Compressive Sequential Learning for Action Similarity Labeling

Jie Qin, Li Liu, Zhaoxiang Zhang, *Senior Member, IEEE*, Yunhong Wang, *Member, IEEE*, and Ling Shao, *Senior Member, IEEE*

***Abstract*— Human action recognition in videos has been extensively studied in recent years due to its wide range of applications. Instead of classifying video sequences into a number of action categories, in this paper, we focus on a particular problem of action similarity labeling (ASLAN), which aims at verifying whether a pair of videos contain the same type of action or not. To address this challenge, a novel approach called compressive sequential learning (CSL) is proposed by lever- aging the compressive sensing theory and sequential learning. We first project data points to a low-dimensional space by effec- tively exploring an important property in compressive sensing: the restricted isometry property. In particular, a very sparse measurement matrix is adopted to reduce the dimensionality efficiently. We then learn an ensemble classifier for measuring similarities between pairwise videos by iteratively minimizing its empirical risk with the AdaBoost strategy on the training set. Unlike conventional AdaBoost, the weak learner for each iteration is not explicitly defined and its parameters are learned through greedy optimization. Furthermore, an alternative of CSL named compressive sequential encoding is developed as an encoding technique and followed by a linear classifier to address the similarity-labeling problem. Our method has been systematically evaluated on four action data sets: ASLAN, KTH, HMDB51, and Hollywood2, and the results show the effectiveness and superiority of our method for ASLAN.**

***Index Terms*— Action recognition, action similarity labeling, pair matching, boosting, sparse random projection.**

Manuscript received May 4, 2015; revised November 6, 2015; accepted December 8, 2015. Date of publication December 17, 2015; date of current version January 5, 2016. This work was supported in part by the Foundation for Innovative Research Groups through the National Natural Science Foundation of China under Grant 61573045 and Grant 61421003, in part by the Hong Kong, Macao and Taiwan Science and Technology Cooperation Program of China under Grant L2015TGA9004, in part by the National Natural Science Foundation of China under Grant 61528106, Grant 61375036, and Grant 61511130079, in part by the Newton International Exchanges Scheme, in part by the Program for New Century Excellent Talents in University, in part by the Beijing Higher Education Young Elite Teacher Project, and in part by the China Scholarship Council. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. DimitriosTzovaras. *(Corresponding authors: Ling Shao and Yunhong Wang).*

J. Qin and Y. Wang are with the Laboratory of Intelligent Recognition and Image Processing, State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China (e-mail: qinjiebuaa@gmail.com; yhwang@buaa.edu.cn).

L. Liu and L. Shao are with the Department of Computer Science and Digital Technologies, Northumbria University, Newcastle upon Tyne NE1 8ST, U.K. (e-mail: li2.liu@northumbria.ac.uk; ling.shao@northumbria.ac.uk).

Z. Zhang is with the Research Center for Brain-Inspired Intelligence, CAS Center for Excellence in Brain Science and Intelligence Technology, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: zhaoxiang.zhang@ia.ac.cn).

I. Introduction

HUMAN action recognition in videos has been an active research area in computer vision spanning a wide range of application domains, such as intelligent surveillance, human-computer interaction, health monitoring and content-based video retrieval. Due to its vital importance, a variety of approaches have been proposed to address the problem of action classification or recognition [1]–[8]. The goal of human action recognition in videos is to classify unlabeled videos containing different types of actions into a predefined set of action categories. Recently, the computer vision community has been mainly focusing on actions performed in unconstrained videos, targeting real-world action recognition. Unsurprisingly, these videos always present large variances in viewpoint, illumination, scale and appearance, and other complications such as occlusions, background cluttering and multiple actors performing actions in the same scene.

In such realistic settings, there is always an inevitable problem that different types of actions could exist in a short video sequence. It becomes extremely difficult to classify such a sequence into an exact type of action. For instance, a "springboard diving" mainly consists of "jumping", "falling", and "swimming". How to define complex actions appropriately is highly subjective and may vary from one subject to another. In other words, class ambiguity is a common problem existing in multi-class action recognition, especially in unconstrained videos. Consequently, to avoid this ambiguity problem, action similarity labeling or action pair matching has been introduced by Kliper-Gross *et al.* [9] as another critical task, which is the focus of this paper, together with a benchmark dataset named The Action Similarity Labeling (ASLAN) Challenge (see Fig. 1 for examples of action pairs). The aim of action similarity labeling is to determine if the actors in two video sequences are performing the same action or not. In addition to avoiding the ambiguity of multi-class action recognition, action similarity labeling has its own advantages. Firstly, it is not limited to a given set of actions. In terms of multi-class action recognition, one may first learn the models corresponding to some predefined training actions and classify unknown videos based on these models. However, in the setting of action similarity labeling, no action labels are given. Therefore, it is a more generalized approach to determining whether two never-before-seen video sequences contain the same action. Secondly, action similarity labeling provides insights toward content-based video retrieval. Given an unknown video, retrieving videos

(a) (b)

Fig. 1. Examples of (a) pairs of "same"-labeled actions and (b) pairs of "not-same"-labeled actions.

including the same type of action is the basic characteristic of image/video search engines. Thereby, addressing the similarity labeling/pair matching problem effectively and efficiently will enhance the performance of video retrieval significantly.

To address the action similarity labeling problem, several methods have been proposed [9]–[16], which can be mainly divided according to two perspectives: similarity measurement [9]–[13] and feature representation [14]–[16]. However, novel feature extraction and representation is a non-trivial task and learning a suitable metric or projection matrix to measure similarities of high dimensional vectors is typically complicated and time-consuming. In this work, we address the problem of action similarity labeling "in the wild" based on a novel approach called Compressive Sequential Learning (CSL), by exploring the techniques of compressive sensing and sequential learning. Due to complexities of realistic actions, high dimensional feature vectors are typically extracted, which usually leads to low computing efficiency and large storage space. To eliminate these shortcomings, we first project data points to a lower dimensional space based on an important property in compressive sensing: the Restricted Isometry Property (RIP) [17], which is able to preserve the distances between the points in a vector space. The projection can also obtain more compact data representations and balance the components' variance [18]. Instead of the typical measurement matrix (random Gaussian matrix) satisfying RIP, we employ a very sparse random measurement matrix, which will reduce the memory and computational cost significantly. After projecting original high dimensional vectors to a relatively low dimensional space, a novel pair-wise sequential learning method motivated by [19] is proposed to exploit the information between pair-wise data based on the spirit of the boosting theory. In [19], a boosting strategy was utilized to learn each bit of an extremely compact binary local feature descriptor. However, in this work, by leveraging the boosting-trick, we simultaneously optimize the output of an ensemble classifier and reduce the empirical loss on pair-wise training data. Specifically, for each iteration of boosting, unlike [19], in which the weak learner is explicitly pre-defined, we learn a non-linear weak learner whose parameters are optimized based on the pair-wise training data and by efficiently using greedy optimization. The obtained CSL classifier is of the same form as a typical AdaBoost [20] strong classifier, being the sign of a linear combination of previously learned non-linear weak learners. Additionally, as an alternative to the sign function, a linear SVM classifier is applied on the training data after aggregating the outputs of non-linear weak learners. In this case, our CSL method turns into Compressive Sequential Encoding (CSE), working as a strategy for extremely compact feature coding [21]–[23] for pair-wise data.

The rest of this paper is organized as follows. In Section II, related work for action similarity labeling is discussed. Section III describes our method in detail. The global sequential learning based on the boosting-trick is first proposed, and then the weak learner is defined and learned via greedy optimization. Subsequently, sparse random projection is introduced based on the compressive sensing theory. Finally, the overall framework is revisited and an alternative method to CSL is discussed. In Sections IV and V, extensive experimental results on four public action datasets, i.e., ASLAN, KTH, HMDB51 and Hollywood2, are shown, which prove the superiority of our method.

II. Related Work

Action similarity labeling or action pair matching is a recently introduced task in the area of action recognition, focusing on the same/not-same classification of two never-before-seen video sequences. Performance in this task highly depends on the feature representation and the suitability of the similarity measurement used for comparing video pairs.

In terms of the similarity measurement, the first attempt [9] employed twelve pre-defined (dis-)similarity functions (e.g., Euclidean and chi-square distances) and their combinations as the baseline method. After that, Kliper-Gross *et al.* [11] proposed a supervised metric learning method called One Shot Similarity Metric Learning (OSSML) to learn a projection matrix which improves the (dis-)similarity between same/not-same training pairs in a reduced subspace of the original feature space. Moreover, before applying the final OSSML projection, PCA or Cosine Similarity Metric Learning (CSML) was adopted as the initial projection. It has been validated on ASLAN that OSSML achieves better performance (approximately 4% improvement in terms of the Area Under Curve (AUC)) than manually defined (dis-)similarity functions. Similar to the metric learning method, Peng *et al.* [13] developed a large margin dimensionality reduction (LMDR) method to compress high-dimensional feature representations encoded by both Fisher Vector (FV) [24] and Vector of Locally Aggregated Descriptors (VLAD) [18]. A projection matrix was learned to increase the similarities of "same" video pairs and meanwhile separate "not-same" video pairs by a large margin. Instead of learning a proper metric, Cheng [10] proposed a similarity-score based paradigm to learn a rectangular similarity matrix of a fixed rank and applied it to pairwise-based action recognition. A Riemannian manifold-based optimization framework was exploited to learn the similarity matrix. Apart from learning a proper metric or a similarity matrix, the similarities of neighboring spatio-temporal points were explored in [12]. The space where the similarities lie in was discovered and mapped to another space with a fixed, smaller dimensionality. Thus, a novel similarity measurement between two image sequences was defined to form a similarity vector for further classification.

