Improving Human Action Recognition using Score Distribution and Ranking

Minh Hoai[1,2] and Andrew Zisserman[1]

[1]Visual Geometry Group, Dept. Engineering Science, University of Oxford.
[2]Department of Computer Science, Stony Brook University.

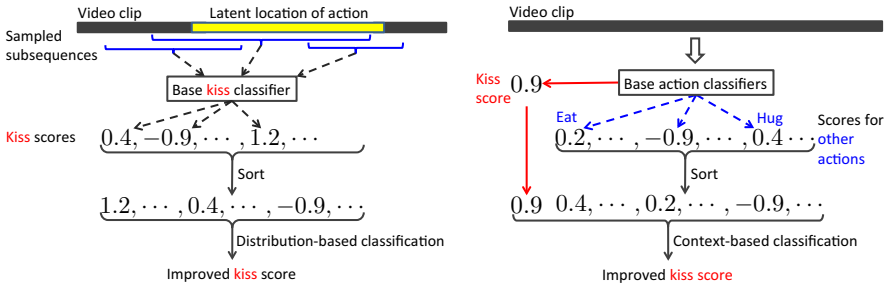
Abstract. We propose two complementary techniques to improve the performance of action recognition systems. The first technique addresses the temporal interval ambiguity of actions by learning a classifier score distribution over video subsequences. A classifier based on this score distribution is shown to be more effective than using the maximum or average scores. The second technique learns a classifier for the relative values of action scores, capturing the correlation and exclusion between action classes. Both techniques are simple and have efficient implementations using a Least-Squares SVM. We demonstrate that taken together the techniques exceed the state-of-the-art performance by a wide margin on challenging benchmarks for human actions.

1 Introduction

Action recognition is an active research area. Recent research focuses on realistic datasets collected from TV shows [1], movies [2, 3], and web videos [3]. However, there exists an inherent ambiguity for actions in realistic data: when does an action begin and end? Consider the action "handshake." When is the precise moment that two people begin to shake hands? When they start extending their hands or when the two hands are in contact? Moreover, when does the action end? For TV shows and movies, this is even more difficult to determine due to the existence of shot boundaries. Should we consider the action has ended when the camera cuts to a different shot? What if the action extends over multiple shots? Many works in action recognition ignore this temporal ambiguity problem, and simply classify the entire video clip, e.g., [1–4]. However, as shown in [5, 6], refining the temporal extent of actions can improve the recognition performance.

So, how should we handle this ambiguity? A possible approach is to treat the temporal extent of an action as a latent variable (e.g., [7–11]) and embed the problem in a Multiple Instance Learning (MIL) framework such as [12, 13]. MIL [14] is a generalization of supervised classification in which class labels are associated with sets of samples (called bags) instead of individual samples (called instances). For action recognition, a bag is a video clip, and the instances in each bag can be generated by varying the temporal extent of sequences within the clip (e.g., all subsequences of a video).

However, the efficacy of MIL for solving the temporal ambiguity of action in video is unproven. Moreover, the underlying assumptions and design principles of MIL are often violated. For example, the MIL algorithms of [15, 16] implicitly



(a) Using subsequence-score distribution (b) Using relative class scores

Fig. 1. Complementary techniques to improve recognition performance. (a): improved action score is computed based on the score distribution of video subsequences. (b): an improved action score is computed based on the relative action scores

assume that instances are drawn i.i.d. (independently and identically distributed) from some distribution and randomly placed into bags. This is not valid for action recognition where there exist temporal correlation between subsequences of a video. Another basic assumption of MIL is that a bag is positive if at least one of its instances is positive. This leads to a practical procedure adopted by most MIL algorithms (e.g., MI-SVM [12]): an instance classifier is learned (or iteratively learned) and the maximum classifier score is used to find the positive instance of a bag (for prediction during testing or for iterative update during training). In practice using only the maximum score is not robust, especially in action recognition where the state-of-the-art classifiers are far from perfect [4].

Empirical evidence also suggests the inadequacy of using the maximum subsequence score for video classification. In many test cases of our experiments, which will be seen in Sec. 5, we observe that using the average score of video subsequences is better than using the maximum score. In another context of MIL beyond action recognition, this observation has also been reported [17]. For example, [18] observed that MIL algorithms could be outperformed by a simple approach that used supervised learning together with label inheritance, which simply assigned the bag label to its instances.

On the other hand, the mean is not always better than the max. The mean works well when the influence of negative instances in positive bags is low. This does not hold if the percentage of positive instances in a positive bag is small. In that situation, the mean score is inferior to the maximum score, which will be empirically confirmed in Sec. 3.2 and Sec. 5. Note, if we consider the instances of a bag as the output of a generative process with a latent variable, the comparison between the max and the mean is equivalent to the comparison between the maximum and marginal likelihoods. Others have used measures between the mean and the max, or defined set kernels for bags of multiple instances [19–27].

In this paper, we propose to use the *distribution* of classifier scores, rather than just the max or mean, to improve action recognition. We first train a base classifier and then use the scores of *all* subsequences of a video clip to predict its label (Fig. 1(a)). The scores of video subsequences are ordered and

combined using a weight vector, which can be learned from the same set of training data that is used to train the base classifier. We will show that the ordered score distribution preserves more information than both extreme (i.e., max) and summary (i.e., mean) statistics, and it is more effective in practice.

Complementary to using the score distribution, this paper addresses another fundamental drawback of many current action recognition systems that action classes are recognized independently. In the second part of the paper, we propose an approach to learn the correlation and exclusion between action classes. In particular, we learn a classifier that reweights the action score based on the ordered scores of other classes, as illustrated in Fig. 1(b).

The rest of this paper is structured as follows. Section 2 reviews related prior work. Section 3 shows that using the score distribution is more effective than using the maximum and average scores (and shed some light on the poor performance of using maximum score for action recognition and MIL in general). Section 4 presents a learning formulation to capture the correlation and exclusion of the actions to improve the performance of action classifiers. Subsection 5.1 details the experimental setup on Hollywood2, TVHI, and HMDB51, which are among the most challenging datasets for human action recognition. Subsection 5.1 also describes another technical contribution of our work, which is the use of data augmentation to obtain stronger performance. Since a video and its left-right mirrored video depict the same action, we propose to learn a classifier that is invariant to flipping by data augmentation. This is related to several works that use virtual samples [28, 29]. Subsection 5.2 demonstrates that our proposed techniques significantly improve the state-of-the-art performance. This is achieved using standard, publicly available, features and encodings.

