Universidad Autónoma
de Madrid

Biblos-e Archivo
Repositorio Institucional UAM

Repositorio Institucional de la Universidad Autónoma de Madrid

<https://repositorio.uam.es>

Esta es la **versión de autor** del artículo publicado en:
This is an **author produced version** of a paper published in:

Pattern Recognition Letters 105 (2018): 4 – 12

DOI: <https://doi.org/10.1016/j.patrec.2017.06.010>

Copyright: © 2017 Elsevier B.V. All rights reserved

El acceso a la versión del editor puede requerir la suscripción del recurso

Access to the published version may require subscription

Aggregating the temporal coherent descriptors in videos using multiple learning kernel for action recognition

Adel Saleh^{a,**}, Mohamed Abdel-Nasser^c, Miguel Angel Garcia^b, Domenec Puig^a

^a*Intelligent Robotics and Computer Vision group, Departament d'Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili, Av. Països Catalans, 26, 43007 Tarragona, Spain*

^b*Department of Electronic and Communications Technology, Autonomous University of Madrid, Madrid, Spain*

^c*Electrical Engineering Department, Faculty of Engineering, Aswan University, 81542 Aswan, Egypt*

ABSTRACT

Action recognition methods enable several intelligent machines to recognize human action in their daily life videos. Indeed, many action recognition methods give a noticeable misclassification rate due to the big variations within the videos of the same class, and the changes in viewpoint, scale and background. In this paper, we propose a new video representations method that captures temporal evolution of the action happening in the whole video. We show that 1) combining the descriptors of improved dense trajectories with a multiple kernel learning technique can reduce the misclassification rate, and also 2) aggregating the coherent frames in each video may have a different impact on the recognition results. Our experimental results using HMDB51 and Hollywood datasets demonstrate that our method is on par with the state-of-the-art methods.

1. Introduction

The main goal of the action recognition field is to give the computer the ability to recognize human action in real life videos. The need for action recognition methods in areas such as robot learning, analyzing video surveillance and human-computer interaction led to a big progress in the field. However, constructing robust models to recognize free-form activities of the same class still an open problem (Wang et al., 2016). One of the sources of misclassification is the wide margin of variation inside the same class. Other sources can be the change in viewpoint, scale and background clutter. In addition, there are other high level unmeasurable factors that can play a role in human action recognition, such as human-objects interaction, human poses and scene context. In fact, most of the development in the action recognition area over the last years has been related to two approaches. The

first approach is the definition of local spatio-temporal features, such as spatio-temporal features (Laptev, 2005) and densely sampled local visual features (Laptev et al., 2008). We can also mention interest points, motion-based gradient descriptors (Jain et al., 2013) and dense trajectories (Wang et al., 2013). The second approach is concentrated on the exploration of encoding methods, which has a long history of success in object recognition field. A good example is Fisher vectors (Wang and Schmid, 2013). In spite of the fact that the interest in action and events recognition has been increased (Tang et al., 2012; Izadinia and Shah, 2012), there are not a lot of works to model temporal patterns inside videos. Obviously, modeling the temporal evolution of video is a challenging open problem due to the level of complexity mentioned before. In addition, actions in videos are performed at varying speeds, and often the speed of the same action can non-linearly change within a single video. A number of methods have been proposed to capture temporal evolution in videos for action recognition tasks exploiting the learning-to-rank technique (Fernando et al., 2016) and hidden Markov models (Wang and Mori, 2011; Wu and Shao, 2014). In the last few years, conditional random fields based methods were also proposed (Song et al., 2013), and recently, deep neural networks based methods were proposed to tackle the activity recognition problem (Simonyan

^{**}Corresponding author.

e-mail: adelsalehali1982@gmail.com (Adel Saleh),
egnaser@gmail.com (Mohamed Abdel-Nasser),
miguelangel.garcia@uam.es (Miguel Angel Garcia),
domenec.puig@urv.ca (Domenec Puig), adelsalehali1982@gmail.com (Domenec Puig)

and Zisserman, 2014; Husain et al., 2016).

One can easily realize that the impact of these works on the action recognition performance has been disappointing so far. Simpler but more powerful approaches, such as temporal pyramids, which are similar to spatially divide the images or objects are not enough. Nevertheless, it became clear that many activities have ordered temporal patterns. In this paper, we propose a new video representation method that captures the temporal evolution of the action in the whole video. We show that the aggregation of coherent frames can have a different impact on the results. We also propose clever combinations of different features based on multiple kernel learning.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 explains the proposed method. Section 4 presents and discusses the experimental results. The conclusion and some lines of the future work are given in Section 5.

2. Related Work

In the literature, several researchers have studied the problem of capturing temporal patterns of videos for activity recognition. Important improvements have been seen in local motion patterns modeling. Jain et al. (2013) proposed to determine the location of actions in the video and then use them to refine recognition. Saleh et al. used the curvature of the trajectories of local descriptors and other kinematic properties to improve performance of activity recognition methods.

To avoid designing features, it is reasonable to apply deep learning methods (Le et al., 2011; Taylor et al., 2010). The temporal pattern can thus be captured by the convolutional neural network (CNN) using two approaches. The first approach extends the connectivity of the network architecture along time. An example of this approach was proposed in (Karpathy et al., 2014). The second approach stacks optical flow frames and inputs them into a CNN (Simonyan and Zisserman, 2014). This approach is called two-stream neural network because it has two kinds of inputs: raw RGB and stacked optical flow. This network can learn visual appearance and motion information in an unsupervised way from video cubes. Indeed, CNNs have been oriented to capture different types of information. For instance, Chéron et al. (2015) represented different parts of human body with motion and appearance based descriptors to extract features from human pose. These descriptors were calculated from each frame and then aggregated over time to form a video descriptor. They captured temporal patterns by considering temporal differences between frames and then concatenated the difference of vectors. In (Gkioxari and Malik, 2015), two CNNs were used to capture both appearance and motion based features in action tubes. The first network is called spatial-CNN network, which takes RGB frames as input and captures the appearance information of the actor and other visual details from the scene as well. The second network is called motion-CNN, which works on the optical flow and captures the temporal patterns of the actors. By combining the output of the hidden layers of the two networks, they extract spatio-temporal features. Srivastava

et al. (2015) used the state of the long-short term memory (LSTM) encoder after observing the last input frame as a video representation. Du et al. (2015) showed that skeleton based action recognition can be managed using a hierarchical recurrent neural network. Another popular way to tackle action recognition using deep neural networks is to combine the CNN and LSTM as demonstrated in (Yue-Hei Ng et al., 2015).