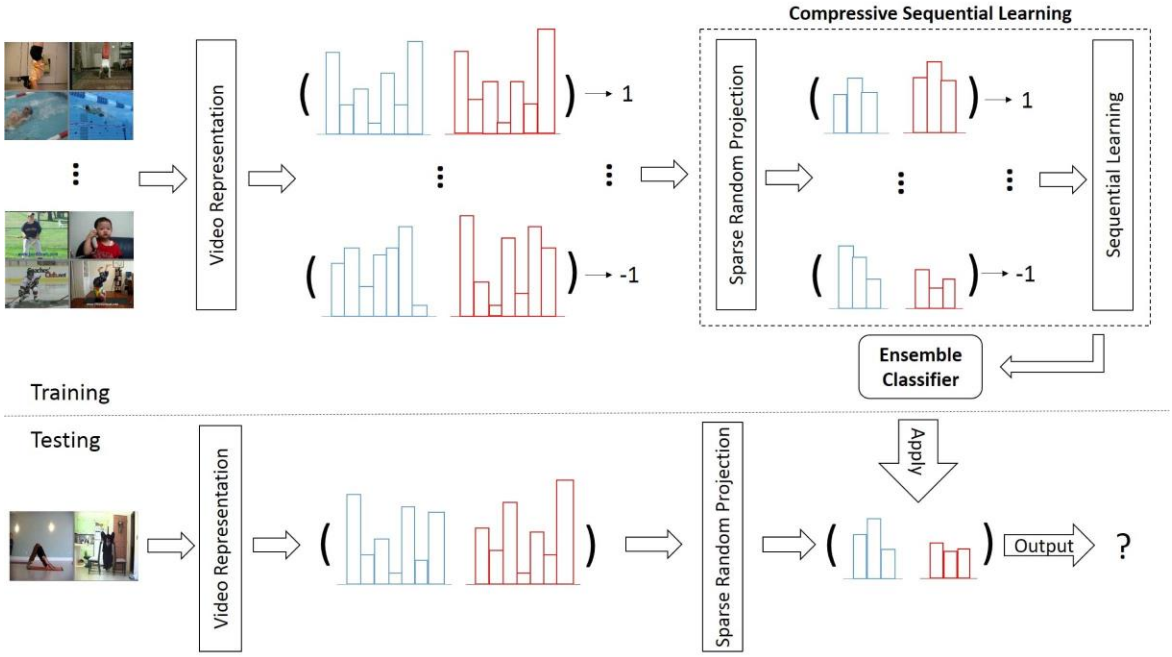


Fig. 2. The overall framework of Compressive Sequential Learning (CSL): Space-time features are first extracted from pair-wise action videos, followed by Bag-of-Features video representation. Two feature vectors extracted from a pair containing the same action are assigned label '+1' and '−1' means they belong to different action classes. Subsequently, sparse random projection is employed to project the high dimensional feature space to a lower one. Then our sequential learning is utilized to learn an optimized ensemble classifier, which can predict the labels for pair-wise data. Given a pair of testing videos, the learned classifier is applied to label the similarity of this pair.

As for the feature representation, most of the methods in the action similarity labeling task adopted those widely used in action recognition. For instance, the baseline results on ASLAN were obtained by leveraging Space-Time Interest Points (STIPs) followed by the Bag-of-Features (BoF) representation [25]. Kliper-Gross *et al.* [14] then proposed an enhanced feature encoding method named Motion Interchange Patterns (MIP) by capturing local changes in motion directions. MIP was able to decouple image edges from the motion and compensate for global camera motion. Hanani *et al.* [15] explored new variants of the MIP framework by incorporating gradient-based descriptors and made the framework achieve a better description of the motion's structure. In particular, two variants histMIP and DoGMIP were proposed, which replaced the original patch representation by histogram of gradient orientations and Difference of Gaussians representations, respectively. These new variants were further combined with other features, such as STIPs [26], dense trajectories [27], [28] and Motion Boundary Histograms (MBH) [29], to achieve better performance. Inspired by a recently proposed tool in text processing named word2vec [30] neural network, a piggyback representation [16] was proposed to encode the local spatial arrangements of keypoints by employing a variant of the word2vec method.

III. Methodology

In this section, we will introduce our Compressive Sequential Learning method in detail. The overall framework of CSL is shown in Fig. 2. The proposed method particularly aims at addressing the classification problem of pair-wise data, in which only the pair labels are available during training rather than the individual label of each single data point.

A. Problem Formulation

Given a set of N training pairs of videos, $\{(v_i^x, v_i^y)\}$, with feature representation $\{(\mathbf{x}_i, \mathbf{y}_i)\}$, where $\mathbf{x}_i$ and $\mathbf{y}_i \in \mathrm{R}^m$ and label $\{l_i\} \in \{+1, -1\}$, such that $l_i = +1$ if v_i^x and v_i^y contain the same action, and $l_i = -1$ otherwise. We hope to find a binary decision function (classifier) $H(\mathbf{x}, \mathbf{y})$ that is able to minimize the empirical loss on the training data:

$$L = \sum_{i=1}^{N} (H(\mathbf{x}_i, \mathbf{y}_i) = l_i) \quad (1)$$

Specifically, we adopt the boosting strategy similar to what is used in an AdaBoost classifier, aiming at iteratively learning the binary classifier to jointly minimize the empirical loss. Thus, the final classification function is the sign of a linear combination of non-linear weak learners:

$$H(\mathbf{x}, \mathbf{y}) = \mathrm{sign}\Big(\sum_{t=1}^{T} \alpha^{(t)} \mathrm{h}^{(t)}(\mathbf{x}, \mathbf{y})\Big), \quad (2)$$

where T is the number of iterations, $\alpha^{(t)}$ stands for the t-th vote corresponding to $\mathrm{h}^{(t)}$, and $\mathrm{sign}(x) = +1$ if $x \geq 0$, and $\mathrm{sign}(x) = -1$ otherwise. α adjusts the classification result of pair-wise data in each iteration. In addition, $\mathrm{h}(\mathbf{x}, \mathbf{y})$ is the weak learner whose definition and optimization will be introduced in Section III-C.

In the following sub-sections, we first introduce the boosting strategy for iteratively learning the final pair-wise

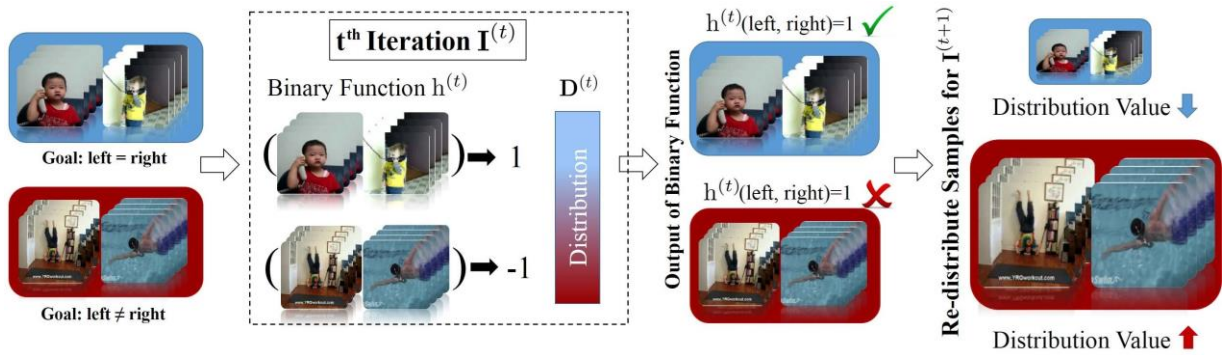


Fig. 3. The working flow of our sequential learning algorithm: In the t^{th} iteration, a binary function $h^{(t)}$ (i.e., weak learner) is learned based on labeled pair-wise video samples and the current distribution over all the samples is updated as well. In the next iteration, if a video pair is misclassified by the weak learner, it will be assigned a larger distribution value, and vice versa. Hence, the following weak learner tends to correct the errors of the preceding ones. The final ensemble classifier is a linear combination of these weak learners.

ensemble classifier. Subsequently, how to formulate and optimize the non-linear weak learner is addressed. Finally, to reduce the computational cost, a sparse random projection method is proposed to further boost the performance.

B. Sequential Learning via Boosting

In this section, a boosting-based optimization method is adopted to iteratively minimize Eq. (1). The overall framework can been seen from Fig. 3. Boosting was first proposed in the computational learning theory literature [20], [31], [32] and has since received much attention. AdaBoost, also known as Adaptive Boosting, has been extensively studied and applied to many research areas. The goal of AdaBoost is to construct an ensemble classifier that minimizes the training error. The ensemble classifier consists of a number of weak classifiers, each of which can obtain a weak hypothesis whose training error is better than chance (i.e., less than 50%) on the training data, weighted by a distribution $\mathbf{D}$. It is computationally infeasible to optimize the ensemble classifier to achieve the global minimal error on the training data. Therefore, a greedy optimization method is utilized to address this problem. The vote α corresponding to the weak learner and distribution $\mathbf{D}$ that weights the training data are updated iteratively to ensure the overall decrease of the training error for the ensemble classifier. Specifically, in each iteration, α is computed according to the weighted training error, and $\mathbf{D}$ is updated so that the pair-wise data points that are misclassified by the previous iteration are assigned a higher weight, whereas those points that are correctly classified are assigned a lower weight.

During the last few decades, a variety of AdaBoost variants have been proposed, including Discrete AdaBoost, Real AdaBoost [33], LogitBoost and Gentle AdaBoost [34]. In practice, we follow the spirit of Gentle AdaBoost and adopt it to optimize the global objective function in Eq. (1). Gentle AdaBoost is different from the widely used traditional AdaBoost with regard to how they use the weighted training error to update the vote α. Empirical evidence indicates that Gentle AdaBoost has similar or superior performance to the traditional AdaBoost on regular data, but presents its advantage on noisy data and is much more resistant to outliers. Meanwhile, due to the different rule of updating the vote α, Gentle AdaBoost is proved to be more efficient than the traditional AdaBoost.

C. Optimized Weak Learner

Unlike other weak learners used in the boosting strategy, our weak learner should be able to address the classification problem of pair-wise data. In the pair-wise setting, only the label of each pair is available instead of the label of any single data point. The weak learner $h(\mathbf{x}, \mathbf{y})$ adopted in the boosting algorithm should be capable of predicting whether two data points belong to the same class or not. Thereby, the objective of this weak learner is to minimize the least-squares loss on the training data:

$$\min \sum_{i=1}^{N} (h(\mathbf{x}_i, \mathbf{y}_i) - l_i)^2. \tag{3}$$

Intuitively, we define the weak learner as a product of two sign functions as follows:

$$h(\mathbf{x}, \mathbf{y}) = \text{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b)\ \text{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{y} + b), \tag{4}$$

where the function inside the sign has a standard form of a linear function with $\mathbf{w}$ and b as its parameters.