2 Reviews of Related Work and Least-Squares SVM

The need for considering the temporal extent of actions in training or testing has been studied before. Duchenne *et al.* [6] and Satkin & Hebert [5] observed that temporal boundaries of actions in training videos are not precisely defined in practice. They proposed methods to crop training videos using discriminative clustering and cross-validation performance. Nowozin *et al.* [30] and Nguyen *et al.* [31] presented algorithms that sought discriminative subsequences in video. Yuan *et al.* [32] proposed a branch-and-bound algorithm for 3D bounding-box cropping, by maximizing the mutual information of features and actions. Hoai *et al* [33] performed joint segmentation and classification. Gaidon *et al.* [34] learned actoms for modeling and localizing actions. Accurate temporal localization, however, is not a focus of this paper. Instead, our effort is to improve the classifier performance based on the distribution of classification scores.

The inadequacy of using the maximum score in MIL has been observed before. Cheung & Kwok [35] suggested combining bag and instance feature vectors to improve the classification performance. Hu *et al.* [17] considered both the maximum and the average scores, and reported better classification performance for the average score in many experiments. In this paper, we propose

to consider the classifier scores of all instances instead. It will be seen that the distribution-based decision is more effective than both the extreme (max) and summary (mean) statistics.

A part of this work is related to Ordered Weighted Averaging (OWA) [36, 37], which is an aggregation operator for multiple criteria. Our work is also based on ranking and aggregation, but it considers a single criterion of multiple instances instead. There are also some multiple instance learning formulations that use OWA or a similar fusion operator [38, 39].

High-level representation from the outputs of multiple classifiers have been shown to help object detection and image classification. Aytar *et al.* [40] combined the outputs of multiple concept detectors to improve retrieval results. Rabinovich *et al.* [41] incorporated semantic object context to improve a categorization model. Torresani *et al.* [42], Li *et al.* [43], and Sadanand & Corso [44] proposed classemes, object bank, and action bank, respectively, which are generic classifiers for generating high-level feature vectors. Bourdev *et al.* [45] obtained attribute classifiers from poselet outputs. Song *et al.* [46] proposed Context-SVM that provided mutual benefits for object detection and image classification. Felzenszwalb *et al.* [13] re-scored a detector based on the scores and locations of multiple detectors. Unlike the aforementioned approaches that consider a fixed order of classifiers, our methods ground the decisions on the distribution and ordering of action scores. As will be seen, this is crucial for action recognition.

Least-Squares SVM. We propose to use Least-Squares Support Vector Machines (LSSVM) [47]. LSSVM, also known as kernel Ridge regression [48], has been shown to perform equally well as SVM in many classification benchmarks [49]. LSSVM has a closed-form solution, which is a computational advantage over SVM. Furthermore, once the solution of LSSVM has been computed, the solution for a reduced training set obtaining by removing any training data point can found efficiently. This enables reusing training data for further calibration (e.g., used in [50, 51]). This section reviews LSSVM and the leave-one-sample-out formula.

Given a set of n data points $\{\mathbf{x}_i | \mathbf{x}_i \in \Re^d\}_{i=1}^n$ and associated labels $\{y_i | y_i \in \{1, -1\}\}_{i=1}^n$, LSSVM optimizes the following:

$$\underset{\mathbf{w},b}{\text{minimize}} \;\; \lambda ||\mathbf{w}||^2 + \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i + b - y_i)^2. \tag{1}$$

For high dimensional data ($d \gg n$), it is more efficient to obtain the solution for $(\mathbf{w}, b)$ via the representer theorem, which states that $\mathbf{w}$ can be expressed as a linear combination of training data, i.e., $\mathbf{w} = \sum_{i=1}^n \alpha_i \mathbf{x}_i$. Let $\mathbf{K}$ be the kernel matrix, $k_{ij} = \mathbf{x}_i^T \mathbf{x}_j$. The optimal coefficients $\{\alpha_i\}$ and the bias term b can be found using closed-form formula: $[\boldsymbol{\alpha}^T, b]^T = \mathbf{M}\mathbf{y}$. Where $\mathbf{M}$ and other auxiliary variables are defined as:

$$\mathbf{R} = \begin{bmatrix} \lambda \mathbf{K} & \mathbf{0}_n \\ \mathbf{0}_n^T & 0 \end{bmatrix}, \mathbf{Z} = \begin{bmatrix} \mathbf{K} \\ \mathbf{1}_n^T \end{bmatrix}, \mathbf{C} = \mathbf{R} + \mathbf{Z}\mathbf{Z}^T, \mathbf{M} = \mathbf{C}^{-1}\mathbf{Z}, \mathbf{H} = \mathbf{Z}^T\mathbf{M}. \tag{2}$$

If $\mathbf{x}_i$ is removed from the training data, the optimal coefficients can be computed:

$$\begin{bmatrix} \boldsymbol{\alpha}_{(i)} \\ b_{(i)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} + \left(\frac{[\boldsymbol{\alpha}^T \ b]\mathbf{z}_i - y_i}{1 - h_{ii}} \right) \mathbf{m}_i. \tag{3}$$

Here, $\mathbf{z}_i$ is the i^{th} column vector of $\mathbf{Z}$ and h_{ii} is the i^{th} element in the diagonal of $\mathbf{H}$. Note that $\mathbf{R}, \mathbf{Z}, \mathbf{C}, \mathbf{M}$, and $\mathbf{H}$ are independent of the label vector $\mathbf{y}$. Thus, training LSSVMs for multiple classes is efficient as these matrices need to be computed once. A more gentle derivation of the above formula is given in [52].

3 Subsequence-Score Distribution (SSD)

To handle the ambiguity of the temporal extent of an action in a video clip, we sample subsequences of the video clip at multiple locations and scales. We compute the improved action score for the video clip based on the score distribution of the subsequences.