Jain et al. (2015) showed how they can take advantage of objects in the videos for action classification. The aforementioned methods have a common point, which is capturing the local changes in small temporal window, but they are not designed to capture motion patterns to model the whole video, which is an important factor in some actions.

Hidden Markov models (HMM) and discriminative conditional random fields (CRF) have been also proposed to model dynamics of videos along time (Yamato et al., 1992; Sminchisescu et al., 2006). HMMs are used to learn a joint distribution over both observations and action targets. Thus, the observations are the visual appearance and/or local motion feature vectors acquired from input videos. HMMs can learn the appearance and/or the motion pattern along time of a given action class. The biggest challenge is to learn different variations of one action class. Because of that, HMMs may require a lot of training samples to capture discriminative joint probability distributions. In turn, CRFs can learn to discriminate a pair of action classes by modeling the conditional distribution over their class labels. Like the HMMs, CRFs methods also need a large amount of input samples to train and capture all parameters of the models. The proposed method does not depend on the target class to extract temporal patterns from the video sequence. It relies on standard aggregation of video specific dynamic information, support vector regression (SVR) and support vector machines (SVM) to discriminate between action classes.

In the recent past, new approaches based on CRFs, HMMs and action grammars were proposed to model motion patterns for action recognition tasks (Song et al., 2013; Tang et al., 2012; Wang and Mori, 2011). Wang and Mori (2011) combined a part-based approach with large-scale template features to obtain a discriminative model based on max-margin CRFs. Song et al. (2013) defined a feature function for the aforementioned CRFs model using a series of complex heuristics. In addition, Tang et al. (2012) proposed a max-margin method to model the temporal structure in a video. They used an HMM to learn the transitions of action appearances and the duration of actions.

In the recent literature, one can find works that apply temporal ordering models for activity recognition tasks (Izadinia and Shah, 2012; Ponce-López et al., 2015; Sun and Nevatia, 2013). Their aim was to infer complex actions from predefined useful basic-level action detectors. Sun and Nevatia (2013) imposed a representation for low-level action events that includes statistical information of the action transition probabilities. Spatio-temporal primitives of sub-gestures were used to model the goal actions after detecting them through genetic algorithms. The dynamics of the target actions were modeled using HMMs or dynamic time warping. Owing to the variability of motion patterns in videos, latent sequential

models are not capable to overcome this problem. Thus, temporal pyramids were introduced in (Gaidon et al., 2012; Laptev et al., 2008) to address this problem. A method that applies Fourier analysis was also proposed in (Revaud et al., 2013). In this paper, we model the temporal structure of videos following (Fernando et al., 2016), which models the evolution of appearance or local motion using a learning-to-rank technique. We show the effect of aggregation of coherent frames. We also demonstrate how useful is to process input videos as subsequences of related frames.

3. Methods

Fig. 1 presents the steps of the proposed method: feature extraction, coherence analysis, weighted combination, representation learning and classification. In the feature extraction step, we compute local descriptors from each frame ($F_1, F_2, \dots, F_N$), and then we encode each type of descriptors using Fisher vector technique ($FVdesc_1, FVdesc_2, \dots, FVdesc_k$). In the coherence analysis step, we automatically determine the subsequences in each video. In the weighted combination step, for each subsequence we use an aggregation operator to combine the Fisher vectors of each type of descriptors extracted from each frame ($FVdesc_1, FVdesc_2, \dots, FVdesc_k$) into one Fisher vectors ($FVdesc_c$), and then we use the multiple kernel learning (MKL) technique to produce a weighted combination (FV_{sub}) of the Fisher vectors of all descriptors ($FV1_c, FV2_c, \dots, FVN_c$). In the representation learning step, we input the Fisher vectors of all subsequences into a learning-to-rank technique to produce a representation of the whole video. Finally, the representations of all videos are input into a SVM classifier to discriminate between the actions.

Fig. 1: The steps of the proposed method.

3.1. Feature extraction

Feature trajectories have shown to be efficient for representing videos. Wang and Schmid (2013) used dense sampled local visual features at several spatial scales. They estimated a homography with RANSAC (random sample

consensus) using SURF matching (speeded up robust features) between two consecutive frames. The calculated low-level descriptors were computed on the warped optical flow to capture motion patterns. Below, we explain the feature extraction methods used in this paper: histogram of oriented gradients (HOG), histogram of optical Flow (HOF), and motion boundary histogram (MBH). We also explain the steps of Fisher vectors (FVs).