Particularly, by applying the weak learner of this form, the predictions of labels for pair-wise data will be $+1$ if two data points belong to the same class, and -1 otherwise. In the following, we will learn the parameters $\mathbf{w}$ and b via greedy optimization.

Note that

$$\begin{aligned}
&\min_{\mathbf{w},b} \sum_{i=1}^{N} (h(\mathbf{x}_i, \mathbf{y}_i) - l_i)^2 \\
&= \min_{\mathbf{w},b} \sum_{i=1}^{N} (h(\mathbf{x}_i, \mathbf{y}_i)^2 + l_i^2 - 2l_i\, h(\mathbf{x}_i, \mathbf{y}_i)) \\
&= \min_{\mathbf{w},b} \sum_{i=1}^{N} (1 + 1 - 2l_i h(\mathbf{x}_i, \mathbf{y}_i)).
\end{aligned} \tag{5}$$

The weak learner $h(\mathbf{x}, \mathbf{y})$ that minimizes Eq. (5) is also the one that maximizes the correlation of its output and the

data labels. Based on this fact, parameters $\mathbf{w}$ and b can be optimized by:

$$\begin{aligned}\max_{\mathbf{w},b} & \sum_{i=1}^{N} l_i h(\mathbf{x}_i, \mathbf{y}_i) \\ = \max_{\mathbf{w},b} & \sum_{i=1}^{N} l_i \, \mathrm{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b)\mathrm{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{y}_i + b) \\ = \max_{\mathbf{w}} & \sum_{i=1}^{N} l_i \mathrm{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i)\mathrm{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{y}_i), \end{aligned} \quad (6)$$

where $\mathbf{w} \rightarrow \mathbf{w}$ by simply adding b behind the last entry of $\mathbf{w}$ and $\mathbf{x}_i \rightarrow \mathbf{y}_i$ by adding 1 behind the last entry of $\mathbf{x}_i$ or $\mathbf{y}_i$.

Since the sign function in the weak learner is non-differentiable, Eq. (6) is still hard to solve. Here, the spectral relaxation trick [35], [36] is applied and the sign function is approximated by using its signed magnitude, i.e., $\mathrm{sign}(x) = x$. Therefore, by dropping the sign function, Eq. (6) can be relaxed as:

$$\begin{aligned}\max_{\mathbf{w}} & \sum_{i=1}^{N} l_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i)(\mathbf{w}^{\mathrm{T}}\mathbf{y}_i) \\ = \max_{\mathbf{w}} & \sum_{i=1}^{N} l_i \mathbf{w}^{\mathrm{T}}\mathbf{x}_i\mathbf{y}_i{}^{\mathrm{T}}\mathbf{w} \\ = \max_{\mathbf{w}} & \Big(\mathbf{w}^{\mathrm{T}} \sum_{i=1}^{N} l_i \mathbf{x}_i\mathbf{y}_i{}^{\mathrm{T}}\Big)\mathbf{w}. \end{aligned} \quad (7)$$

Moreover, $\mathbf{w}$ in Eq. (7) can be optimized by solving:

$$\max_{\mathbf{w}} \ \mathbf{w}^{\mathrm{T}}\mathrm{Z}\mathbf{w}, \quad (8)$$

where $\mathrm{Z} = \sum_{i=1}^{N} l_i \mathbf{x}_i \mathbf{y}_i{}^{\mathrm{T}}$.

It is worth noting that Eq. (8) turns into a standard eigen decomposition problem. Hence, the optimal solution for $\mathbf{w}$ can be obtained in a closed-form as the eigenvector of Z associated with the largest eigenvalue. Additionally, since $\mathbf{x}_i$ and $\mathbf{y}_i$ are equivalent to each other (i.e., $h(\mathbf{x}_i, \mathbf{y}_i) = h(\mathbf{y}_i, \mathbf{x}_i)$), Z can be transformed into its symmetric form such that $\mathrm{Z} = \sum_{i=1}^{N} l_i(\mathbf{x}_i\mathbf{y}_i{}^{\mathrm{T}} + \mathbf{y}_i\mathbf{x}_i{}^{\mathrm{T}})$ before solving $\mathbf{w}$, which simplifies the calculation.

Although the optimization is not global, it yields a practical approximation to the optimal solution to Eq. (6). Furthermore, owing to the iterative optimization of the global boosting strategy, a series of sub-optimal weak learners can also achieve promising classification performance after several iterations.

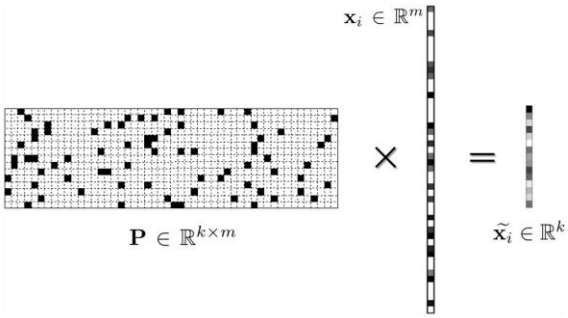


Fig. 4. Sparse Random Projection: $\mathbf{P}$ is the sparse random measurement matrix. Each row of $\mathbf{P}$ contains β non-zero entries (the white grids represent zero entries). By applying $\mathbf{P}$ on the original high-dimensional data $\mathbf{x}_i$, a more compact $\widetilde{\mathbf{x}}_i$ can be obtained with a low computational cost (i.e., totally $k\beta$ multiplications, where $k, \beta \pm m$).

video sequences containing actions "in the wild" (computing $\mathbf{w}$s and bs requires eigen decompositions of $(m+1)\times(m+1)$ matrices). In this sub-section, we focus on projecting the pair-wise data from the high dimensional space to a suitable low one. In particular, we adopt the spirit of sparse random projection. Fig. 4 shows our sparse random projection method, which has the following advantages: (1) simultaneously preserving the structure of the original high dimensional space; (2) obtaining compact low-dimensional feature representations; (3) balancing the components' variance of the original feature vector. In the following, we will first introduce random projection and further propose our sparse random projection method.

Given a set of data points $\{\mathbf{x}_i\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathrm{R}^m$, our goal is to find a random matrix $\mathbf{P} \in \mathrm{R}^{k\times m}$, which projects $\mathbf{x}_i$ to $\widetilde{\mathbf{x}}_i$ in a much lower dimensional space R^k ($k \pm m$) for speeding up the computation:

$$\widetilde{\mathbf{x}}_i = \mathbf{P}\mathbf{x}_i. \quad (9)$$

According to the Johnson-Lindenstrauss (JL) lemma [37], distances between the points in a vector space are very likely preserved if they are projected onto a randomly selected subspace with suitably high dimensions. Furthermore, as [38] proves, the random matrix satisfying the JL lemma also holds for the Restricted Isometry Property in compressive sensing.

Additionally, due to the simplification of our weak learner, computational costs in each iteration can be significantly reduced.

D. Sparse Random Projection

In the above sub-sections, a global and local optimization framework for dealing with the classification of pair-wise data has been proposed. Although the computational complexity can be reduced to a certain extent, it is still high due to the high dimensionality of complex features extracted from

Therefore, if the random matrix $\mathbf{P}$ satisfies the JL lemma, $\mathbf{P}$ is able to preserve the structure of the high dimensional space, namely the distances between all pairs of original data points. In other words, if $\mathbf{x}_i$ is compressive, such as audio or image data, it can be reconstructed with a minimum error from $\mathbf{x}_i$ with high probability. In addition to structure-preserving, mentioned in [18], $\mathbf{P}$ can balance the components' variance of $\mathbf{x}_i$ after performing the projection, which will directly benefit the optimization of weak learners.

In terms of conventional random projections [39], the entries of the measurement matrix $\mathbf{P}$ (denoted by $\{p_{ij}\}$) should be i.i.d. with a zero mean. A convenient way is to let p_{ij} follow a symmetric distribution about zero with a unit variance. Therefore, a typical measurement matrix $\mathbf{P}$ is drawn from a Gaussian ensemble randomly with $p_{ij} \sim N(0, 1)$. In this case, $\mathbf{P}$ is a dense matrix of real numbers. Multiplying $\mathbf{x}_i$ with $\mathbf{P}$ becomes

Algorithm 1 Compressive Sequential Learning

Input: A set of N training pairs of data $\{(\mathbf{x}_i, \mathbf{y}_i)\}$ with labels $\{l_i\} \in \{+1, -1\}$, where $l_i = +1$ means $\mathbf{x}_i$ and $\mathbf{y}_i$ correspond to the same class, and $l_i = -1$ otherwise; maximum number of iterations T.

Output: The ensemble classifier that minimizes Eq. (1).

Set the initial distribution $\mathbf{D}_i^{(t=1)} = \frac{1}{N}$ on all the training pairs of data $\{(\mathbf{x}_i, \mathbf{y}_i)\}$;
for $t = 1 : T$ **do**
  Construct sparse random projection matrix $\mathbf{P}_t$ so that $\{(\mathbf{x}_i, \mathbf{y}_i)\} \rightarrow \{(\widetilde{\mathbf{x}}_i, \widetilde{\mathbf{y}}_i)\}$, where $\widetilde{\mathbf{x}}_i = \mathbf{P}_t \mathbf{x}_i$ and $\widetilde{\mathbf{y}}_i = \mathbf{P}_t \mathbf{y}_i$;
  Train weak learner $\mathrm{h}^{(t)}(\mathbf{x}, \mathbf{y})$ defined in Eq. (4) by solving Eq. (8) based on all the training pairs of data $\{(\widetilde{\mathbf{x}}_i, \widetilde{\mathbf{y}}_i)\}$ under the current distribution $\mathbf{D}^{(t)}$;
  Predict the label of each pair $(\widetilde{\mathbf{x}}_i, \widetilde{\mathbf{y}}_i) \rightarrow \{+1, -1\}$ and compute the weighted error $E_w := [\sum_{\mathrm{h}^{(t)}(\widetilde{\mathbf{x}}_i, \widetilde{\mathbf{y}}_i) \neq l_i} \mathbf{D}_i^{(t)}]_{i=1}^N$;
  Calculate the vote $\alpha^{(t)}$ corresponding to this weak learner $\alpha^{(t)} := 1 - 2E_w$;
  Update the distribution $\mathbf{D}_i^{(t+1)} := \mathbf{D}_i^{(t)} \exp(-\alpha^{(t)} l_i \mathrm{h}^{(t)}(\mathbf{x}_i, \mathbf{y}_i))$ and re-normalize $\mathbf{D}^{(t+1)}$ so that $\sum_{i=1}^N \mathbf{D}_i^{(t+1)} = 1$;
end
return The ensemble classifier $\mathcal{H}(\mathbf{x}, \mathbf{y}) = \mathrm{sign}(\sum_{t=1}^T \alpha^{(t)} \mathrm{h}^{(t)}(\mathbf{x}, \mathbf{y}))$.