3.1 Formulation

Assume we have learned a base classifier for a particular action. Given a video clip $\mathbf{x}$, we sample l subsequences $\mathbf{x}^1 \cdots \mathbf{x}^l$ of $\mathbf{x}$ with replacement, compute their action scores, and sort the scores in descending order to obtain a vector $\mathbf{d}$, i.e., $\mathbf{d} = [sort(f(\mathbf{x}^1), \cdots, f(\mathbf{x}^l))]^T$. Here, *sort* is the function that reorders the inputs in descending order. With a sufficiently large l, $\mathbf{d}$ represents the score distribution of subsequences from $\mathbf{x}$. In practice, for computational efficiency, it is unnecessary to sample from the set of all video subsequences because of strong temporal correlation between nearby frames. We therefore restrict our consideration to a subset of video subsequences. In particular, we divide a video clip into several intervals, and only consider the subsequences that can be obtained by concatenating a set of adjacent intervals. For example, if a video is divided into 10 intervals, then there are $l = 55$ possible subsequences. Details of how a video is divided into intervals and the number of subsequences are given in Sec. 5.1.

Given n video clips $\{\mathbf{x}_i\}_{i=1}^n$, each represented by the distribution feature vector $\mathbf{d}_i$, we learn an SSD classifier by optimizing the following objective:

$$\underset{\mathbf{s},b}{\text{minimize}} \ \sum_{i=1}^{n} \max(1 - y_i(\mathbf{s}^T\mathbf{d}_i + b), 0) \tag{4}$$

$$\text{s.t.} \ \sum_{j=1}^{l} s_j = 1, \text{and } s_1 \geq s_2 \geq \cdots \geq s_l \geq 0. \tag{5}$$

The above optimization problem seeks a weight vector $\mathbf{s}$ and the bias term b for separating between score distribution vectors of positive and negative data. The objective in Eq. (4) is the sum of Hinge losses, as used in the standard SVM objective. Constraint (5) requires the weights to be non-negative, monotonic,

and have unit sum. Recall that $\mathbf{d}_i$ are classification scores sorted in descending order. The weights should be non-negative and monotonic to emphasize the relative importance of higher classification scores. The weights should have unit sum because we are learning the weights for score distributions. The feasible set of $\mathbf{s}$ subsumes two special cases:

1. $s_1 = 1, s_2 = \cdots = s_l = 0$. This corresponds to using the maximum score.
2. $s_1 = s_2 \cdots = s_l = \frac{1}{l}$. This corresponds to using the average score.

The optimization problem (4) is linear. It can be efficiently solved by any linear programming tool, e.g., Cplex[1]. Once $\mathbf{s}, b$ have been learned, we compute the improved recognition score for a test video $\mathbf{x}$ as follows. First, sample l subsequences $\mathbf{x}^1, \cdots, \mathbf{x}^l$ of $\mathbf{x}$. The improved classifier is defined as: $f^*(\mathbf{x}) = \mathbf{s}^T[sort(f(\mathbf{x}^1), \cdots, f(\mathbf{x}^l))]^T + b$. This technique is illustrated in Fig. 1(a).

The same set of training data can be used to learn the base classifier f and the improved classifier f^*. To avoid overfitting, we compute $\mathbf{d}_i$ using the leave-one-out versions of f. Let f' be the base classifier by removing $\mathbf{x}_i$ from the training data, $\mathbf{d}_i = [sort(f'(\mathbf{x}_i^1), \cdots, f'(\mathbf{x}_i^l))]^T$. If LSSVM is used, the leave-one-out classifiers can be computed efficiently with closed form formula, as shown in Sec. 2. Nevertheless, the technique proposed here can be applied to any type of classifiers.

3.2 Controlled Experiments on Synthetic Data

When would it be beneficial to use SSD and how much improvement should we expect? To answer this question, we perform a set of controlled experiments on synthetic data. We vary two important factors that affect the difficulty of action recognition: (i) the separation between positive and negative descriptors; and (ii) the proportion of the target action in each video clip.

We generate 200 positive and 2000 negative video clips, half of them are used for training and half for testing. Each positive video contains the action of interest, however, only a portion of the video depicts the action. The percentage of the video that corresponds to the action is a controlled parameter of the experiment, which will be referred as the *action percentage*. For simplicity, we assume the action part is contiguous, and its location is randomly distributed. Each video is represented by a sequence of 1000 synthetic descriptors; these descriptors are analogous to dense trajectory descriptors for real videos [4] (explained in Sec. 5.1). Negative descriptors (for negative videos or outside the non-action parts of positive videos) are generated from a 10-dimensional Normal distribution. Positive descriptors (for the action parts of positive videos) are also generated from a 10-dimensional Normal distribution, except for the first dimension which is shifted by a value μ. μ is a controlled parameter of this experiment, and it inversely correlates with the difficulty of separating between positive and negative descriptors. We use Fisher vector [53] with two Gaussians to encode descriptors. Each video is divided into 10 intervals, and 55 possible

[1] http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

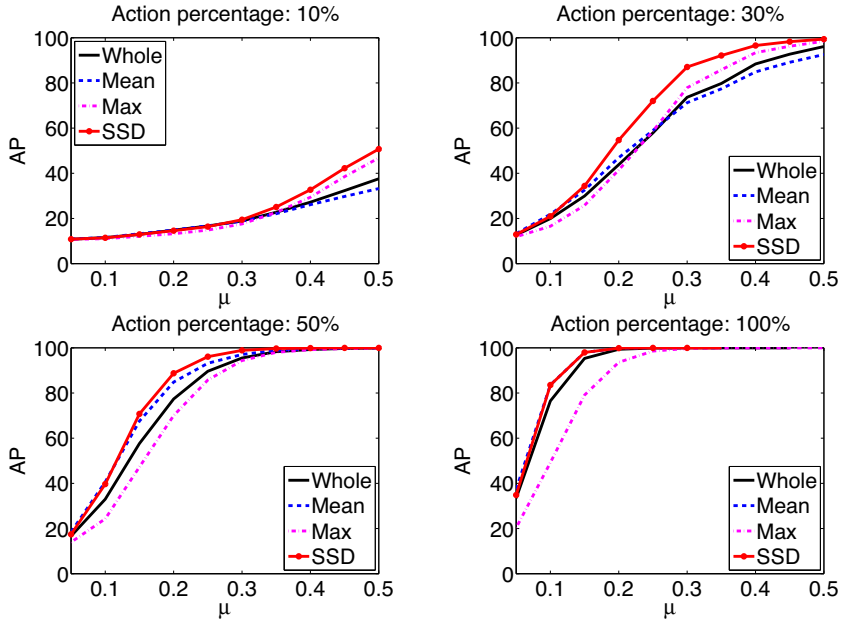


Fig. 2. Average precision as a function of: i) the action percentage, which is the percentage in each positive video that depicts the action; and ii) the separation between positive and negative descriptors. SSD (red solid curves) outperforms competing approaches, but the relative advantage depends on these two factors

video subsequences are considered, each is represented by a Fisher vector. As will be seen in Sec. 5, except for using synthetic data, the experiment setup here is the same as the setup for real data.

