhou, "Multi-view active learning in the non-realizable case," in *Advances in Neural Information Processing Systems 23*. Vancouver, Canada, Dec. 2010, pp. 2388–2396.
[29] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, Jun. 2010, pp. 3304–3311.
[30] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image Classification with the Fisher Vector: Theory and Practice," *International Journal of Computer Vision*, vol. 105, no. 3, pp. 222–245, 2013.
[31] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.
[32] J. Wu and H. Yang, "Linear Regression Based Efficient SVM Learning for Large Scale Classification," *IEEE Trans. Neural Networks and Learning Systems*, vol. 26, pp. 2357–2369, Oct. 2015.
[33] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, "Large-scale image retrieval with compressed fisher vectors," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, Jun. 2010, pp. 3384–3391.
[34] J. Wu, Y. Zhang, and W. Lin, "Towards Good Practices for Action Video Encoding," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, Jun. 2014, pp. 2577–2584.
[35] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Advances in Neural Information Processing Systems 11*, Denver, CO, Nov. 1998, pp. 487–493.
[36] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, Jun. 2007, pp. 1–8.
[37] D. Zhang, J. He, and R. Lawrence, "MI2LS: Multi-Instance Learning from Multiple Information Sources," in *Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Chicago, IL, Aug. 2013, pp. 149–157.
[38] S. Vijayanarasimhan and K. Grauman, "Keywords to Visual Categories: Multiple-Instance Learning for Weakly Supervised Object Categorization," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, Jun. 2008, pp. 1–8.
[39] D. Zhang, F. Wang, L. Si, and T. Li, "Maximum Margin Multiple Instance Clustering with Applications to Image and Text Clustering," *IEEE Trans. Neural Networks*, vol. 22, pp. 739–751, May. 2011.
[40] O. Maron and A. L. Ratan, "Multiple-instance learning for natural scene classification," in *Proceedings of the 15th International Conference on Machine Learning*, Madison, WI, USA, Jul. 1998, pp. 341–349.
[41] S. Ray and M. Craven, "Supervised versus Multiple Instance Learning: An Empirical Comparison," in *Proceedings of the 22th International Conference on Machine Learning*, Bonn, Germany, Aug. 2005, pp. 697–704.
[42] M. Naphade, L. Kennedy, J. R. Kender, S. F. Chang, P. Over, and A. Hauptmann, "A light scale concept ontology for multimedia understanding for TRECVID 2005," IBM Research, Tech. Rep., 2005.
[43] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

Xiu-Shen Wei received the BS degree in Computer Science and Technology in 2012. He is currently a PhD candidate in the Department of Computer Science and Technology at Nanjing University, China. His research interests are computer vision and machine learning.

Jianxin Wu (M'09) received his BS and MS degrees in computer science from Nanjing University, and his PhD degree in computer science from the Georgia Institute of Technology. He is currently a professor in the Department of Computer Science and Technology at Nanjing University, China, and is associated with the National Key Laboratory for Novel Software Technology, China. He was an assistant professor in the Nanyang Technological University, Singapore, and has served as an area chair for ICCV 2015 and senior PC member for AAAI 2016. His research interests are computer vision and machine learning. He is a member of the IEEE.

Zhi-Hua Zhou (S'00-M'01-SM'06-F'13) received the BSc, MSc and PhD degrees in computer science from Nanjing University, China, in 1996, 1998 and 2000, respectively, all with the highest honors. He joined the Department of Computer Science & Technology at Nanjing University as an Assistant Professor in 2001, and is currently Chair Professor and Standing Deputy Director of the National Key Laboratory for Novel Software Technology; he is also the Founding Director of the LAMDA group. His research interests are mainly in artificial intelligence, machine learning, data mining and pattern recognition. He has authored the book Ensemble Methods: Foundations and Algorithms, and published more than 100 papers in top-tier international journals or conference proceedings. He has received various awards/honors including the National Natural Science Award of China, the IEEE CIS Outstanding Early Career Award, the Microsoft Professorship Award, the Fok Ying Tung Young Professorship Award, and twelve international journals/conferences papers or competitions awards. He also holds 15 patents. He is an Executive Editor-in-Chief of the *Frontiers of Computer Science*, Associate Editor-in-Chief of the *Science China: Information Sciences*, Associate Editor of the *ACM Transactions on Intelligent Systems and Technology*, *IEEE Transactions on Neural Networks and Learning Systems*, etc. He served as Associate Editor-in-Chief for *Chinese Science Bulletin* (2008-2014), Associate Editor for *IEEE Transactions on Knowledge and Data Engineering* (2008-2012) and *Knowledge and Information Systems* (2003-2008). He founded ACML (Asian Conference on Machine Learning), served as Advisory Committee member for IJCAI, Steering Committee member for PAKDD and PRICAI, and Chair of various conferences such as General co-chair of ACML 2012, PCM 2013, PAKDD 2014, Program co-chair of SDM 2013, ICDM 2015, IJCAI 2015 Machine Learning Track, Workshop co-chair of KDD 2012, ICDM 2014, Tutorial co-chair of CIKM 2014, KDD 2015, and Area chair of ICML, NIPS, etc. He is the Chair of the IEEE CIS Data Mining and Big Data Analytics Technical Committee, the Chair of the CAAI (Chinese Association of Artificial Intelligence) Machine Learning Society, the Chair of the CCF (China Computer Federation) Artificial Intelligence & Pattern Recognition Society, and the Chair of the IEEE Computer Society Nanjing Chapter. He is an ACM Distinguished Scientist, IEEE Fellow, IAPR Fellow, IET/IEE Fellow and CCF Fellow.