a non-trivial task in many practical computational environments. To address this problem, Achlioptas [37] proposed sparse random projections using the measurement matrix $\mathbf{P}$ with i.i.d. entries in

$$p_{ij} = \sqrt{s} \begin{cases} 1, & with\ probability\ \frac{1}{2s} \\ 0, & with\ probability\ 1 - \frac{1}{s} \\ -1, & with\ probability\ \frac{1}{2s} \end{cases} \quad (10)$$

where $s = 1$ or $s = 3$. In this case, the random measurement matrix $\mathbf{P}$ becomes very sparse and is very easy to compute. By setting $s = 3$, a three-fold speedup can be guaranteed since only one third of the data need to be computed. Furthermore, in [39], Li et al. showed that even larger values of s could be employed, such as $s = \sqrt{m}$, with little loss in accuracy. If $s = \sqrt{m}$, a $\sqrt{m}$-fold speedup can be achieved. In particular, we utilize a coefficient $\xi \in (\frac{1}{m}, 0.1]$ to set s and control the sparsity of the random projection matrix: $s = m \times \xi$. When $\xi = 0.1$, the random projection matrix $\mathbf{P}$ could be extremely sparse, i.e., a $\frac{m}{10}$-fold speedup can be achieved. However, in practice, s should not be set too aggressively to retain robustness.

E. Compressive Sequential Learning and Encoding

As aforementioned, our method incorporates the spirits of compressive sensing and sequential learning, and the focus of our method is classification of pair-wise data. Therefore, we name it Compressive Sequential Learning. The overall framework is outlined in Fig. 2, and the figure inside the dashed box is the core process of our CSL method. Furthermore, Algorithm 1 illustrates the detailed procedure of our method. It can be seen from Fig. 2 and Algorithm 1 that the boosting-trick is utilized as the global optimization technique. In each iteration, sparse random projection is first employed to reduce the dimensionality of the original feature space. Based on the low dimensional feature vectors from the training set, our weak learner is optimized by solving Eq. (8). After finding the optimized weak learner, we follow the formulation in Algorithm 1 to obtain and update the distribution $\mathbf{D}$ and vote α. Finally, we obtain a linear combination of non-linear optimized weak learners. By simply feeding to the sign function, an ensemble classifier is learned to measure similarities between pair-wise data points. Meanwhile, before applying the sign function, we obtain the confidence values and treat them as one dimensional feature vectors for pairs of data. And a linear SVM classifier is trained on these 1D vectors from the training set and then applied to predict labels for testing pairs of data. In this way, our CSL can be seen as an encoding technique.

for embedding pair-wise data to extremely compact codes, i.e., one dimensional feature vectors. This method, named Compressive Sequential Encoding (CSE), can be seen as an alternative to the CSL classifier. Both CSL and CSE are extensively evaluated in Section V.

F. Complexity Analysis

As illustrated in Algorithm 1, the cost of our CSL algorithm mainly consists of three parts within the "for" loop. The first part is for the construction of sparse random projection matrices. From Section III-D, we can conclude that its time complexity is $O(km)$, where m is the dimensionality of the original space and k is that of the projected low-dimensional space. The second part is for the multiplications of projection matrices and N data points $\mathbf{x}_i \in \mathrm{R}^m$. The time complexity of the matrix multiplication is $O(kmN)$. The last part is for the optimization of weak learners as shown in Eq. (8), which is a standard eigen-decomposition problem. Therefore, the time complexity of the last part is $O(k^3)$. In total, the time complexity of our CSL algorithm is $O(T \times (km + kmN + k^3))$, where T is the number of iterations of the global boosting framework. Furthermore, since $k \pm m$, the computational complexity of our method is equivalent to $O(TkmN)$.

In addition, we analyze the computational complexities of the state-of-the-art methods for action similarity learning [11], [13]–[15], which are shown in Table I. The complexity of OSSML [11] mainly consists of two parts, i.e., computing gradient of its objective function, and using cross validation to estimate the optimal values of its parameters. Specifically, the complexity of OSSML is $O(T\ km^2Nbc)$, where T is the number of iterations used to optimize the projection

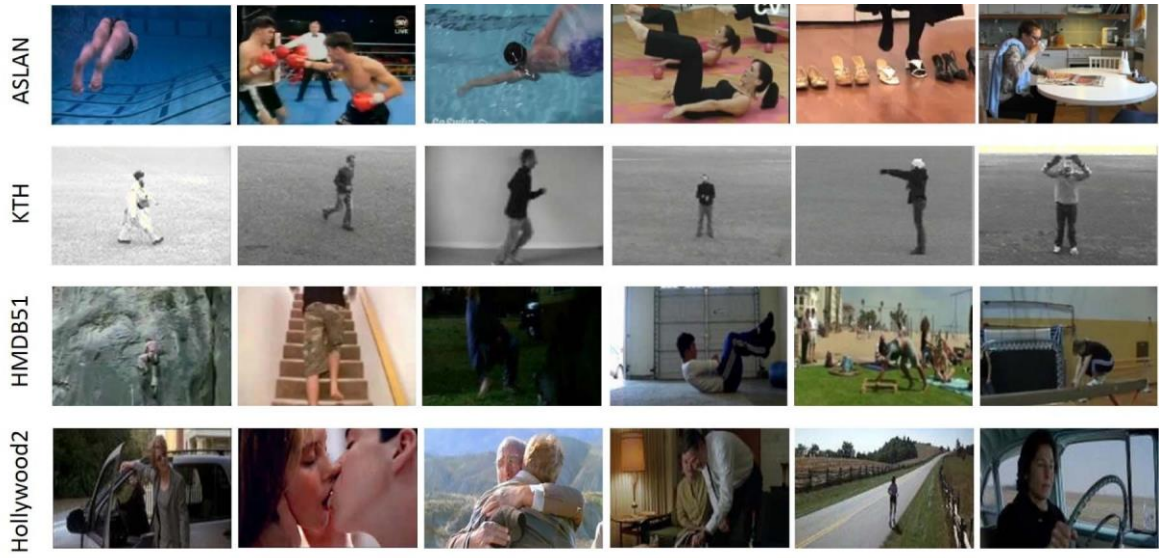


Fig. 5. Example frames of actions in the four datasets, i.e., ASLAN, KTH, HMDB51 and Hollywood2.

TABLE I

COMPUTATIONAL COMPLEXITIES OF DIFFERENT METHODS. T, T AND T DENOTE NUMBERS OF ITERATIONS OF DIFFERENT OPTIMIZATION ALGORITHMS USED BY DIFFERENT METHODS. PLEASE SEE TEXT FOR MORE DETAILS

CSL	[11]	[13]	[14] [15]
$O(TkmN)$	$O(T\ km\ Nbc)$	$O(T\ kmN)$	$O(T\ kmNbc)$

matrix, b is the number of values of the parameter tested in the cross validation process, c is the number of steps in the gradient descent method. Both [14] and [15] mainly focus on the feature representation problem and employ the metric learning method named CSML [40] for similarity learning, whose complexity is similar to that of OSSML. Specifically, the complexities of [14] and [15] are both $O(T\ kmNbc)$. Obviously, our CSL method is much more efficient than [11], [14], and [15]. In terms of LMDR [13], its complexity is mainly due to the quadratic optimization about the projection matrix, which is addressed by the stochastic sub-gradient descent method. Thus, the overall cost of LMDR is $O(T\ kmN)$, where T denotes the number of iterations of the stochastic sub-gradient descent method. As shown in [13], LMDR needs 10k iterations (i.e., T = 10,000) to achieve the optimal results, while only 50 iterations (i.e., T = 50) are enough for our method to achieve comparable results as discussed in Section V-D. Therefore, our method also needs less computational complexity than LMDR.

IV. DATASETS AND EXPERIMENTAL SETUP

To evaluate our approach, extensive experiments are conducted on a variety of action datasets, especially on challenging action datasets containing videos "in the wild". Specifically, four public datasets for action recognition (i.e., ASLAN, KTH, HMDB51 and Hollywood2) are used in our experiments. Fig. 5 illustrates some example frames of the action sequences in the datasets. From the figure, we can see that actions in ASLAN, HMDB51 and Hollywood2 are of extremely high complexity. The video sequences usually contain large variations in camera motion, scale, view, background, illumination, etc. Additionally, we evaluate our method on KTH, which contains videos under constrained environments. We conduct experiments on both constrained and realistic action datasets to illustrate the challenge posed when using videos "in the wild" compared to laboratory produced videos. In the following sub-sections, we describe each of the four datasets briefly. Since we focus on the problem of similarity labeling, the protocol for creating video pairs is also explained. Subsequently, spatial-temporal feature extraction and video representation methods are presented in detail, as well as the overall experimental protocol.

A. Datasets

1) ASLAN: The Action Similarity Labeling (ASLAN) Challenge is a recent dataset, which contains 3,631 action videos collected from the web with over 400 complex action classes. The goal of this dataset is to decide if two videos present the same action or not, following training with "same" and "not-same"-labeled video pairs. We follow the test protocol used in [9], in which the action pairs for training and test are mutually exclusive, i.e., actions in the test set are neither available during training nor from any action classes in the training set. In short, this protocol is known as the unseen pair matching. Specifically, ASLAN includes two views, one view (View-1) for model selection and algorithm development, and the other one (View-2) for reporting performance. We demonstrate the performance of our method on View-2, which consists of 10 mutually exclusive subsets. Each subset contains 600 action video pairs: 300 same and 300 not-same.

2) KTH: KTH [41] is an early action dataset acquired in a controlled environment. It contains six types of human actions, e.g., walking and jogging. Each action is performed several times by 25 subjects in four conditions. All action

sequences were taken over homogeneous backgrounds with a static camera with a 25fps frame rate. To validate our method in terms of unseen pair matching, we follow the setting in [9]. Specifically, six actions in KTH are divided into three mutually exclusive subsets.