For each setting of the controlled parameters, we repeat the experiment 50 times and average the results. We compare SSD with *Whole*, *Max*, and *Mean*. Whole is the method that is trained on the whole video clips and tested on the whole video clips. Whole is the base classifier for Max, Mean, and SSD. Max is the method that uses the maximum subsequence score, while Mean uses the average score. SSD is the method that is based on the entire score distribution.

Fig. 2 plots the Average Precision (AP) for action recognition as a function of: i) the action percentage; and ii) the offset between positive and negative descriptors. The four subplots correspond to four different action percentages, which are 10%, 30%, 50%, 100%. The black solid lines are the performance curves of Whole, which correlate with the action percentage and μ. In general, using the maximum score of video subsequences (Max) is better than using the average score when the the separation between positive and negative descriptors is easy (high μ) and when the action percentage is low. Conversely, it is better to use the mean when the base classifier is less accurate and the action percentage is higher. The proposed approach outperforms both the max and the mean, but the relative advantage depends on the action percentage and the separability

between positive and negative descriptors. For more experiments and further analysis, please see the supplementary material.

4 Relative Class Scores (RCS)

To capture the correlation and exclusion among action classes, we learn a classifier for the ranked list of action scores. This section describes this technique.

Consider a particular action and suppose we have learned a base classifier f. Given a video clip $\mathbf{x}$, $f(\mathbf{x})$ is the score for the given action. Suppose we also have m classifiers $f_1 \cdots f_m$ for m other action classes. For each training video clip $\mathbf{x}_i$, we construct a Relative Class Score (RCS) vector (Fig. 1(b)) as follows:

$$\mathbf{a}_i = [f(\mathbf{x}_i), sort(f_1(\mathbf{x}_i), \cdots, f_m(\mathbf{x}_i))]^T. \tag{6}$$

We train a linear SVM to separate between RCS vectors of positive data from RCS vectors of negative data, obtaining a weight vector $\mathbf{v}$ and bias term b. The improved classifier for the target action is defined as:

$$f^*(\mathbf{x}) = \mathbf{v}^T[f(\mathbf{x}), sort(f_1(\mathbf{x}), \cdots, f_m(\mathbf{x}))]^T + b. \tag{7}$$

The same set of training data can be used to learn the base classifiers $f, f_1, \cdots, f_m$ and the improved classifier f^*. To avoid overfitting, we compute $\mathbf{a}_i$ using the leave-one-out classifiers $f', f'_1, \cdots, f'_m$, obtained by removing $\mathbf{x}_i$ from the training data. This technique is applicable to any type of classifiers. However, if LSSVM is used, the leave-one-out classifiers can be computed efficiently using closed form formula, as shown in Sec. 2.

An alternative to the above is to calculate the RCS vector keeping the order of classes: $\mathbf{a}_i = [f(\mathbf{x}_i), f_1(\mathbf{x}_i), \cdots, f_m(\mathbf{x}_i)]^T$ (without sorting), as is often done. However, this performs poorly for action recognition, as will be seen in Sec. 5.2.

5 Experiments on Real Data

5.1 Experimental setup

Datasets. This section describes Hollywood2, HMDB51, TVHI, which are three datasets used in our experiments. These challenging datasets are widely used for benchmarking human action recognition methods.

The Hollywood2 dataset [2] has 12 action classes and contains 1707 video clips collected from 69 different Hollywood movies. The videos are split into a training set of 823 videos and a testing set of 884 videos. The training and testing videos come from different movies.

The TVHI dataset [1] consists of 300 video clips compiled from 23 different TV shows. There are four classes, corresponding to four types of human interaction: Handshake, Highfive, Hug, and Kiss. Each of these interactions has 50 videos. There are 100 negative examples, which do not contain any of the above interactions. We keep the training/testing split of the dataset [1].

The HMDB51 dataset [3] contains 6766 video sequences for 51 action categories. The videos are collected from various sources including digitized movies and YouTube videos. We follow the suggested protocol for three train-test splits [3]. For every class and split, there are 70 videos for training and 30 videos for testing. We report the mean performance over the three splits as the overall performance. Note that we use the original videos and not the stabilized ones.

We measure performance using Average Precision (AP), which is an accepted standard for action recognition [2, 1, 54–58]. We only compute Accuracy (ACC) for HMDB51 to compare with published results [4, 59, 3]. In this paper, the default reporting performance is in AP, if not stated otherwise.

Trajectory features. The feature representation is based on improved Dense-Trajectory Descriptors (DTDs) [4]. DTD extracts dense trajectories and encodes gradient and motion cues along trajectories. Each trajectory leads to four feature vectors: Trajectory, HOG, HOF, and MBH, which have dimensions of 30, 96, 108, and 192 respectively. We refer the reader to [4] for more details.

The procedure for extracting DTDs is the same as [4] with two subtle modifications: (i) videos are normalized to have the height of 360 pixels, and (ii) frames are extracted at 25 fps. These modifications are added to standardize the feature extraction procedure across videos and datasets. They do not significantly alter the performance of the action recognition system, as verified in our experiments.

Fisher vector encoding. To encode features, we use Fisher vector [53]. Fisher vector encodes both first and second order statistics between the feature descriptors and a Gaussian Mixture Model (GMM). In [4], Fisher vector shows an improved performance over bag of features for action classification. Following [53, 4], we first reduce the dimension of DTDs by a factor of two using Principal Component Analysis (PCA). We set the number of Gaussians to $k = 256$ and randomly sample a subset of 1,000,000 features from the training sets of TVHI and Hollywood2 to learn the GMM. There is one GMM for each feature type. A video sequence is represented by a $2dk$ dimensional Fisher vector for each descriptor type, where d is the descriptor dimension after performing PCA. As in [53, 4], we apply power ($\alpha = 0.5$) and L_2 normalization to the Fisher vectors. We combine all descriptor types by concatenating their normalized Fisher vectors, leading to a single feature vector of $109,056$ dimensions.