- **HOG and HOF descriptors:** HOG captures information about the appearance whereas HOF captures local motion information. Wang et al. (2009) reported that HOG and HOF descriptors yield good results on different activity recognition datasets compared to classical descriptors. To compute the HOG descriptor, we computed gradient magnitude responses in the horizontal and vertical directions. To compute the HOF descriptors, optical flow displacement vectors in the horizontal and vertical directions were determined (Wang and Schmid, 2013). As a result, a 2D vector field per frame was calculated. The orientations were quantized into eight bins for HOG and nine bins for HOF as described in (Laptev et al., 2008). The length of HOG was 96 ($2 \times 2 \times 3 \times 8$) while the length of HOF was 108 ($2 \times 2 \times 3 \times 9$).
- **MBH descriptor:** MBH is a very popular descriptor for video classification and detection tasks. Dalal et al. (2006) showed the robustness of MBH descriptor against camera and background motion. The simple idea behind MBH is computing oriented gradients over the vertical and horizontal optical flow displacements. Indeed, the horizontal and vertical displacements of the optical flow are treated as gray-level images of the motion displacements. For each optical flow component, a histogram of oriented gradients is computed using the same configuration used for still images. Indeed, information related to motion changes in the object boundaries is obtained using the flow difference whereas information about unmoving objects and camera motion is discarded.
- **Fisher vectors (FVs):** In computer vision field, FVs have been widely used as an image representation by pooling local image features. It is also used as a global descriptor for image classification. FV can be used for frame level pooling, a subsequence level pooling or video level pooling as well. FV coding was derived from the well-known Fisher kernel, which is based on the assumption that the generation process of a local descriptor X can be modeled as a probability density function $p(X, \theta)$. Using the gradient of the G log-likelihood, it is possible to describe how the parameters in θ contribute to the generation process of X . Then, the sample can be described as follows:

$$G_{\theta}^X = \nabla_{\theta} \log(p(X; \theta)) / N \quad (1)$$

The dimensionality of this vector depends on the number of parameters in θ . A Gaussian mixture model is used

to model the probability density function. Note that $\theta = \{\pi_1, \mu_1, \sigma_1, \dots, \pi_K, \mu_K, \sigma_K\}$ contains the parameters of the model, where π, μ, σ are Gaussian mixture weights, means and diagonal covariance, respectively. An improved Fisher vector was proposed in (Perronnin et al., 2010) as follows:

$$\rho_k = \frac{1}{\sqrt{\pi_k} \gamma_k (x - \mu_k) / \sigma_k} \quad (2)$$

$$\tau_k = \frac{1}{\sqrt{\pi_k ((x - \mu_k)^2 / \sigma_k^2 - 1)}} \quad (3)$$

where γ_k is the weight of local descriptor x corresponding to k^{th} Gaussian component, and thus

$$m = \pi_1 N(x; \mu_1; \sigma_1) + \dots + \pi_K N(x; \mu_K; \sigma_K). \quad (4)$$

The parameter γ_k can be determined as follows:

$$\gamma_k = \pi_k N(x; \mu_k; \sigma_k) / m \quad (5)$$

To construct the FV, the gradients of Eq. 2 and 3 are concatenated as follows:

$$FV = [\rho_1, \tau_1, \dots, \rho_K, \tau_K] \quad (6)$$

3.2. Coherence Analysis

To find the coherent frames, we calculate dense HOG descriptors for each frame in the input video $H = [h_1, h_2, \dots, h_N]$, where N indicates the number of frames of the input video. Assume that h_t and h_{t+1} are HOG descriptors of two consecutive frames, we calculate the similarity between them. Thus, we determine a similarity vector for each video, $S = [s_1, s_2, \dots, s_{n-1}]$. To decide if two consecutive frames are coherent, we use the following hard threshold:

$$S'(t) = \begin{cases} 1 & S(t) \geq Z_{th} \\ 0 & S(t) < Z_{th} \end{cases} \quad (7)$$

where Z_{th} is the mean of the similarity measure vector S , '1' indicates that the two frames are coherent and they belong to the same subsequence. In turn, '0' indicates that the two frames do not belong to the same subsequence. In this paper, we evaluate the impact of three similarity measures on the recognition rate (correlation, Euclidean distance and cosine measures). Fig. 2 shows an example of the coherence analysis in a video. Note that yellow circles include the frame number, green circles show the similarity indicator, and each red rectangle represents one subsequence (i.e., coherent frames). Frames 1, 2 and 3 are coherent (they lie in the same subsequence) while frames 3 and 4 are not coherent (they lie in the different subsequence). Although we used the HOG descriptor for coherence analysis, any other descriptor can be used to perform that task.

3.3. The weighted combination

In this subsection, we explain how we use an aggregation method to combine the FVs of each type of descriptors. In addition, we explain how we use the MKL technique to produce a weighted combination of the FVs of all descriptors.

Fig. 2: An example of coherence analysis in a video.

3.3.1. Aggregation methods

For each subsequence, we aggregate the FVs of each type of descriptors (HOG, HOF and MBH) extracted from each frame ($FVdesc_1, FVdesc_2, \dots, FVdesc_k$) into one FV ($FVdesc_c$) using an aggregation operator. In this paper, we evaluate the performance of two aggregation operators: mean and median. For example, we can use the mean operator to aggregate the FVs of HOG descriptors ($FVHOG_1, FVHOG_2, \dots, FVHOG_k$) extracted from a subsequence of k frames as follows: $FVHOG_c = \frac{1}{k} \sum_{a=1}^k FVHOG_a$. This step reduces the complexity of the problem because it reduces the dimension of the vector that we input into the MKL algorithm.

3.3.2. Multiple kernel learning (MKL)

MKL methods are used to learn an optimal linear or non-linear combination of predefined kernels. MKL methods have been used for long time in many applications, such as event recognition, object recognition in images and biomedical data fusion (Chen et al., 2013; Bucak et al., 2014; Yu et al., 2010). Obviously, the nature of these resources are different, consequently they have different notions of similarity and thus require different kernels. MKL algorithms can be used to combine kernels already established from different data sources instead of creating a new kernel.

In this paper, we input the FVs obtained from the aggregation step of each type of descriptors ($FVHOG_c, FVHOF_c$ and $FVMBH_c$) into a MKL algorithm to produce a weighted combination (FV_{sub}) for each subsequence. Indeed, there are a lot of reasons to use MKL, such as:

- It gives us the ability to choose optimal kernel and parameters from a larger set of kernels (which is the case in practice). This automatic selection reduces the bias.
- It provides an intelligent way to combine data from different sources (in this paper we combine the FVs of HOG, HOF and MBH).