3) HMDB51: HMDB51 [42] is known as a large video database for human motion recognition, containing 6849 video clips collected from various sources such as movies and YouTube. There are totally 51 action categories, each of which includes at least 101 video sequences. In our experiments, we use a subset of HMDB51, which consists of 2963 clips belonging to 19 action categories of general body movements. In order to create action pairs, we randomly choose five mutually exclusive subsets on the 19 action classes. Similar to the ASLAN dataset, each subset contains 600 action video pairs: 300 same and 300 not-same. Note that we use the original videos rather than the stabilized ones in our experiments.

4) Hollywood2: The Hollywood2 [43] dataset is composed of 1707 action samples with 12 action classes. All the videos are extracted from 69 different Hollywood movies. The dataset intends to provide a comprehensive benchmark for human action recognition in realistic and challenging settings. To create action pairs, we randomly divide 823 training samples into three mutually exclusive subsets, each of which contains 300 same action video pairs and 300 not-same ones.

B. Space-Time Video Representation

We use the same features employed in [9] to produce baseline results on the ASLAN dataset, which are sparse space-time features first proposed in [26]. These features can provide tolerance to clutter, occlusions and scale changes. Specifically, Space-Time Interest Points are detected using a space-time extension of the Harris operator and three types of local descriptors are computed to represent local motion and appearance features, i.e., Histogram of Oriented Gradients (HOG), Histogram of Optical Flow (HOF), and a composition of these two referred to as HNF. These local feature descriptors are computed for 3D video patches in the neighborhood of detected STIPs. Each video patch is divided into a grid with $3\times3\times2$ blocks. Then 4-bin HOG, 5-bin HOF, and 8-bin HNF descriptors are computed for each block. By concatenating the blocks, each detected STIP is referred to as a 72-element, 90-element or 144-element descriptor.

To represent a video sequence containing a set of such descriptors, we adopt the widely used spatial-temporal Bag-of-Features (BoF) model. A visual vocabulary is assembled by clustering all the descriptors of training videos. Each STIP of the video sequence is assigned a label corresponding to its closest visual word of the vocabulary. Therefore, a video sequence is the composite of a set of visual words. The frequency of occurrences of each word belonging to the video sequence is calculated and normalized into a histogram, which is the final BoF representation. In our experiments, one visual vocabulary is needed for each of our N experiments (N=10, 3, 5 and 3 for ASLAN, KTH, HMDB51 and Hollywood2, respectively). For each experiment, k-means (k=5000) clustering is employed to cluster a subset of 100k STIP descriptors randomly selected from the training data, and Euclidean distance is used to calculate the distances between STIP descriptors and 5000 visual words.

C. Experimental Protocol

In terms of reporting performance, we follow the testing protocol used in [9]. The final results are reported in an N-fold leave-one-out cross-validation scheme on the aforementioned four datasets. In each experiment, $N-1$ of the subsets are used for training, with the rest one used for testing. This results in N separate classification problems and the final performance is reported using the mean accuracy μ (see Eq. (11)) and the standard error of the mean S_E (see Eq. (12)) over the N subsets. Meanwhile, the ROC curve is constructed and the Area Under the Curve (AUC) is calculated for classifiers used on the N test subsets.

$$\mu = \frac{\sum_{n=1}^{N} P_n}{N}, \tag{11}$$

where P_n is the classification accuracy using subset n for testing.

$$S_E = \frac{\sigma}{\sqrt{N}}, \qquad \sigma = \sqrt{\frac{\sum_{n=1}^{N}(P_n - \mu)^2}{N-1}} \tag{12}$$

We do individual experiments using each of the three features as well as the combination of these features. Additionally, to allow for efficient computation, PCA is employed to project the high dimensional feature vectors to a reduced m dimensional space before applying our method (the value of m is

decided according to extensive experiments with different PCA dimensions). After that, sparse random projection is applied to further reduce the dimension and boost the performance. Due to the randomness of our sparse projection, each experiment is performed 200 times and the final results reported are averaged from the 200 runs.

V. Experimental Results

A. Comparison With Baselines

We first apply our CSL method on the four datasets and compare our results with the baseline results in terms of the accuracy, standard error and AUC (see Tables II-V for details). Specifically, the baseline method in [9] calculates 12 distances/similarities between feature vectors of the benchmark pairs. The specific formulations of these similarity functions can be seen from [9]. Due to space limitation, we manually select the best performance of these similarity functions and show the best results among them, which correspond to rows of "Baseline (Best Sim)" in the tables. Furthermore, the values of the 12 (dis-)similarities are concatenated into vectors and a linear SVM classifier is trained on the 12D vectors of pairs of action samples from the training set. As shown in [9], the combination of 12 similarity functions can often achieve the best classification results. Therefore, we compare the performance of our CSL method with the best results of the 12 similarities, as well as the concatenated results corresponding to rows of "Baseline (All Sim)" in the tables.

TABLE II

CLASSIFICATION PERFORMANCE ON ASLAN: ACCURACY ± STANDARD ERROR AND (AUC) (%), AVERAGED OVER THE 10-FOLDS. PLEASE SEE TEXT FOR MORE DETAILS

	HOG	HOF	HNF	ALL Descriptors
B l (B t S m)	58.55±0 0(61 53)	56.82±0.57(58.56)	58.87±0.89(62.16)	60.08±1 0 (63 2)
Baseline (All Sim)	59.78±0.82(63.20)	56.68±0.56(58.97)	59.47±0 66(63 30)	60.88±0.77(65.30)
CSML	60.15±0.70(64.20)	58.88±0.70(62.40)	60.52±0.60(64.30)	63.12±0 90(6 00)
OSSML	60.22±0.70(64.10)	58.05±0.80(60.70)	60.53±0.80(64.00)	62.52±0 0(66 60)
OSSML after CSML	60.63±0.60(**65.00**)	60.05±0 50(63 0)	60.83±0 0(65 10)	64.25±0.70(69.10)
CSL	**62.63**±0.62(64.62)	61.80±0.51(63.92)	62.92±0.66(**65.51**)	65.00±0.80(**69.91**)
CSE	**62.63**±0.62(64.63)	**61.82**±0.56(**64.34**)	**62.98**±0.65(65.28)	**65.68**±0.78(69.76)

TABLE III

CLASSIFICATION PERFORMANCE ON KTH: ACCURACY ± STANDARD ERROR AND (AUC) (%), AVERAGED OVER THE 3-FOLDS. PLEASE SEE TEXT FOR MORE DETAILS

	HOG	HOF	HNF	ALL Descriptors
B l (B t S m)	83.22±2.17(90.16)	85.94±0 53(90 31)	90.28±1.01(96.75)	90.00±1.76(96.46)
Baseline (All Sim)	80.11±1 06(91 93)	79.83±1 35(90 06)	86.67±4.22(97.47)	83.06±3.75(97.86)
CSL	**96.89**±1.16(**99.31**)	**99.39**±0.33(**99.86**)	**99.89** ±0.03(**99.94**)	**99.44**±0.30(**99.94**)
CSE	96.78±1.22(98.84)	99.33±0.37(99.64)	**99.89** ±0.03(**99.94**)	99.17±0.37(99.92)

TABLE IV

CLASSIFICATION PERFORMANCE ON HMDB51: ACCURACY ± STANDARD ERROR AND (AUC) (%), AVERAGED OVER THE 5-FOLDS. PLEASE SEE TEXT FOR MORE DETAILS

	HOG	HOF	HNF	ALL Descriptors
B l (B t S m)	58.83±1.92(62.30)	56.33±0.80(56.43)	56.63±0.72(56.68)	90.00±1.76(96.46)
Baseline (All Sim)	58.93±2.05(62.70)	55.90±0.85(56.74)	56.47±0.80(57.38)	83.06±3.75(97.86)
CSL	60.90±0.92(63.72)	57.50±0.35(**58.90**)	59.67±0 60(61 03)	**99.44**±0.30(**99.94**)
CSE	**61.07**±1.00(**63.75**)	**57.70**±0.38(58.47)	**60.17**±0.72(**61.26**)	99.17±0.37(99.92)

TABLE V

CLASSIFICATION PERFORMANCE ON HOLLYWOOD 2: ACCURACY ± STANDARD ERROR AND (AUC) (%), AVERAGED OVER THE 3-FOLDS. PLEASE SEE TEXT FOR MORE DETAILS

	HOG	HOF	HNF	ALL Descriptors
B l (B t S m)	55.94±0.62(58.00)	57.61±1.16(58.79)	57.67±0.44(60.62)	90.00±1.76(96.46)
Baseline (All Sim)	55.94±1.21(**58.18**)	58.17±2.51(**60.69**)	59.00±0.96(61.23)	83.06±3.75(97.86)
CSL	**57.83**±0 50(5 16)	58.78±0.31(59.43)	**60.94** ±0 65(61 51)	**99.44**±0.30(**99.94**)
CSE	**57.83**±0.50(57.97)	**58.89**±0.27(59.39)	60.72±0.63(**61.65**)	99.17±0.37(99.92)

As aforementioned in Section III, apart from our CSL method, we obtain the confidence values before applying the sign function in Eq. (2), which are 1D feature vectors for pairs of samples. Then a binary, linear SVM classifier is trained on these vectors and their corresponding binary labels to map unknown pairs to a binary decision of same/not-same. These results are shown on the last row of each table corresponding to "CSE".

In order to acquire the fusion results of the aforementioned three features, i.e., HOG, HOF and HNF, two strategies are employed. In terms of our CSL method, the weighted voting strategy is adopted. In particular, the final fused prediction of our CSL classifier is the sign of summation of confidence outputs of three individual CSL classifiers applied on these three features. As for the fusion results of the baseline and CSE, we follow [9] and utilize the stacking technique [44]. Specifically, the (dis-)similarity values of all three features (w.r.t. baseline) or the confidence outputs of CSL (w.r.t. CSE) are concatenated to form three dimensional vectors, each of which represents a pair of example videos. A linear SVM classifier is then trained on these vectors with their corresponding same or not-same labels. All the fusion results are shown in the last columns of the tables.