Video subsequences. We divide each video clip into roughly 10 intervals and only consider video subsequences that can be composed by concatenating adjacent intervals. The division into intervals is based on shot boundaries and shot lengths. We first run a shot boundary detection algorithm (explained below) to obtain all shot boundaries. We divide the video clip into intervals such that intervals either include or do not include shot boundaries. The video clip is divided into a sequence of intervals that alternate between those straddling shot boundaries (which are restricted to about 0.6s in duration) and those in between. This division is unique and can be constructed deterministically. If the

number of intervals is more than or equal 10, we terminate the interval division procedure. If the number of intervals is less than 10, we partition the longest intervals (those within shots) to create a total of 10 intervals. Thus, a video clip is normally divided into 10 intervals except when the video clip has many shot boundaries. The number of intervals can be smaller than 10 if the clip length is less than 25 frames (1s). If a video clip is divided into k intervals, the total number of different subsequences is $k(k+1)/2$. Regardless of k, we sample a fixed l number of subsequences with replacement. We found $l = 1100$ sufficiently large for the sample set to represent the true distribution of subsequences. Note that if k was constant, it would be sufficient to use $l = k(k+1)/2$, i.e., to use all subsequences. But since k varies around 10, we use $l = 1100$ ($\gg k(k+1)/2$) for the sample set for the sample set to be a stable fixed length vector (i.e., not too sensitive to randomness in the sampling process).

For shot boundary detection, we develop an algorithm based on normalized color histograms, HOG, and SIFT. Based on normalized color histograms and HOG, the algorithm produces a set of candidate shot boundaries by thresholding the difference between pairs of consecutive frames. Subsequently, SIFT matching is used to remove false candidates. Evaluated on the TVHI dataset, this shot boundary detection algorithm has 0 false positive and 1 false negative.

5.2 Experimental results

SVM, LSSVM, and the validity of feature extraction. To validate our video processing and feature extraction procedure, we compare the baseline performance with published results [4]. We evaluate LSSVM and SVM on the Fisher vectors computed for the entire video clips of the Hollywood2 dataset. These classifiers achieve APs of 64.63% and 64.71%, respectively. These numbers are comparable to the AP of 64.30% published by [4]. On HMDB51, LSSVM and SVM achieve ACCs of 56.3 and 57.0, which are comparable to ACC of 57.2 published by [4]. On the TVHI dataset, LSSVM and SVM yield APs of 61.94% and 61.92%, respectively. From this experiment, we conclude that: i) the features are properly extracted and consistent with [4], and ii) LSSVM and SVM perform similarly. In subsequent experiments, we choose LSSVM because of its computational advantage.

Data augmentation. Based on the observation that a video and its left-right mirrored video depict the same concept, we propose data augmentation to improve the performance. We double the amount of training data by adding videos obtained by left-right flipping. In testing, we average the classification scores of a test video and its mirrored version. This leads to a few percent improvement in AP. Specifically, on Hollywood2, TVHI, and HMDB51 datasets, the APs increase from 64.63%, 61.94%, 57.53% to 66.68%, 66.6%, 59.70%, respectively. These improved results will serve as the new baselines in subsequent experiments.

Discussion of results. Tab. 1 displays the APs of various methods on three datasets. Overall, the proposed methods (SSD, RCS, SSD+RCS) significantly outperform the improved baseline (Whole+). On the HMDB51 dataset, RCS and

Table 1. Average Precisions on three datasets. Whole: the classifier is trained on the whole video clips and tested on the whole video clips. Whole+: same as Whole, but with data augmentation. Other methods in the table use Whole+ as the base classifier. Mean, Max: using either the mean or maximum score. SSD: learned classifier for the score distribution of video subsequences. RCS: improved classifier based on scores of multiple action classes. SSD+RCS: combined method

			With proposed data augmentation					
		Baseline	New baseline			Proposed		
	Class/Mean	Whole	Whole+	Mean	Max	SSD	RCS	SSD+RCS
	Hollywood2-Mean	64.6	66.7	68.1	64.8	69.1	69.9	**72.7**
	TVHI-Mean	61.9	66.6	69.1	65.0	69.5	67.0	**70.6**
	HMDB51-Mean	57.5	59.7	58.4	55.9	58.3	**63.1**	62.2
Hollywood2	AnswerPhone	29.0	33.9	34.0	23.8	32.7	44.6	**45.4**
	DriveCar	94.7	94.5	94.9	92.4	94.7	95.1	**96.0**
	Eat	65.2	65.8	69.3	66.4	**69.6**	63.9	68.6
	FightPerson	84.8	86.1	85.7	86.0	87.4	88.4	**89.1**
	GetOutCar	62.4	67.6	70.7	63.1	70.9	72.4	**74.3**
	HandShake	44.7	48.0	50.3	57.1	57.1	56.9	**65.0**
	HugPerson	51.5	53.4	51.0	48.7	52.8	57.8	**58.6**
	Kiss	65.5	66.2	67.8	61.4	66.3	67.0	**68.7**
	Run	86.1	87.4	87.7	86.2	88.6	88.7	**90.4**
	SitDown	78.3	79.0	81.4	75.6	80.8	82.9	**84.6**
	SitUp	35.4	38.6	41.1	38.0	44.2	39.2	**46.4**
	StandUp	77.8	79.8	82.8	79.2	84.0	81.6	**85.4**

SSD+RCS outperform Whole+, but SSD does not. On this dataset, Mean and Max also yield lower performance than Whole+. This is due to the idiosyncrasy of HMDB51 as it consists of very short video clips (2-3s), which are well clipped to the target actions.

For a more detailed analysis, Tab. 1 also reports the APs for individual action classes of Hollywood2. This reveals some notable facts. Max generally performs poorly, leading to inferior results than Whole+ and Mean. This is perhaps because human action recognition is a hard problem, so the base classifier is far from perfect and the maximum subsequence score is unreliable. However, Max can sometimes outperforms both Whole+ and Mean, e.g., for HandShake. Among Whole+, Mean, Max, and SSD, the proposed method SSD performs the best or close to the best in all cases. The combination of SSD and RCS is consistently outstanding.

The results provided in Tab. 1 are obtained with the base classifiers trained on the whole video clips. We consider an alternative way of training the base classifiers as follows. We represent a training video clip by the average of Fisher vectors computed for its subsequences. We train the base classifiers for this feature representation. These base classifiers improve the performance of all methods on Hollywood2 and TVHI datasets, but yield lower results on HMDB51.