MKL algorithms have been developed for supervised, semi-supervised and unsupervised learning. Big part of the progress has been done on supervised learning with linear combinations of kernels. One can explain MKL algorithms as the addition of an extra parameter to the minimization problem of the learning algorithm. For instance, in case of supervised learning of a linear combination of a set of n kernels. Let us introduce a kernel $K' = \sum_{i=1}^n \beta_i K_i$ where β is coefficients vector for kernels. Since the kernels are additive, the new function is still a kernel. Given a set of data X with labels Y , one can

rewrite the minimization problem as follows: $\min_{\beta} E(Y, K') + R(K)$, where R is a regularization term and E is the error function. The definition of E depends on the solving method, which can be the square loss function or the hinge loss function (for SVM algorithms), and R is an ℓ_n -norm or any combination of the norms. This optimization problem can then be solved by any of standard optimization methods (Bach et al., 2004).

Notation: In this work, we follow Aioli and Donini (2015) to solve a classification problem with training samples defined as $\{(x_1, y_1), (x_2, y_2), \dots, (x_L, y_L)\}$ and test samples defined as $\{(x_{L+1}, y_{L+1}), (x_{L+2}, y_{L+2}), \dots, (x_L, y_L)\}$, where $x \in \mathbb{R}^m$ and $y \in \{-1, 1\}$ if we solve a binary classification problem or $y \in \{1, 2, \dots, C\}$ if we solve a multi-class classification problem. For simplicity, suppose that our problem is binary. Let $X \in \mathbb{R}^{L \times m}$ denote the matrix where samples are arranged in rows and $y \in \mathbb{R}^L$ the vector of labels. Let $K \in \mathbb{R}^{L \times L}$ denote the kernel matrix, which contains the kernel values of each data point (training and testing). Given a training set, the domain of probability distributions $\gamma \in \mathbb{R}_+^L$ defined over the sets of positive and negative training examples can be denoted as $\hat{\Gamma}$:

$$\hat{\Gamma} = \left\{ \gamma \in \mathbb{R}_+^L, \sum_{i \in \Phi} \gamma_i = 1, \sum_{i \in \Theta} \gamma_i = 1 \right\} \quad (8)$$

Note that any element γ in $\hat{\Gamma}$ corresponds to pair of data points.

A Scalable multiple kernel learning algorithm (EasyMKL): In this paper, we use the EasyMKL method (Aioli and Donini, 2015) because it can easily deal with hundreds of thousands of kernels and more. EasyMKL method showed robustness in comparison with other alternative methods (random, average, etc.) even for data containing noise. EasyMKL method learns a vector of coefficients η that define the combined kernel according to:

$$K = \sum_{r=1}^R \eta_r K_r, \eta_r \geq 0 \quad (9)$$

It is clear that one must restrict the possible choices of K , and this can be done by a regularization term. The problem of learning the kernel combination is posed as a $\min - \max$ problem over variables γ and η . Aioli and Donini (2015) proposed to maximize the distance between the positive and negative samples with a unitary norm vector η as the weak kernels combination vector:

$$\max_{\eta: \|\eta\|=1} \min_{\gamma \in \hat{\Gamma}} (1 - \lambda) \gamma^T \hat{Y} \left(\sum_{r=1}^R \eta_r \hat{K}_r \right) \hat{Y} \gamma + \lambda \|\gamma\|^2 \quad (10)$$

Consider the following equation:

$$Q(\eta, \gamma) = (1 - \lambda) \gamma^T \hat{Y} \left(\sum_{r=1}^R \eta_r \hat{K}_r \right) \hat{Y} \gamma + \lambda \|\gamma\|^2 \quad (11)$$

If the vector of the r th entry is defined as $d_r(\gamma) = \gamma^T \hat{Y} \hat{K}_r \hat{Y} \gamma$, then

$$Q(\eta, \gamma) = (1 - \lambda) \eta^T d(\gamma) + \lambda \|\gamma\|^2 \quad (12)$$

Consequently, the main problem can be rewritten as:

$$\min_{\gamma \in \hat{\Gamma}} \max_{\eta: \|\eta\|=1} Q(\eta, \gamma) = \min_{\gamma \in \hat{\Gamma}} \max_{\eta: \|\eta\|=1} (1 - \lambda) \eta^T d(\gamma) + \lambda \|\gamma\|^2 \quad (13)$$

Finding η^* , which is a simple analytic solution that maximizes Q :

$$\eta^* = \frac{d(\gamma)}{\|\gamma\|_2} \quad (14)$$

and

$$\min_{\gamma \in \hat{\Gamma}} Q(\eta^*, \gamma) = \min_{\gamma \in \hat{\Gamma}} (1 - \lambda) \|d(\gamma)\|_2 + \lambda \|\gamma\|_2^2 \quad (15)$$

The optimal γ is a regularized minimizer of ℓ_2 -norm of the distances vector. It is possible to find the minimizer as this is a convex function. To simplify the problem further, Aioli and Donini (2015) proposed to minimize an upper-bound instead of corresponding to the ℓ_1 -norm of the distances, and thus they obtained the following equation:

$$\min_{\gamma \in \hat{\Gamma}} (1 - \lambda) \|d(\gamma)\|_1 + \lambda \|\gamma\|_2^2 = \min_{\gamma \in \hat{\Gamma}} (1 - \lambda) \gamma^T \hat{Y} \left(\sum_{r=1}^R \hat{K}_r \right) \hat{Y} \gamma + \lambda \|\gamma\|_2^2 \quad (16)$$

3.4. Representation learning

To learn a representation of each video, we use the learning-to-rank technique, which consists of the following three steps: frame-level representation, sequence smoothing, and learning the representation of the whole video (Fernando et al., 2016).