As shown in Tables II-V, we can conclude that, for either using CSL as a classification method or an encoding method, it can always outperform the manually-defined similarity functions and their combinations. In comparison of the best classification results between the baseline and our methods, a 4.80%, 9.44%, 2.40% or 3.22% improvement can be guaranteed on ASLAN, KTH, HMDB51 and Hollywood2, respectively. For the most challenging datasets, such as ASLAN and HMDB51, applying a linear SVM classifier on the confidence outputs of CSL can achieve slightly better performance. In most cases, the standard errors of our methods are lower than those of the baseline, showing the generalization of our classification method on different subsets. It is noteworthy that our method can achieve a significant improvement over the baseline on the KTH dataset, especially when using HOG and HOF features. This demonstrates the capability of our method on action similarity labeling under constrained environments.

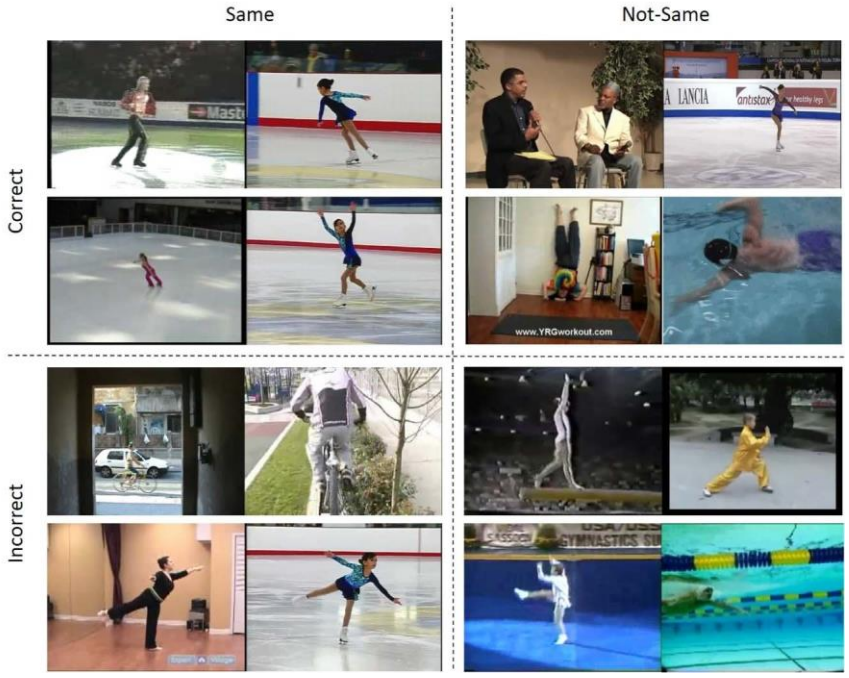


Fig. 6. The most confident results using CSL on ASLAN. The Same/Not-Same labels are the ground truth labels, and the Correct/Incorrect labels indicate whether our CSL method predicted correctly. For example, the bottom left quadrant displays same action pairs that were most confidently labeled as not-same.

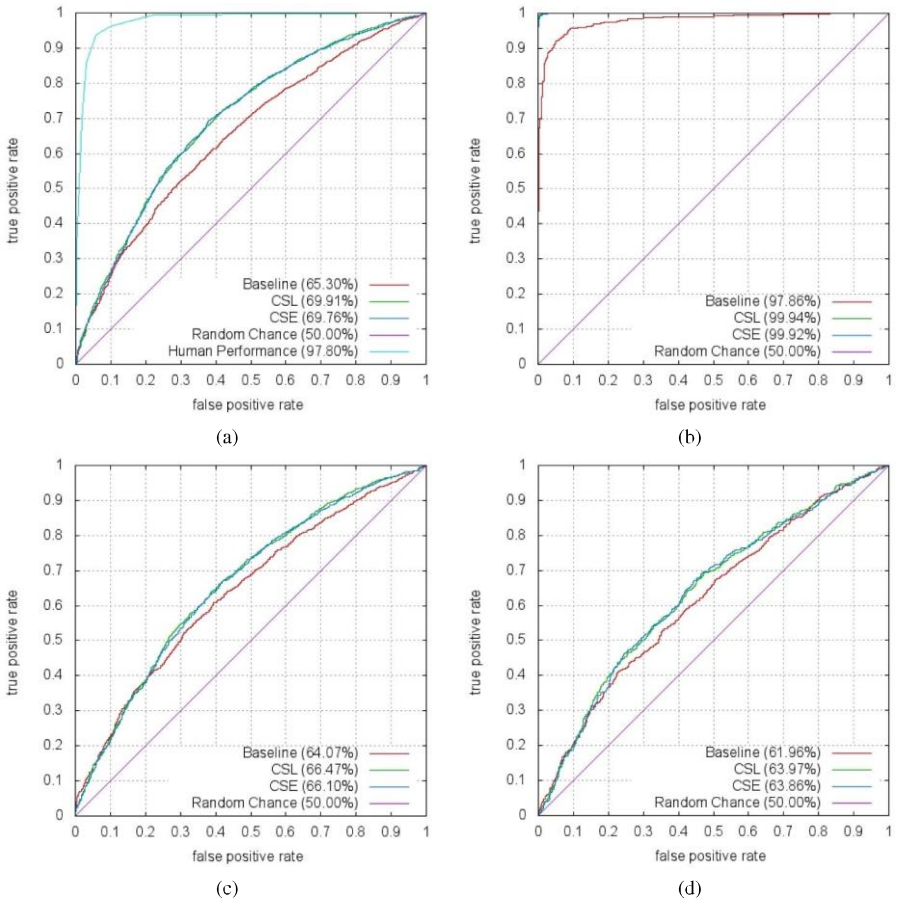


Fig. 7. ROC curves of different methods evaluated on the (a) ASLAN, (b) KTH, (c) HMDB51 and (d) Hollywood2 datasets.

In addition to demonstrating the accuracies, standard errors and AUC, ROC curves of the classification results of different approaches are shown in Fig. 7. The ROC curves are constructed based on the results of N classification experiments. As specified in Section IV-C, in each experiment, $N-1$ subsets are used for learning our CSL classifier and we

evaluate the results on the rest of the set. For each dataset, an ROC curve is constructed for all of its subsets together. In other words, the confidence output for each testing pair is computed in each experiment and all the outputs for N testing sets are concatenated together.

To gain further insight into our experimental results, we illustrate the most confident predictions on the ASLAN dataset made by our CSL method. Here, the level of confidence is measured by the value before applying the sign function in Eq. (2), i.e., the highest values correspond to the most confident predictions. Fig. 6 presents the most confident correct same and not-same predictions, and the most confident incorrect same and not-same ones. These images demonstrate the challenges and complexities of action similarity labeling in realistic scenes. Many of the mistakes result from misleading context. For instance, same action pairs are misclassified because of pose ambiguity and viewpoint variance. Meanwhile, not-same action pairs are mistaken as same ones due to a similar background and camera motion.

B. Comparison With State-of-the-Art

Since the ASLAN dataset has been collected as a benchmark focusing on the similarity labeling challenge, we further compare our method with the state-of-the-art methods.

Firstly, we evaluate our method in terms of similarity learning by comparing with the state-of-the-art metric learning methods (i.e., CSML and OSSML). The 3rd to 5th rows of Table II show the results of CSML, OSSML and "OSSML after CSML" which are taken from [11]. Initial PCA projection was done before applying the OSSML and CSML, and "OSSML after CSML" means applying the OSSML method with the matrix obtained by using the CSML algorithm as the initial projection. Obviously, "OSSML after CSML" can always achieve the best performance in comparison with its counterparts, i.e., CSML and OSSML. Specifically, our method achieves a 2.00%, 1.77%, 2.15% or 1.43% improvement over "OSSML after CSML" in accuracy with regard to different features and their combinations, respectively. Therefore, from the table, we can conclude that our learning algorithm is well suited for measuring the similarities between pairs of videos, even better than some state-of-the-art metric learning algorithms. Moreover, by applying very sparse random projection, our optimization algorithm is solved in the very low (less than 50) dimensional space, which can reduce the computational complexity significantly.

Secondly, for fairly comparing the performance of our method with the state-of-the-art methods on the ASLAN benchmark, we follow [13] and extract improved dense trajectories (IDT) with HOG, HOF and MBH concatenated 396-element descriptors using the code from Wang and Schmid [45]. After feature extraction, VLAD [18] is employed for video representation and its codebook size is fixed to 256 as commonly used in the literature [45]. Fig. 8 shows the comparison results with the state-of-the-art methods. Apart from [13], both our CSL and CSE methods outperform other state-of-the-art methods. The accuracy of our method is 0.57% less than that of [13]. It should be noted that [13] utilized a dimensionality reduction method to compress high-dimensional feature vectors and their best results were obtained by using 59% compression ratio, which reduced the original dimensionality to 874. However, as aforementioned, our method is performed on feature vectors with less than 50 dimensions. Therefore, the very little loss in accuracy is acceptable with large enhancement in computational efficiency.

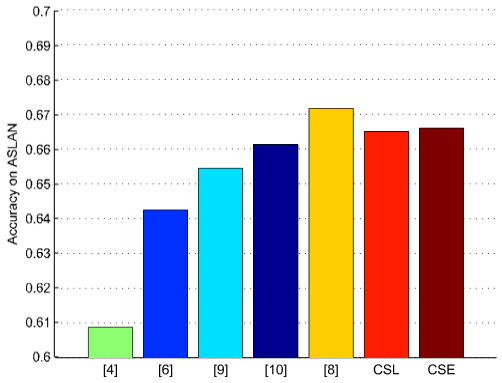


Fig. 8. Comparison with the state-of-the-art methods on the ASLAN benchmark.