Table 2. Comparison with previously published results. The proposed methods (last row) outperform state-of-the-art approaches on Hollywood2 and TVHI datasets. The second last row is our reproduction results for Wang *et al* [4]. These methods and ours use exactly the same features

Hollywood2	AP	TVHI	AP	HMDB51	ACC	AP
Vig *et al.* [54]	59.4	Marin *et al.* [55]	39.2	Kliper *et al.* [59]	29.2	–
Jiang *et al.* [56]	59.5	Patron *et al.* [1]	42.4	Jiang *et al.* [56]	40.7	–
Mathe *et al.* [57]	61.0	Gaidon *et al.* [58]	55.6	Jain *et al.* [61]	52.1	–
Jain *et al.* [61]	62.5	Yu *et al.* [62]	55.9	Wang *et al.* [4]	57.2	–
Wang *et al.* [4]	64.3	Hoai *et al.* [63]	56.3	Peng *et al.* [60]	**66.8**	–
DTD-SVM [4]	64.7	DTD-SVM [4]	61.9	DTD-SVM [4]	57.0	57.8
SSD+RCS	**73.6**	SSD+RCS	**71.1**	TempoPyra+RCS	60.8	**65.9**

On the Hollywood2 dataset, the APs for Whole+ (tested on the whole video clips), Mean, Max, SSD, RCS, SSD+RCS are 67.6, 68.9, 66.0, 70.0, 72.4, **73.6**, respectively. On TVHI dataset, the APs for Whole+, Mean, Max, SSD, RCS, SSD+RCS are 66.5, 68.6, 65.8, 69.9, 70.0, **71.1**, respectively.

Tab. 2 compares the performances from SSD+RCS to previously published results. It is evident that the performance exceeds the previous state of the art results by a wide margin. Very recently, after the completion of this work, the state-of-the-art accuracy on HMDB51 was reestablished to be 66.8 [60]. However, [60] used different feature representation and encoding, which may also gain benefit from SSD and RCS.

A complementary technique to the proposed methods is to use temporal pyramid. Following this direction, we develop a method with a 2-layer pyramid: (i) divide a video clip into two halves, (ii) compute the Fisher vectors for both halves and the whole video clip, and (iii) represent the video clip by concatenating the three Fisher vectors. We will refer to this method as TempoPyra. TempoPyra (with data augmentation) outperforms Whole+ on Hollywood2 and HMDB51, but not on TVHI. Specifically, the APs of TempoPyra on Hollywood2, TVHI, and HMDB51, are 68.7, 66.0, and 62.4, respectively. These results are lower than the performance of the proposed methods. Furthermore, it is important to emphasize that TempoPyra and the proposed methods are complementary, as video clips and video subsequences can be represented using a temporal pyramid. For example, the combination of TempoPyra and RCS yields an AP of **65.9** on HMDB51, which outperforms the state-of-the-art results, as shown in Tab. 2.

Analysis of SSD classifiers. Fig. 3 shows the weight distribution of SSD classifiers for several actions of the Hollywood2 dataset. Recall that the scores are sorted in descending order and the weights decrease monotonically. For AnswerPhone, our algorithm suggests to aggregate the scores from the top 80 percentile. Meanwhile, for HandShake, the algorithm suggests to consider the top 35 percentile only. These results are intuitively consistent with the analysis in Sec. 3.2. For a handshake in Hollywood movies, the temporal extent of the actual hand-

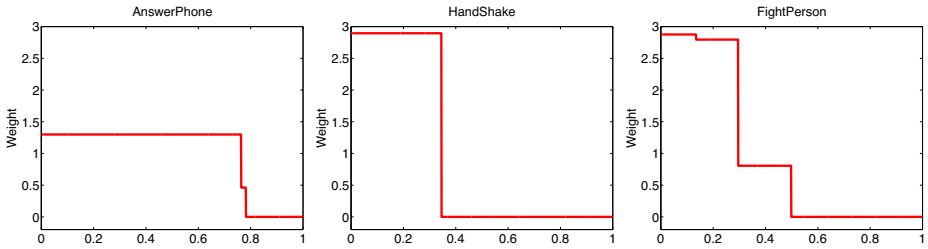


Fig. 3. Weights for SSD classifiers for Hollywood2 dataset. Each subplot depicts the weights for the score distribution. Weights decrease monotonically, and the area under the curve is always 1. AnswerPhone aggregates scores from 80% of video subsequences, while HandShake only uses the top 35 percentile

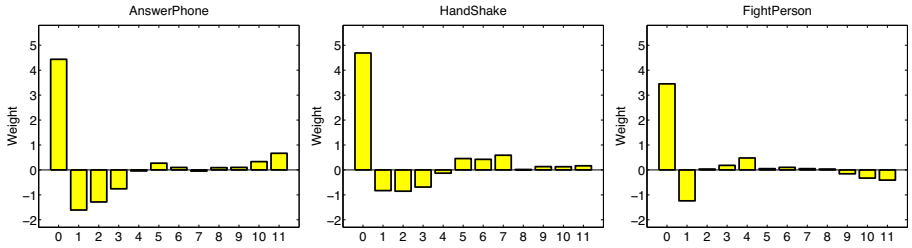


Fig. 4. RCS weights for actions of Hollywood2 dataset. In each subplot, the first bar is the weight for the base classifier of the target action. The remaining bars show the weights for the sorted scores of other classes. All RCS classifiers penalize the top-rank score of other classifiers (negative weights)

shake is usually brief, and it could possibly interrupted when the camera switches to showing other people watching the handshake. In contrast, a video clip for AnswerPhone often shows one or two actors talking on the phone for a period time, as long as the length of the phone conversation. Empirical evidence in Tab. 1 also confirms this intuition. Mean outperforms Max for AnswerPhone, and Max outperforms Mean for HandShake. SSD learns from data, and it performs close to the best of Max and Mean in both cases. For some other actions such as FightPerson and SitUp, SSD outperforms both Max and Mean.

Analysis of RCS classifiers. Fig. 4 shows the RCS weights for the combined SSD+RCS classifiers on the Hollywood2 dataset. For all actions, the RCS classifier emphasizes the score of the target class by using a high positive weight, and it penalizes the highest scores of the other classes (negative weights). The spread of the penalty weights depends on the action, which is learned from the data. The use of RCS drastically improves AP (e.g., from 32.7 to 45.4 for AnswerPhone, and from 57.1 to 65.0 for HandShake).