1. **Frame-level representation:** Each video can be represented as follows: $X = [x_1, x_2, \dots, x_n]$, where x_t contains the descriptors of a frame at time step t .
2. **Sequence smoothing:** To avoid abrupt action changes in videos, we smooth the sequence X using time varying mean as follows: $m_t = \frac{1}{t} \sum_{i=1}^t x_i$. We then normalize m_t as follows: $v_t = \frac{m_t}{\|m_t\|}$. It is clear that v_t only contains information about direction of m_t . Therefore, this approach guarantees a smooth output.
3. **Learning the representation of the whole video:** Learning-to-rank technique was proposed in (Fernando et al., 2016, 2015) for activity recognition. Given the smoothed vectors, the learning-to-rank technique learns their rank (i.e., $m_n > m_{n-1} > m_{n-2} > \dots > m_1$). A linear learning-to-rank technique learns a pairwise linear function $\psi(m_i; u)$, where $u \in \mathbb{R}^D$ is a vector containing its parameters. The ranking score of m_i is computed by $\psi(m_i; u) = u^T \cdot v_i$. The learning-to-rank technique optimizes the parameters u of $\psi(m_i; u)$ using the objective function of Eq. 17 with the constraint $\forall t_i, t_j \quad v_{t_i} > v_{t_j} \iff v_{t_i} \cdot u^T > v_{t_j} \cdot u^T$ as follows:

$$\arg \min_u \frac{1}{2} \|u\|^2 + C \sum_{\forall i, j, v_i > v_j} \epsilon_{ij}, \quad s.t. \quad u^T (v_{t_i} - v_{t_j}) \geq 1 - \epsilon_{ij}, \quad \epsilon_{ij} \geq 0. \quad (17)$$

where C is the regularization parameter and ϵ is the margin of tolerance. The parameters u are used to describe the evolution of actions in videos. In other words, we use u to represent the whole video.

3.5. Classification

In the classification step, we input the learned representations of the videos into a SVM classifier with one-against-all strategy to classify them. A SVM classifier is a supervised learning classifier that discriminates between two classes by finding a hyperplane that separates them. Given a labeled training set of the form $(x_i, y_i), i = 1, 2, \dots, k$, where $x_i \in \mathbb{R}^n$ are the feature values, $y_i \in \{1, -1\}$ is the binary classification, n is the number of features and k is the number of instances, the SVM classifier solves the following optimization problem:

$$\begin{aligned} \|\omega\|_{\omega, \xi}^2 + C \sum_{i=1}^k \xi_i \\ \text{s.t. } y_i(\omega^T \phi(x_i) + b) \geq 1 - \xi_i, \\ \xi_i \geq 0. \end{aligned} \quad (18)$$

In this equation, the soft margin parameter C helps the SVM optimization process to avoid misclassifying each training instance. The weight vector ω is normal to the separating hyperplane. The parameter ξ is used to give a degree of flexibility to the algorithm when fitting the data and b represents the bias.

Each sample x_i is mapped into a higher dimensional space using a kernel function, $K(x_i, x_j) = (\phi^T(x_i) \cdot \phi(x_j))$ to make the data linearly separable in the new space. Then, the SVM classifier determines a hyperplane with a maximum margin of separation between the classes. In the case of a linear SVM classifier, ϕ refers to a dot product. In the case of non-linear SVM, the classifier function is formed by non-linearly projecting the training data of the input space to a feature space of a higher dimension by using a kernel function. In this paper, we used a SVM with RBF- χ^2 kernel (radial basis function).

4. Experimental results and discussion

4.1. Datasets

We used two state-of-the-art datasets to evaluate the proposed method: HMDB51 and Hollywood2. Hollywood2 is one of the largest and most challenging available datasets for real world actions. There are 12 classes: answering phone, driving car, eating, fighting, getting out of car, shaking hands, hugging, kissing, running, sitting down, sitting up and standing up. The actions were collected from a set of 69 Hollywood movies. It consists of around 487k frames (about 20 hours of video). It is divided into a training set of 823 sequences and a testing set of 884 sequences. There is no overlap between the 33 movies in the training set and the 36 movies in the testing set (Marszalek et al., 2009). As is standard on this dataset, performance of our method is measured by mean average precision (mAP) over all classes.

HMDB51 is a generic action classification dataset (Kuehne et al., 2011). The videos of this dataset were collected from YouTube and some movies. It contains 6670 videos, which were further grouped into 51 action classes, in which each class contains around 100 videos. To measure the performance of the proposed method, we used three training and testing splits and then the average accuracy over them was computed.

4.2. Experiments Setup

To implement FVs, we set the number of Gaussians to $K=256$ and randomly sample a subset of 256,000 features from the training set to estimate the GMM. Recent evaluations (Chatfield et al., 2011; Oneata et al., 2013; Wang et al., 2013) showed that the aforementioned values give good results with action classification. The resulting GMM models were used to build FVs. We used VIFeat library to implement this part. For the learning-to-rank method, we used the source code of (Fernando et al., 2015). For MKL method, we used the source code of (Aioli and Donini, 2015), we set the value of λ to 0.1 and the value of γ of the radial basis function to 0.1. In our experiments, we set $C = 1.0$ (regularization-loss trade off) for the SVM classifier, which gave good results when validating on a subset of training samples.

4.3. Results

We assess the impact of three similarity measures (correlation, Euclidean distance and cosine measures) and two aggregation operators (mean and median) on the performance of the proposed method. As shown in Table 1 and 2, the cosine similarity and the correlation measure gave the same recognition rates with Hollywood2 dataset because they can be viewed as variants of the inner product. The mean operator is sensitive to extreme observations, and thus it can be pulled to the direction of the outliers (outliers are extreme values that differ greatly from other values). In general, the median operator may provide a better performance with outliers. In the case of Hollywood2 dataset, each subsequence consists of homogeneous frames, and this clearly explains why we have identical values in Table 1 and 2.