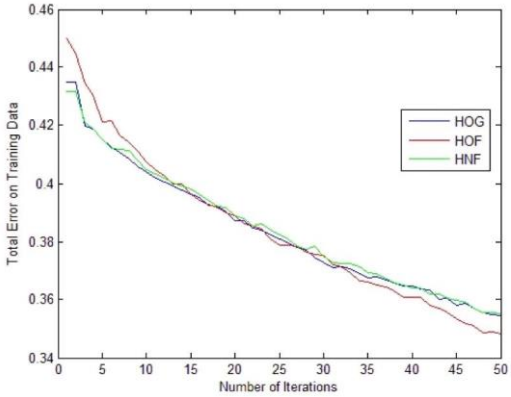
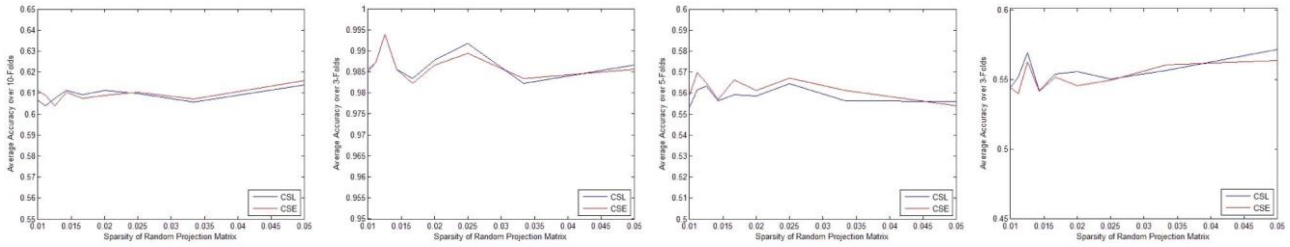


Fig. 9. Total errors on the training data of the ASLAN dataset using different feature descriptors.

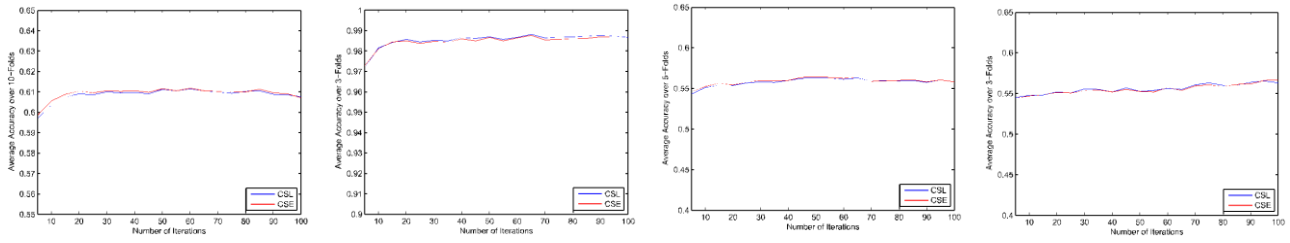
C. Training Error of CSL

Since the boosting strategy is employed in our sequential learning process, the total error on the training data should be evaluated. In particular, we demonstrate the classification error on the ASLAN dataset by using different feature representations, i.e., HOG, HOF and HNF. As aforementioned, there are totally 10 experiments. A training error illustrated in Fig. 9 is averaged over training errors calculated from each of the 10 training sets. As shown in Fig. 9, although there exist several rising points, the overall total error on the training data decreases gradually with the increase of the number of iterations. This downward trend indicates the effectiveness of our boosting-based sequential learning algorithm. As the number of iterations increases, we can learn a pair-wise classification model that minimizes the error on the training sets step by step. Owing to this fact, we are able to label the similarity between pairs of testing examples effectively.



(a) (b) (c) (d)

Fig. 10. Average accuracies of CSL and CSE with different levels of sparsity for random projection matrices by using fused features on the four datasets: (a) ASLAN, (b) KTH, (c) HMDB51 and (d) Hollywood2.



(a) (b) (c) (d)

Fig. 11. Average accuracies of CSL and CSE with different iterations by using fused features on the four datasets: (a) ASLAN, (b) KTH, (c) HMDB51 and (d) Hollywood2.

D. Evaluation of Parameters

To further study the influence of several parameters on the performance of CSL and CSE, we evaluate our methods based on different settings of parameters, i.e., number of iterations w.r.t boosting and sparsity of the random projection matrix. Firstly, as shown in Fig. 10, we demonstrate the performance by using different levels of sparsity ξ for random projection matrices. Fig. 10 shows the average accuracies using the combination of three feature descriptors on all the four datasets, respectively. As shown in Fig. 10, there is less than 2% difference between a lowest average accuracy and a highest one on all the four datasets. Therefore, we can conclude that the sparseness of the random projection matrix has little influence on the final performance. Particularly, we can employ a very sparse measurement matrix to project the high dimensional data, which will result in a significant speed-up in computational efficiency. Secondly, the performance of our method is illustrated with the increasing iterations of the boosting process. From Fig. 11, we can see that only five iterations can already achieve acceptable performance. In most cases, more iterations usually guarantee a higher accuracy. Specifically, the performance grows rapidly when the number of iterations increases to 20, especially on the ASLAN and KTH datasets. In addition, there is little variation in terms of the accuracy after 50 iterations on most datasets. Although more iterations will result in higher accuracies, this will in turn cost more computing time. Therefore, in our experiments, we set the number of iterations as 50 to ensure a good balance between the performance and the computational efficiency.

E. Evaluation of the Overall Framework

As our framework consists of two major steps: dimensionality reduction via sparse random projection and sequential learning/encoding via boosting, we further validate the choices for each step by replacing our method with several other methods in each step. Firstly, for dimensionality reduction, we utilize two methods, i.e., PCA and PCA with orthogonal random rotation (ORR) as suggested by Jégou *et al.* [18], instead of our sparse random projection (SRP). The original feature space is reduced to 50 dimensions after applying these two methods. Then we utilize our sequential learning (SL) or sequential encoding (SE) method based on the low dimensional space. Note that our sequential learning (SL) or encoding (SE) method can be seen as an alternative to each other. If we apply the sign function, our method can be seen as a classification method specifically developed for pair-wise data, otherwise our method can be regarded as a compact encoding method. Therefore, for fairly validating the choice for the second step, we first reduce the original dimension using our SRP method and conduct the experiments from two perspectives. Firstly, we replace our choice for this step with several other classification methods, i.e., Gentle AdaBoost [34], Real AdaBoost [33], SVM [46] and K Nearest Neighbours (KNN), for comparisons with our CSL method. Secondly, we utilize Gentle AdaBoost and Real AdaBoost as encoding methods by discarding the sign function, followed by a linear SVM classifier for the final classification, and this is comparable to our CSE method.

When validating the choice for the second step, our pair-wise sequential learning/encoding method is substituted by other methods, so we need to represent the similarity between a pair of video clips. Here, we follow [14] and [15] and transform representations of each pair, i.e., $\mathbf{x}_i$ and $\mathbf{y}_i$ to a single vector by performing point-wise multiplications of the two vectors ($\mathbf{x}_i . * \mathbf{y}_i$, where $.*$ denotes point-wise multiplication). All the results by using different features are demonstrated in Table VI and VII, and the last column shows the fusion results based on the aforementioned weighted voting or stacking technique [44]. It can be seen that, our choices (i.e., SRP, SL and SE) consistently outperform different choices of other

TABLE VI

EVALUATION OF THE FIRST STEP (i.e., DIMENSIONALITY REDUCTION) OF OUR OVERALL FRAMEWORK ON THE ASLAN BENCHMARK: ACCURACY ± STANDARD ERROR AND (AUC) (%), AVERAGED OVER THE 10-FOLDS. PLEASE SEE TEXT FOR MORE DETAILS

	HOG	HOF	HNF	ALL Descriptors
PCA+SL	55.88±1.10(55.43)	55.23±0.95(53.47)	57.78±0.2 (5 0)	58.53±1.07(59.12)
PCA+ORR+SL	57.25±0.96(57.11)	54.75±0.99(53.41)	58.25±0.95(58.37)	58.55±0.79(59.64)
CSL(SRP+SL)	**62.63**±0.62(**64.62**)	**61.80**±0.51(**63.92**)	**62.92**±0.66(**65.51**)	**65.00**±0.80(**69.91**)
PCA+SE	55.50±1.28(55.77)	55.05±0.94(53.43)	58.22±0 26(5 33)	58.95±0 6(61 03)
PCA+ORR+SE	58.02±0.89(57.72)	54.75±0.98(53.32)	58.25±1.03(58.02)	59.03±0.67(61.36)
CSE(SRP+SE)	**62.63**±0.62(**64.63**)	**61.82**±0.56(**64.34**)	**62.98**±0.65(**65.28**)	**65.68**±0.78(**69.76**)

"PCA+SL" or "PCA+ORR+SL": PCA or PCA+ORR is employed for dimensionality reduction, followed by our sequential learning method; "PCA+SE" or "PCA+ORR+SE": PCA or PCA+ORR is employed for dimensionality reduction, followed by our sequential encoding method.

TABLE VII

EVALUATION OF THE SECOND STEP (i.e., SEQUENTIAL LEARNING/ENCODING) OF OUR OVERALL FRAMEWORK ON THE ASLAN BENCHMARK: ACCURACY ± STANDARD ERROR AND (AUC) (%), AVERAGED OVER THE 10-FOLDS. PLEASE SEE TEXT FOR MORE DETAILS

	HOG	HOF	HNF	ALL Descriptors
SRP+Gentle AdaBoost	57.00±0.23(59.47)	55.90±0.41(56.72)	57.28±0.53(59.43)	59.47±0.41(60.70)
SRP+Real AdaBoost	57.42±0.54(57.91)	55.43±0.35(57.45)	56.58±0.47(57.83)	58.97±0.68(60.37)
SRP+SVM	61.42±0.77(63.80)	59.12±0 53(61 16)	61.38±0.69(63.82)	63.68±0.67(66.51)
SRP+KNN	61.30±0 53(59 05)	59.33±0.38(56.72)	61.40±0 51(59 31)	64.05±0.41(66.75)
CSL(SRP+SL)	**62.63**±0.62(**64.62**)	**61.80**±0.51(**63.92**)	**62.92**±0.66(**65.51**)	**65.00**±0.80(**69.91**)
SRP+Gentle AdaBoost Encoding	57.00±0.20(59.47)	55.87±0.44(56.61)	57.22±0 53(59 56)	59.90±0.44(63.38)
SRP+Real AdaBoost Encoding	57.43±0 5 (5 11)	55.33±0.38(57.42)	56.62±0.51(57.83)	59.43±0.53(62.14)
CSE(SRP+SE)	**62.63**±0.62(**64.63**)	**61.82**±0.56(**64.34**)	**62.98**±0.65(**65.28**)	**65.68**±0.78(**69.76**)

The proposed sparse random projection (SRP) is applied for dimensionality reduction. The first four rows show results by using different classification methods (i.e., Gentle AdaBoost, Real AdaBoost, SVM and KNN); the 6th and 7th rows present results by using Gentle AdaBoost and Real AdaBoost as encoding methods, respectively.

methods for different steps, which proves the superiority of our overall framework.