It is crucial to sort the scores of other action classes when constructing the feature vectors to train the RCS classifier. If not, the RCS classifier brings no benefit. Specifically, if the scores are kept based on the order of the classes, the APs for SSD+RCS on Hollywood2 and TVHI are 69.3 and 69.4. These APs are

similar to the APs of not using RCS; SSD alone achieves APs of 69.1 and 69.5 on these two datasets. We also experimented with applying the sigmoid function on the raw SVM scores before learning the RCS classifier with unsorted scores, but this also performs poorly.

Parameter setting. The proposed methods require tuning few parameters. LSSVM has only a single parameter (λ in Eq. (1)), and its classification performance is not too sensitive to λ. In all of our experiments, we set $\lambda = 10^{-3} \times n$, where n is the number of training examples. The formulation for learning SSD classifiers has no parameter.

6 Conclusions, Discussions, and Future Work

We have proposed several techniques for human action recognition, improving the state-of-the-art performance on three challenging benchmark datasets. First, we used data augmentation to learn a flipping invariant classifier. Second, we replaced SVMs by Least-Squares SVMs, which performed equally well and are more computationally efficient. Moreover, Least-Squares SVM enabled reusing training data for further tuning and calibration. Third, we proposed distribution-based classifiers to address the temporal ambiguity of actions, proving its advantage over using maximum and average scores. Fourth, we showed action recognition can benefit from exploiting the correlation and exclusion between action classes, by learning a classifier for the relative action scores.

We have applied the aforementioned techniques to improve the performance of base classifiers. In this paper, we simply obtained the base classifiers by training them on the whole video clips. We have not considered updating the base classifiers after improvement. A direction for future work is to investigate an iterative scheme for updating and improving the base classifiers, as in MI-SVM [12].

Empirical evidence in this paper suggests that using the maximum score leads to poor performance in many cases. On the one hand, this is consistent with empirical observation reported earlier [35, 17]. On the other hand, it seems to conflict with the success of several weakly-supervised learning systems such as the Deformable Part Model (DPM) [13] for object detection. There are several possible reasons for the good performance of DPM. First, it is trained on supervised data, where the object location is given. Second, the location of a part of a DPM is heavily regularized by a quadratic function enforcing the part to remain close to an anchor point. This prevents the model selecting the part location using the maximum score alone. If this is indeed a reason for the success of the DPM, we could possibly adapt it for action recognition by putting regularization on the location of the action. This is another direction for future investigation.

Acknowledgements.
This work was supported by the EPSRC grant EP/I012001/1 and a Royal Society Wolfson Research Merit Award.

References

1. Patron-Perez, A., Marszalek, M., Reid, I., Zisserman, A.: Structured learning of human interactions in TV shows. IEEE Transactions on Pattern Analysis and Machine Intelligence **34** (2012) 2441–2453
2. Marszalek, M., Laptev, I., Schmid, C.: Actions in context. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2009)
3. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: HMDB: A large video database for human motion recognition. In: Proceedings of the International Conference on Computer Vision. (2011)
4. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: Proceedings of the International Conference on Computer Vision. (2013)
5. Satkin, S., Hebert, M.: Modeling the temporal extent of actions. In: Proceedings of the European Conference on Computer Vision. (2010)
6. Duchenne, O., Laptev, I., Sivic, J., Bach, F.R., Ponce, J.: Automatic annotation of human actions in video. In: Proceedings of the International Conference on Computer Vision. (2009)
7. Buehler, P., Everingham, M., Zisserman, A.: Learning sign language by watching TV (using weakly aligned subtitles). In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2009)
8. Niebles, J.C., Chen, C.W., Fei-Fei, L.: Modeling temporal structure of decomposable motion segments for activity classification. In: Proceedings of the European Conference on Computer Vision. (2010)
9. Lan, T., Wang, Y., Mori, G.: Discriminative figure-centric models for joint action localization and recognition. In: Proceedings of the International Conference on Computer Vision. (2011)
10. Shapovalova, N., Vahdat, A., Cannons, K., Lan, T., Mori, G.: Similarity constrained latent support vector machine: An application to weakly supervised action classification. In: Proceedings of the European Conference on Computer Vision. (2012)
11. Prest, A., Schmid, C., Ferrari, V.: Weakly supervised learning of interactions between humans and objects. IEEE Transactions on Pattern Analysis and Machine Intelligence **34** (2012) 601–614
12. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: Advances in Neural Information Processing Systems. (2003)
13. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. IEEE Transactions on Pattern Analysis and Machine Intelligence **32** (2010) 1627–1645
14. Dietterich, T., Lathrop, R., Lozano-Pérez, T.: Solving the multiple-instance problem with axis-parallel rectangles. Artificial Intelligence **89** (1997) 31–71
15. Maron, O., Lozano-Pérez, T.: A framework for multiple-instance learning. In: Advances in Neural Information Processing Systems. (1998)
16. Zhang, Q., Goldman, S.A.: EM-DD: An improved multiple-instance learning technique. In: Advances in Neural Information Processing Systems. (2002)
17. Hu, Y., Li, M., Yu, N.: Multiple-instance ranking: Learning to rank images for image retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2008)
18. Ray, S., Craven, M.: Supervised versus multiple instance learning: an empirical comparison. In: Proceedings of the International Conference on Machine Learning. (2005)