In Table 3 and 4, we present the recognition accuracy with HMDB51 dataset. As we can see, the cosine similarity and the correlation measure also gave the best recognition results. We notice that in both datasets, the median function gave recognition accuracy better than the one of the mean operator with the Euclidean distance. As we see in Tables 1–4, the recognition results are improved when we combine the proposed method with the baseline of Fernando et al. (2015). After Applying the MKL technique, we got the state-of-the-art results with Hollywood2 dataset and comparable values with HMDB51 dataset.

In Table 5 we show the performance of the proposed method with the two datasets. With Hollywood2 dataset, we achieved an accuracy of 70.2. The application of coherence analysis and Fisher vector aggregation of the obtained subsequence information can help beating the state-of-art methods. It also outperforms the performance of recent deep learning approaches, such as (Sharma et al., 2015) and (Srivastava et al., 2015).

It is clear from Table 5 that the results of the proposed method with HMDB51 and Hollywood2 datasets are comparable to the ones of (Fernando et al., 2015), as a result of the existence of a big number of slow actions in the datasets. The CNN model proposed in (Simonyan and Zisserman, 2014) has a good performance because it represents a higher level of semantic concepts, but it has a high time complexity and requires

Table 1: The impact of changing similarity measures and the use of mean function for aggregation on the recognition rate with Hollywood2 dataset. Each row contains the mAP rate of each similarity metric.

Method	Proposed method	Proposed+ (Fernando et al., 2015)
Euclidean distance	38.8	67.2
Cosine	46.8	69.2
Correlation	46.8	69.2

Table 2: The impact of changing similarity measures and the use of median function for aggregation on the recognition rate with Hollywood2 dataset. Each row contains the mAP rate of each similarity metric.

Method	Proposed method	Proposed+ (Fernando et al., 2015)
Euclidean distance	43.1	68.1
Cosine	46.8	69.2
Correlation	46.8	69.2

Table 3: The impact of changing similarity measures and the use of mean function for aggregation on the recognition rate with HMDB51 dataset. Each row contains the recognition accuracy of each similarity metric.

Method	Proposed method	Proposed+ (Fernando et al., 2015)
Euclidean distance	31.1	53.8
Cosine	37.4	55.4
Correlation	37.4	55.4

Table 4: The impact of changing similarity measures and the use of median function for aggregation on the recognition rate with HMDB51 dataset. Each row contains the recognition accuracy of each similarity metric.

Method	Proposed method	Proposed+ (Fernando et al., 2015)
Euclidean distance	34.5	54.6
Cosine	37.4	55.4
Correlation	37.4	55.4

Table 5: Comparison between the proposed method and related works using HMDB51 and Hollywood2 datasets.

Method	HMB51	Hollywood2
Two-stream ConvNet (Simonyan and Zisserman, 2014)	59.40	-
Learning-to-rank (Fernando et al., 2015)	61.80	70.00
Composite LSTM Mode Srivastava et al. (2015)	44.00	-
Kinematic of the Trajectories	58.20	-
Coherence Analysis + FVs aggregation	40.55	65.00
Coherence Analysis+ Learning-to-rank	61.90	70.20
Coherence Analysis + Learning-to-rank + MKL	62.1	69.73

complicated training passes. In turn, the proposed method is much faster. The model proposed in (Simonyan and Zisserman, 2014) took approximately one day for one temporal CNN on a system consisting of four *NVIDIA Titan cards* (it took 3.1 times the training time if a single GPU was used). In turn, our approach took approximately 16 hours on Core i7 2.5GHz CPU with 16 GB RAM.

Fernando et al. (2015) used the *time varying mean* method for smoothing the feature vectors in order to cope with abrupt changes in human action. They did not use the *moving average* (MA) method for that task because of two reasons:

1. A window size must be chosen, which is not always straightforward as actions often take place at different times.
2. The last frames of a video are ignored.

In our method we automatically divide each video into a set of subsequences, therefore we do not need to define a window size to aggregate or smooth the feature vectors. The window size is defined by the length of each subsequence (adaptive window size). In this way, we avoid the main problem of using the MA method. Furthermore, we compute the mean or the median over an adaptive window size, therefore we guarantee that each vector in the time series is included.

The main reason for the low recognition rates is that temporal coherence can be a weak measure of discriminability with fast actions. If the frames are quite distinct, then maximizing the temporal coherence directly leads to a weak discriminative features. After coherence analysis, if we get distinct frames in each subsequence, we lose the power of using subsequences, which is the case in fast actions. Indeed, slow change across

frames means slow actions. Slow actions make the aggregation more effective. As shown in Table2, there is a degradation in performance of our method when we used the median operator for subsequence aggregation because the subsequences contain few frames (due to the fast actions).

Fig. 3: The confusion matrix of Hollywood2 dataset.

The proposed method also outperforms fresh deep learning approaches because the coherence analysis step suppresses the redundant information in the videos. In other words, we suppress unnecessary information via computing one FV for each subsequence. Fig. 3 shows the confusion matrix of Hollywood2 dataset. As shown in Table 7, with Hollywood2 database our method gives better results in the case of slow actions (with less dramatic changes) as in the following cases: AnswerPhone, FightPerson, SitDown, SitUp and StandUp. The proposed method works better than the baseline of Fernando et al. (2015) in the case of slow actions since it makes the input sequences more generalized and lightens the unnecessary details.

Table 6 shows the diagonal elements of the confusion matrix of HMDB51 database. Note that we cannot display the complete confusion matrix because it has a size of 51×51 . The complete confusion matrix is available at https://www.dropbox.com/s/31jik8zk9v6pbq9/confusion_matrix.csv.zip?dl=0.