VI. CONCLUSION

In this paper, we consider the action similarity labeling challenge in realistic scenarios. A novel method named Compressive Sequential Learning (CSL) has been presented for learning the similarities between pairs of actions. A very sparse random projection based on the Restricted Isometry Property in the compressive sensing theory is employed for projecting data points to a low dimensional vector space, where the distances between the points are preserved. We then address the similarity measurement as a greedy optimization problem, by incorporating a boosting trick globally and optimizing non-linear weak classifiers for pair-wise data locally. Our method has been systematically evaluated on the ASLAN, KTH, HMDB51 and Hollywood2 action datasets. The results demonstrate the superiority of our method over other state-of-the-art methods. For future work, we plan to further improve the optimization algorithm, e.g., to employ more sophisticated weak learners, and investigate its applications to other related areas.

REFERENCES

[1] I. Everts, J. C. van Gemert, and T. Gevers, "Evaluation of color spatio-temporal interest points for human action recognition," *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1569–1580, Apr. 2014.

[2] M. Yu, L. Liu, and L. Shao, "Structure-preserving binary representations for RGB-D action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, doi: 10.1109/TPAMI.2015.2491925.

[3] K. Guo, P. Ishwar, and J. Konrad, "Action recognition from video using feature covariance matrices," *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2479–2494, Jun. 2013.

[4] F. Zhu and L. Shao, "Weakly-supervised cross-domain dictionary learning for visual recognition," *Int. J. Comput. Vis.*, vol. 109, nos. 1–2, pp. 42–59, Aug. 2014.

[5] A. Oikonomopoulos, I. Patras, and M. Pantic, "Spatiotemporal localization and categorization of human actions in unsegmented image sequences," *IEEE Trans. Image Process.*, vol. 20, no. 4, pp. 1126–1140, Apr. 2011.

[6] L. Shao, L. Liu, and M. Yu, "Kernelized multiview projection for robust action recognition," *Int. J. Comput. Vis.*, 2015, doi: 10.1007/s11263-015-0861-6.

[7] L. Liu, L. Shao, X. Li, and K. Lu, "Learning spatio-temporal representations for action recognition: A genetic programming approach," *IEEE Trans. Cybern.*, doi: 10.1109/TCYB.2015.2399172.

[8] L. Shao, X. Zhen, D. Tao, and X. Li, "Spatio-temporal Laplacian pyramid coding for action recognition," *IEEE Trans. Cybern.*, vol. 44, no. 6, pp. 817–827, Jun. 2014.

[9] O. Kliper-Gross, T. Hassner, and L. Wolf, "The action similarity labeling challenge," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 615–621, Mar. 2012.

[10] L. Cheng, "Riemannian similarity learning," in *Proc. 30th Int. Conf. Mach. Learn.*, vol. 28. 2013, pp. 540–548.

[11] O. Kliper-Gross, T. Hassner, and L. Wolf, "One shot similarity metric learning for action recognition," in *Proc. 1st Int. Conf. Similarity-Based Pattern Recognit.*, 2011, pp. 31–45.

[12] I. Kotsia and I. Patras, "Exploring the similarities of neighboring spatiotemporal points for action pair matching," in *Proc. 11th Asian Conf. Comput. Vis.*, 2013, pp. 624–635.

[13] X. Peng, Y. Qiao, Q. Peng, and Q. Wang, "Large margin dimensionality reduction for action similarity labeling," *IEEE Signal Process. Lett.*, vol. 21, no. 8, pp. 1022–1025, Aug. 2014.

[14] O. Kliper-Gross, Y. Gurovich, T. Hassner, and L. Wolf, "Motion interchange patterns for action recognition in unconstrained videos," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 256–269.

[15] Y. Hanani, N. Levy, and L. Wolf, "Evaluating new variants of motion interchange patterns," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2013, pp. 263–268.

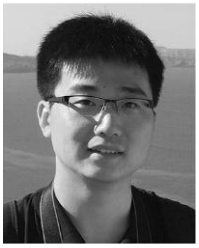
[16] L. Wolf, Y. Hanani, and T. Hassner, "A piggyback representation for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2014, pp. 520–525.

[17] E. J. Candès and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.

[18] H. Jégou, M. Douze, C. Schmid, and P. Perez, "Aggregating local descriptors into a compact image representation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2010, pp. 3304–3311.

[19] T. Trzcinski, M. Christoudias, P. Fua, and V. Lepetit, "Boosting binary keypoint descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2874–2881.

[20] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.
[21] L. Liu, M. Yu, and L. Shao, "Projection bank: From high-dimensional data to medium-length binary codes," in *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 2821–2829, Dec. 2015.
[22] L. Liu and L. Shao, "Sequential compact code learning for unsupervised image hashing," *IEEE Trans. Neural Netw. Learn. Syst.*, doi: 10.1109/TNNLS.2015.2495345.
[23] L. Liu, M. Yu, and L. Shao, "Unsupervised local feature hashing for image similarity search," *IEEE Trans. Cybern.*, doi: 10.1109/TCYB.2015.2480966.
[24] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the Fisher kernel for large-scale image classification," in *Proc. 11th Eur. Conf. Comput. Vis. (ECCV)*, 2010, pp. 143–156.
[25] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2008, pp. 1–8.
[26] I. Laptev, "On space-time interest points," *Int. J. Comput. Vis.*, vol. 64, nos. 2–3, pp. 107–123, 2005.
[27] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Action recognition by dense trajectories," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2011, pp. 3169–3176.
[28] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *Int. J. Comput. Vis.*, vol. 103, no. 1, pp. 60–79, May 2013.
[29] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Proc. 9th Eur. Conf. Comput. Vis.*, 2006, pp. 428–441.
[30] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. ICLR Workshop*, 2013.
[31] R. E. Schapire, "The strength of weak learnability," *Mach. Learn.*, vol. 5, no. 2, pp. 197–227, 1990.
[32] Y. Freund, "Boosting a weak learning algorithm by majority," *Inf. Comput.*, vol. 121, no. 2, pp. 256–285, Sep. 1995.
[33] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Mach. Learn.*, vol. 37, no. 3, pp. 297–336, 1999.
[34] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 2000.
[35] S.-F. Chang, W. Liu, J. Wang, R. Ji, and Y.-G. Jiang, "Supervised hashing with kernels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 2074–2081.
[36] J. Wang, S. Kumar, and S.-F. Chang, "Sequential projection learning for hashing with compact codes," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 1127–1134.
[37] D. Achlioptas, "Database-friendly random projections: Johnson–Lindenstrauss with binary coins," *J. Comput. Syst. Sci.*, vol. 66, no. 4, pp. 671–687, Jun. 2003.
[38] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approx.*, vol. 28, no. 3, pp. 253–263, Dec. 2008.
[39] P. Li, T. J. Hastie, and K. W. Church, "Very sparse random projections," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2006, pp. 287–296.
[40] H. V. Nguyen and L. Bai, "Cosine similarity metric learning for face verification," in *Proc. 10th Asian Conf. Comput. Vis.*, 2011, pp. 709–720.
[41] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach," in *Proc. IEEE/IAPR Int. Conf. Pattern Recognit.*, Aug. 2004, pp. 32–36.
[42] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Nov. 2011, pp. 2556–2563.
[43] M. Marszałek, I. Laptev, and C. Schmid, "Actions in context," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 2929–2936.
[44] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, 1992.
[45] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 3551–3558.
[46] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm



Jie Qin received the B.S. degree from the School of Computer Science and Engineering, Beihang University, Beijing, China, in 2011, where he is currently pursuing the Ph.D. degree with the Laboratory of Intelligent Recognition and Image Processing. From 2014 to 2015, he was a Visiting Researcher with the Department of Electronic and Electrical Engineering, The University of Sheffield, Sheffield, U.K. His research interests include computer vision, machine learning, and in particular, on action recognition.

Li Liu received the B.Eng. degree in electronic information engineering from Xi'an Jiaotong University, Xi'an, China, in 2011, and the Ph.D. degree from the Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield, U.K., in 2014. He is currently a Research Fellow with the Department of Computer Science and Digital Technologies, Northumbria University, Newcastle upon Tyne, U.K. His research interests include computer vision, machine learning, and data mining.

Zhaoxiang Zhang (M'08–SM'15) received the B.S. degree in electronic science and technology from the University of Science and Technology of China, in 2004, and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, in 2009. In 2009, he joined the School of Computer Science and Engineering, Beihang University, as an Assistant Professor and then an Associate Professor. In 2015, he joined the Institute of Automation, Chinese Academy of Sciences, as a Full Professor. Specifically, he is recently focusing on brain-inspired vision and humanlike learning. He has published around 80 papers in reputable journals and conferences. His research interests include computer vision, pattern recognition, and machine learning. He is the Associate Editor of *Neurocomputing*, involved on the Editorial Board of the *Frontiers of Computer Science*, the Program Committee Member of over ten international conferences, and the Reviewer of over 20 international journals. He has been granted several awards, including the MOE New Century Excellent Talents and the Beijing Youth Talents.

Yunhong Wang (M'98) received the B.S. degree from Northwestern Polytechnical University, in 1989, and the M.S. and Ph.D. degrees from the Nanjing University of Science and Technology, in 1995 and 1998, respectively, all in electronics engineering. She was with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China, from 1998 to 2004. Since 2004, she has been a Professor with the School of Computer Science and Engineering, Beihang University, Beijing, where she is also the Director of the Laboratory of Intelligent Recognition and Image Processing, Beijing Key Laboratory of Digital Media. Her research interests include biometrics, pattern recognition, computer vision, data fusion, and image processing. She is a member of the IEEE Computer Society.

Ling Shao (M'09–SM'10) is currently a Professor with the Department of Computer Science and Digital Technologies, Northumbria University, Newcastle upon Tyne, U.K. He was a Senior Lecturer with the Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield, U.K., from 2009 to 2014, and a Senior Scientist with Philips Research, Eindhoven, The Netherlands, from 2005 to 2009. His research interests include computer vision, image/video processing, and machine learning. He is an Associate Editor of IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON CYBERNETICS the, and several other journals. He is a fellow of the British Computer Society and the Institution of Engineering and Technology.