19. Wohlhart, P., Köstinger, M., Roth, P.M., Bischof, H.: Multiple instance boosting for face recognition in videos. In: Proceedings of the International Conference on Pattern Recognition. (2011)
20. Gartner, T., Flach, P.A., Kowalczyk, A., Smola, A.J.: Multi-instance kernels. In: Proceedings of the International Conference on Machine Learning. (2002)
21. Chen, Y., Bi, J., Wang, J.Z.: MILES: Multiple-instance learning via embedded instance selection. IEEE Transactions on Pattern Analysis and Machine Intelligence **28** (2006) 1931–1947
22. Kwok, J.T., Cheung, P.M.: Marginalized multi-instance kernels. In: International Joint Conference on Artificial Intelligence. (2007)
23. Ping, W., Xu, Y., Wang, J., Hua, X.S.: FAMER: Making multi-instance learning better and faster. In: International Conference on Data Mining. (2011)
24. Zhou, Z.H., Sun, Y.Y., Li, Y.F.: Multi-instance learning by treating instances as non-i.i.d. samples. In: Proceedings of the International Conference on Machine Learning. (2009)
25. Ping, W., Xu, Y., Ren, K., Chi, C.H., Shen, F.: Non-i.i.d. multi-instance dimensionality reduction by learning a maximum bag margin subspace. In: AAAI Conference on Artificial Intelligence. (2010)
26. Li, W., Duan, L., Xu, D., Tsang, I.W.H.: Text-based image retrieval using progressive multi-instance learning. In: Proceedings of the International Conference on Computer Vision. (2011)
27. Hajimirsadeghi, H., jinling Li, Mori, G., Sayed, T., Zaki, M.: Multiple instance learning by discriminative training of markov networks. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence. (2013)
28. Poggio, T., Vetter, T.: Recognition and structure from one 2D model view: Observations on prototypes, object classes and symmetries. Technical Report AIM-1347, MIT (1992)
29. Vedaldi, A., Blaschko, M., Zisserman, A.: Learning equivariant structured output svm regressors. In: Proceedings of the International Conference on Computer Vision. (2011)
30. Nowozin, S., Bakir, G., Tsuda, K.: Discriminative subsequence mining for action classification. In: Proceedings of the International Conference on Computer Vision. (2007)
31. Nguyen, M.H., Torresani, L., De la Torre, F., Rother, C.: Weakly supervised discriminative localization and classification: a joint learning process. In: Proceedings of the International Conference on Computer Vision. (2009)
32. Yuan, J., Liu, Z., Yu, Y.: Discriminative subvolume search for efficient action detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2009)
33. Hoai, M., Lan, Z.Z., De la Torre, F.: Joint segmentation and classification of human actions in video. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2011)
34. Gaidon, A., Harchaoui, Z., Schmid, C.: Actom sequence models for efficient action detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2011)
35. Cheung, P.M., Kwok, J.T.: A regularization framework for multiple-instance learning. In: Proceedings of the International Conference on Machine Learning. (2006)
36. Yager, R.R.: On ordered weighted averaging aggregation operators in multicriteria decisionmaking. IEEE Transactions on Systems, Man and Cybernetics **18** (1988) 183–190

37. Yager, R.R., Filev, D.P.: Induced ordered weighted averaging operators. IEEE Transactions on Systems, Man and Cybernetics **29** (1999) 141–150
38. Hajimirsadeghi, H., Mori, G.: Multiple instance real boosting with aggregation functions. In: Proceedings of the International Conference on Pattern Recognition. (2012)
39. Li, F., Sminchisescu, C.: Convex multiple-instance learning by estimating likelihood ratio. In: Advances in Neural Information Processing Systems. (2010)
40. Aytar, Y., Orhan, O.B., Shah, M.: Improving semantic concept detection and retrieval using contextual estimates. In: ICME. (2007)
41. Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., Belongie, S.: Objects in context. In: Proceedings of the International Conference on Computer Vision. (2007)
42. Torresani, L., Szummer, M., Fitzgibbon, A.: Efficient object category recognition using classemes. In: Proceedings of the European Conference on Computer Vision. (2010)
43. Li, L.J., Su, H., Xing, E.P., Fei-Fei, L.: Object bank: A high-level image representation for scene classification and semantic feature sparsification. In: Advances in Neural Information Processing Systems. (2010)
44. Sadanand, S., Corso, J.J.: Action bank: A high-level representation of activity in video. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2012)
45. Bourdev, L., Maji, S., Malik, J.: Describing people: A poselet-based approach to attribute classification. In: Proceedings of the International Conference on Computer Vision. (2011) 1543–1550
46. Song, Z., Chen, Q., Huang, Z., Hua, Y., Yan, S.: Contextualizing object detection and classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2010)
47. Suykens, J.A.K., Vandewalle, J.: Least squares support vector machine classifiers. Neural Processing Letters **9** (1999) 293–300
48. Saunders, C., Gammerman, A., Vovk, V.: Ridge regression learning algorithm in dual variables. In: Proceedings of the International Conference on Machine Learning. (1998)
49. Suykens, J.A.K., Gestel, T.V., Brabanter, J.D., DeMoor, B., Vandewalle, J.: Least Squares Support Vector Machines. World Scientific (2002)
50. Tommasi, T., Caputo, B.: The more you know, the less you learn: From knowledge transfer to one-shot learning of object categories. In: Proceedings of the British Machine Vision Conference. (2009)
51. Hoai, M.: Regularized max pooling for image categorization. In: Proceedings of the British Machine Vision Conference. (2014)
52. Cawley, G.C., Talbot, N.L.: Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. Neural Networks **17** (2004) 1467–1475
53. Perronnin, F., Sánchez, J., Mensink, T.: Improving the fisher kernel for large-scale image classification. In: Proceedings of the European Conference on Computer Vision. (2010)
54. Vig, E., Dorr, M., Cox, D.: Space-variant descriptor sampling for action recognition based on saliency and eye movements. In: Proceedings of the European Conference on Computer Vision. (2012)
55. Marin-Jimenez, M.J., Yeguas, E., de la Blanca, N.P.: Exploring STIP-based models for recognizing human interactions in TV videos. PRL **34** (2013) 1819–1828

56. Jiang, Y.G., Dai, Q., Xue, X., Liu, W., Ngo, C.W.: Trajectory-based modeling of human actions with motion reference points. In: Proceedings of the European Conference on Computer Vision. (2012)
57. Mathe, S., Sminchisescu, C.: Dynamic eye movement datasets and learnt saliency models for visual action recognition. In: Proceedings of the European Conference on Computer Vision. (2012)
58. Gaidon, A., Harchaoui, Z., Schmid, C.: Recognizing activities with cluster-trees of tracklets. In: Proceedings of the British Machine Vision Conference. (2012)
59. Kliper-Gross, O., Gurovich, Y., Hassner, T., Wolf, L.: Motion interchange patterns for action recognition in unconstrained videos. In: Proceedings of the European Conference on Computer Vision. (2012)
60. Peng, X., Zou, C., Qiao, Y., Peng, Q.: Action recognition with stacked fisher vectors. In: Proceedings of the European Conference on Computer Vision. (2014)
61. Jain, M., Jégou, H., Bouthemy, P.: Better exploiting motion for better action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2013)
62. Yu, G., Yuan, J., Liu, Z.: Propagative hough voting for human activity recognition. In: Proceedings of the European Conference on Computer Vision. (2012)
63. Hoai, M., Zisserman, A.: Talking heads: Detecting humans and recognizing their interactions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2014)