Table 8 shows the performance of the proposed method

Table 6: The diagonal elements of the confusion matrix of HMDB51 database.

Class	Accuracy	Class	Accuracy	Class	Accuracy
Brush hair	0.76	Hit	0.57	Shoot ball	0.89
Cartwheel	0.52	Hug	0.90	Shoot bow	1.00
Catch	1.00	Jump	0.38	Shoot gun	0.85
Chew	0.47	Kick ball	0.31	Sit	0.39
Clap	0.46	Kick	0.68	Situp	1.00
Climb stairs	0.98	Kiss	0.93	Smile	0.44
Climb	0.65	Laugh	0.41	Smoke	0.43
Dive	0.85	Pick	0.27	Somersault	0.52
Draw	0.47	Pour	1.00	Stand	0.27
Dribble	1.00	Pullup	1.00	Swing	0.16
Drink	0.51	Punch	0.51	Sword exercise	0.42
Eat	0.51	Push	0.848	Sword	0.13
Fall floor	0.35	Pushup	1.00	Talk	0.67
Fencing	0.61	Ride bike	1.00	Throw	0.16
Flic flac	0.68	Ride horse	1.00	Turn	0.222
Golf	1.00	Run	0.41	Walk	0.38
Handstand	0.73	Shake hands	0.82	Wave	0.14

with 12 randomly selected actions from the 51 classes of the HMDB51 dataset. We obtained better recognition with the following actions: Chew, Climb, Climbs tair, Dribble, Drink, Eat, Fall floor, and Kick. All these actions are slow. The main reason of these results is the difficulty of the recognition of some videos and the high variation within the videos of several classes. We notice that the available action recognition datasets lack of diversity. Consequently, the trained models are suffering from overfitting, thus training and testing datasets should be collected in the cross category style to generalize actions and avoid over-fitting.

Table 7: Analyzing the performance of the proposed method and (Fernando et al., 2015) with Hollywood2 dataset.

Action	(Fernando et al., 2015)	Proposed method
AnswerPhone	0.39	0.41
DriveCar	0.96	0.95
Eat	0.74	0.73
FightPerson	0.84	0.85
GetOutCar	0.74	0.72
HandShake	0.55	0.52
HugPerson	0.53	0.53
Kiss	0.63	0.63
Run	0.89	0.88
SitDown	0.80	0.81
SitUp	0.45	0.50
StandUp	0.83	0.84

Table 8: Analyzing the performance of the proposed method and (Fernando et al., 2015) with the HMDB51 dataset.

Action	(Fernando et al., 2015)	Proposed method
Brush hair	0.89	0.76
Cartwheel	0.52	0.52
Chew	0.39	0.47
Clap	0.84	0.46
Climb stairs	0.52	0.98
Climb	0.62	0.65
Dive	0.53	0.85
Dribble	0.56	1.00
Drink	0.42	0.51
Eat	0.37	0.51
Fall floor	0.32	0.35
Kick ball	0.23	0.31

5. Conclusion

In this paper, we have modeled the evolution of human actions for activity recognition. The proposed method consists of five steps: feature extraction, coherence analysis, weighted combination, representation learning and classification. In the feature extraction step, we used HOG, HOF and MBH feature extraction methods, and then we encoded each type of descriptors using Fisher vector technique. In the coherence analysis step, we determined the subsequences of each

video. We evaluated the effect of three similarity measures (correlation, Euclidean distance and cosine measures) on the performance of the proposed method. The cosine similarity and correlation measure gave the same recognition rates. For each subsequence, we used an aggregation method to combine the FVs of each type of descriptors extracted from each frame into one FV. To do so, we used the mean and median operators. We used the MKL technique to produce a weighted combination of the FVs of all descriptors. We input the FVs of all subsequences into a learning-to-rank method to produce a representation of the whole video. The representations of all videos were fed into a SVM classifier to discriminate between the actions. We showed the effect of the aggregation of coherent frames on the recognition rate. We also demonstrated how useful is to process input videos as subsequences of related frames. With HMDB51 and Hollywood databases, we showed how different methods of aggregation can change the recognition performance and combining the aggregated FVs using the MKL method can have a positive impact on the recognition results.

To further improve the results, the future work will focus on using features extracted from deep CNNs in the feature extraction step. We will also use the dynamic image technique in the representation learning step.

Acknowledgment

This work was partly supported by Universitat Rovira i Virgili (Spain), and Hodeidah University (Yemen).

