Rank Pooling for Action Recognition

Basura Fernando, Efstratios Gavves, José Oramas M., Amir Ghodrati and Tinne Tuytelaars

Abstract—We propose a function-based temporal pooling method that captures the latent structure of the video sequence data - e.g. how frame-level features evolve over time in a video. We show how the parameters of a function that has been fit to the video data can serve as a robust new video representation. As a specific example, we learn a pooling function via ranking machines. By learning to rank the frame-level features of a video in chronological order, we obtain a new representation that captures the video-wide temporal dynamics of a video, suitable for action recognition. Other than ranking functions, we explore different parametric models that could also explain the temporal changes in videos. The proposed functional pooling methods, and rank pooling in particular, is easy to interpret and implement, fast to compute and effective in recognizing a wide variety of actions. We evaluate our method on various benchmarks for generic action, fine-grained action and gesture recognition. Results show that rank pooling brings an absolute improvement of 7-10 average pooling baseline. At the same time, rank pooling is compatible with and complementary to several appearance and local motion based methods and features, such as improved trajectories and deep learning features.

Index Terms—action recognition, temporal encoding, temporal pooling, rank pooling, video dynamics

1 Introduction

A recent statistical study has revealed more than 300 hours of video content are added to YouTube every minute [1]. Moreover, a recent survey on network cameras has indicated that a staggering 28 million network cameras will be sold in 2017 alone [58]. Given the steep growth in video content all over the world, the capability of modern computers to process video data and extract information from them remains a huge challenge. As such, human action and activity recognition in realistic videos is of great relevance.

Most of the progress in the field of action recognition over the last decade has been associated with either of the following two developments. The first development has been the *local spatio-temporal descriptors*, including spatio-temporal [30] and densely sampled [6], [32] interest points, dense trajectories [65], and motion-based gradient descriptors [22]. The second development has been the adoption of *powerful encoding schemes* with an already proven track record in object recognition, such as Fisher Vectors [66]. Despite the increased interest in action [6], [22], [26], [30], [32], [52], [65] and event [21], [40], [47], [61] recognition, however, relatively few works have dealt with the problem of modeling the temporal information within a video.

Modeling the *video-wide temporal evolution* of appearance in videos is a challenging task, due to the large variability and complexity of video data. Not only actions are performed at largely varying speeds for different videos, but often the speed of the action also varies non-linearly even within a single video. Hence, while methods have been proposed to model the *video-wide temporal evolution* in actions (*e.g.* using HMM [67], [68], CRF-based methods [56] or deep networks [62]), the impact of these on action recognition performance so far has been somewhat disappointing. What is more, simple but robust techniques such as temporal pyramids that are similar to spatially dividing images [33] and objects [16] are insufficient. Nevertheless, it is clear that many actions and activities have a characteristic temporal ordering. See for instance the "moving out of the house" action in Figure 1. Intuitively, one would expect that a video representation that encodes this temporal change of appearances should help to better distinguish between different actions. Obtaining a good video-wide representation from a video still remains a challenge.

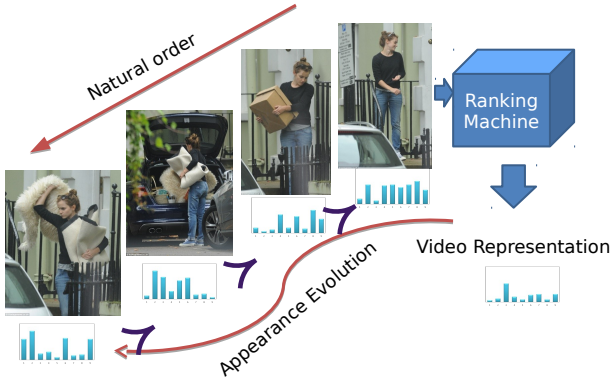


Fig. 1: Illustration of how rank pooling works. In this video, as Emma *moved out from the house*, the appearance of the frames evolves with time. A ranking machine learns this evolution of the appearance over time and returns a ranking function. We use the parameters of this ranking function as a new video representation which captures vital information about the action.

In this paper, we propose a new video representation that captures this video-wide temporal evolution. We start from the observation that, even if the execution time of actions varies greatly, the *temporal ordering is typically preserved*. We propose to capture the temporal ordering *of a particular video* by training a linear ranking machine on the frames of that video. More precisely, given all the frames of the video, we learn how to arrange them in chronological order, based on the content of the frames.

- *B. Fernando is with The Australia National University and was a PhD candidate at KU Leuven, ESAT-PSI, iMinds, Belgium. E-mail: see http://users.cecs.anu.edu.au/ basura/*
- *E. Gavves is with QUVA Lab, University of Amsterdam, Netherlands and was with KU Leuven, ESAT-PSI, iMinds, Belgium. E-mail: see http://www.egavves.com/*
- *J. Oramas, A. Ghodrati and T. Tuytelaars are with KU Leuven, ESAT-PSI, iMinds, Belgium. E-mail: firstname.lastname@esat.kuleuven.be*

Manuscript received November 30, 2015

The parameters of the linear ranking function encodes the video-wide temporal evolution of appearance of that video in a principled way. To learn such ranking machines, we use the supervised learning to rank framework [36]. Ranking machines trained on different videos of the same action can be expected to have similar ranking functions. Therefore, we propose to use the parameters of the ranking machine as a new video representation for action recognition. Classifiers trained on this new representation turn out to be remarkably good at distinguishing actions. Since the ranking machines act on frame content, they actually capture both the appearance and their evolution over time. We call our method ***rank pooling***.