References

- Aioli, F., Donini, M., 2015. Easymkl: a scalable multiple kernel learning algorithm. *Neurocomputing* 169, 215–224.
- Bach, F.R., Lanckriet, G.R., Jordan, M.I., 2004. Multiple kernel learning, conic duality, and the smo algorithm, in: *Proceedings of the twenty-first international conference on Machine learning*, ACM. p. 6.
- Bucak, S.S., Jin, R., Jain, A.K., 2014. Multiple kernel learning for visual object recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 1354–1369.
- Chatfield, K., Lempitsky, V.S., Vedaldi, A., Zisserman, A., 2011. The devil is in the details: an evaluation of recent feature encoding methods., in: *BMVC*, p. 8.
- Chen, L., Duan, L., Xu, D., 2013. Event recognition in videos by learning from heterogeneous web sources, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2666–2673.
- Chéron, G., Laptev, I., Schmid, C., 2015. P-cnn: Pose-based cnn features for action recognition, in: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3218–3226.
- Dalal, N., Triggs, B., Schmid, C., 2006. Human detection using oriented histograms of flow and appearance, in: *Computer Vision–ECCV 2006*. Springer, pp. 428–441.
- Du, Y., Wang, W., Wang, L., 2015. Hierarchical recurrent neural network for skeleton based action recognition, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1110–1118.
- Fernando, B., Gavves, E., Oramas, J., Ghodrati, A., Tuytelaars, T., 2016. Rank pooling for action recognition. *IEEE transactions on pattern analysis and machine intelligence*.
- Fernando, B., Gavves, E., Oramas, J.M., Ghodrati, A., Tuytelaars, T., 2015. Modeling video evolution for action recognition, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5378–5387.
- Gaidon, A., Harchaoui, Z., Schmid, C., 2012. Recognizing activities with cluster-trees of tracklets, in: *BMVC 2012-British Machine Vision Conference*, BMVA Press. pp. 30–1.
- Gkioxari, G., Malik, J., 2015. Finding action tubes, in: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Husain, F., Dellen, B., Torras, C., 2016. Action recognition based on efficient deep feature learning in the spatio-temporal domain. *IEEE Robotics and Automation Letters* 1, 984–991.
- Izadinia, H., Shah, M., 2012. Recognizing complex events using large margin joint low-level event model, in: *European Conference on Computer Vision*, Springer. pp. 430–444.
- Jain, M., van Gemert, J.C., Snoek, C.G., 2015. What do 15,000 object categories tell us about classifying and localizing actions?, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 46–55.
- Jain, M., Jegou, H., Bouthemy, P., 2013. Better exploiting motion for better action recognition, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2555–2562.
- Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L., 2014. Large-scale video classification with convolutional neural networks, in: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725–1732.
- Kuehne, H., Huang, H., Garrote, E., Poggio, T., Serre, T., 2011. Hmdb: a large video database for human motion recognition, in: *IEEE International Conference on Computer Vision (ICCV)*, IEEE. pp. 2556–2563.
- Laptev, I., 2005. On space-time interest points. *International journal of computer vision* 64, 107–123.
- Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B., 2008. Learning realistic human actions from movies, in: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, IEEE. pp. 1–8.
- Le, Q.V., Zou, W.Y., Yeung, S.Y., Ng, A.Y., 2011. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis, in: *Computer Vision and Pattern Recognition (CVPR)*, 2011 IEEE Conference on, IEEE. pp. 3361–3368.
- Marszalek, M., Laptev, I., Schmid, C., 2009. Actions in context, in: *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE. pp. 2929–2936.
- Oneata, D., Verbeek, J., Schmid, C., 2013. Action and event recognition with fisher vectors on a compact feature set, in: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1817–1824.
- Perronnin, F., Sánchez, J., Mensink, T., 2010. Improving the fisher kernel for large-scale image classification, in: *Computer Vision–ECCV 2010*. Springer, pp. 143–156.
- Ponce-López, V., Escalante, H.J., Escalera, S., Baró, X., 2015. Gesture and action recognition by evolved dynamic subgestures., in: *BMVC*, pp. 129–1.
- Revaud, J., Douze, M., Schmid, C., Jégou, H., 2013. Event retrieval in large video collections with circulant temporal encoding, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2459–2466.
- Saleh, A., Akram, M.A.G.F., Abdel-Nasser, M., Puig, D., . Exploiting the kinematic of the trajectories of the local descriptors to improve human action recognition .
- Sharma, S., Kiros, R., Salakhutdinov, R., 2015. Action recognition using visual attention. *arXiv preprint arXiv:1511.04119*.
- Simonyan, K., Zisserman, A., 2014. Two-stream convolutional networks for action recognition in videos, in: *Advances in neural information processing systems*, pp. 568–576.
- Sminchisescu, C., Kanaujia, A., Metaxas, D., 2006. Conditional models for contextual human motion recognition. *Computer Vision and Image Understanding* 104, 210–220.
- Song, Y., Morency, L.P., Davis, R., 2013. Action recognition by hierarchical sequence summarization, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3562–3569.
- Srivastava, N., Mansimov, E., Salakhutdinov, R., 2015. Unsupervised learning of video representations using lstms., in: *ICML*, pp. 843–852.
- Sun, C., Nevatia, R., 2013. Active: Activity concept transitions in video event classification, in: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 913–920.
- Tang, K., Fei-Fei, L., Koller, D., 2012. Learning latent temporal structure for complex event detection, in: *Computer Vision and Pattern Recognition (CVPR)*, 2012 IEEE Conference on, IEEE. pp. 1250–1257.
- Taylor, G.W., Fergus, R., LeCun, Y., Bregler, C., 2010. Convolutional learning of spatio-temporal features, in: *European conference on computer vision*, Springer. pp. 140–153.
- Wang, H., Kläser, A., Schmid, C., Liu, C.L., 2013. Dense trajectories and

- motion boundary descriptors for action recognition. *International journal of computer vision* 103, 60–79.
- Wang, H., Schmid, C., 2013. Action recognition with improved trajectories, in: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3551–3558.
- Wang, H., Ullah, M.M., Klaser, A., Laptev, I., Schmid, C., 2009. Evaluation of local spatio-temporal features for action recognition, in: *British Machine Vision Conference (BMVC)*, BMVA Press. pp. 124–1.
- Wang, Y., Mori, G., 2011. Hidden part models for human action recognition: Probabilistic versus max margin. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1310–1323.
- Wang, Y., Song, J., Wang, L., Van Gool, L., Hilliges, O., 2016. Two-stream sr-cnns for action recognition in videos, *BMVC*.
- Wu, D., Shao, L., 2014. Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 724–731.
- Yamato, J., Ohya, J., Ishii, K., 1992. Recognizing human action in time-sequential images using hidden markov model, in: *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on, IEEE*. pp. 379–385.
- Yu, S., Falck, T., Daemen, A., Tranchevent, L.C., Suykens, J.A., De Moor, B., Moreau, Y., 2010. L 2-norm multiple kernel learning and its application to biomedical data fusion. *BMC bioinformatics* 11, 309.
- Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G., 2015. Beyond short snippets: Deep networks for video classification, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4694–4702.