The key contribution of rank pooling is to use the parameters of the ranking functions as a new video representation that captures the *video-wide temporal evolution of the video*. Our new video representation is based on a principled learning approach, it is efficient and easy to implement. Last but not least, with the new representation we obtain state-of-the art results in action and gesture recognition. The proposed use of parameters of functions as a new representation is by no means restricted to action recognition or ranking functions. Other than ranking functions, we explore different parametric models, resulting in a whole family of pooling operations.

The rest of the paper is organized as follows: in Section 2 we position our work w.r.t. existing work. Sections 3 and 4 describe our method, while Section 5 provides an insight from its application for action classification. This is followed by the evaluation of our method in Section 6. We conclude this paper in Section 7.

2 Related work

2.1 Action recognition

Capturing temporal information of videos for action recognition has been a well studied research domain. Significant improvements have been witnessed in modeling local motion patterns present in short frame sequences [30], [65], [66]. Jain *et al*. [23] proposed to first localize the actions in the video and exploit them for refining recognition.

To avoid using hand-engineered features, deep learning methodologies [34], [62] have also been investigated. Dynamics in deep networks can be captured either by extending the connectivity of the network architecture in time [26] or by using stacked optical flow instead of frames as input for the network [52]. The two stream stacked convolutional independent subspace analysis method, referred to as ConvISA [29], is a neural network architecture that learns both visual appearance and motion information in an unsupervised fashion on video volumes. A human pose driven CNN feature extraction pipeline is presented in [5]. In [5], authors represent body regions of humans with motion-based and appearance-based CNN descriptors. Such descriptors are extracted at each frame and then aggregated over time to form a video descriptor. To capture temporal information, authors consider temporal differences of frames and then concatenate the difference of vectors. Two convolutional neural networks are used to capture both appearance-based and motion-based features in *action tubes* [18]. In this method [18], the first spatial-CNN network takes RGB frames as input and captures the appearance of the actor as well as other visual cues from the scene. The second network, referred as the motion-CNN, operates on the optical flow signal and captures the movement of the actor. Spatio-temporal features are extracted by combining the output from the intermediate layers of the two networks. The benefits of having objects in the video representation for action classification is presented in [24].

Although the aforementioned methods successfully capture the local changes within small time windows, they are not designed to model the higher level motion patterns and video-wide appearance and motion evolution associated with certain actions.

2.2 Temporal and sequential modeling

State-space models such as generative Hidden Markov Models (HMMs) or discriminative Conditional Random Fields (CRFs) have been proposed to model dynamics of videos since the early days [54], [71]. Generative methods such as HMMs usually learn a joint distribution over both observations and action labels. In these early works, most often, the observations consist of visual appearance or local motion feature vectors obtained from videos. This results in HMMs that learn the appearance or the motion evolution of a specific action class. Then the challenge is to learn all variations of a single action class. Given the complexity, variability and the subtle differences between action classes, these methods may require a lot of training samples to learn meaningful joint probability distributions.

Discriminative CRF methods learn to discriminate two action classes by modeling conditional distribution over class labels. However, similar to HMMs, CRFs may also require a large amount of training samples to estimate all parameters of the models. In contrast, our proposed method does not rely on class labels to encapsulate temporal information of a video sequence. The proposed method captures video specific dynamic information and relies on standard discriminative methods such as SVM to discriminate action classes.

More recently, new machine learning approaches based on CRF, HMM and action grammars, have been researched for action recognition [46], [50], [56], [61], [67] by modeling higher level motion patterns. In [67], a part-based approach is combined with large-scale template features to obtain a discriminative model based on max-margin hidden conditional random fields. In [56], Song *et al*. rely on a series of complex heuristics and define a feature function for the proposed CRF model. In [61] Tang *et al*. propose a max-margin method for modeling the temporal structure in a video. They use a HMM model to capture the transitions of action appearances and duration of actions.

Temporal ordering models have also been applied in the context of complex activity recognition [21], [45], [49], [59]. They mainly focus on inferring composite activities from pre-defined, semantically meaningful, basic-level action detectors. In [59], a representation for events is presented that encodes statistical information of the atomic action transition probabilities using a HMM model. In [45], a set of shared spatio-temporal primitives, subgestures, are detected using genetic algorithms. Then, the dynamics of the actions of interest are modeled using the detected primitives and either HMMs or Dynamic Time Warping (DTW). Similar to the above works, we exploit the temporal structure of videos but in contrast, we rely on ranking functions to capture the evolution of appearance or local motion. Using the learning-to-rank paradigm, we learn a functional representation for each video.

Due to the large variability of motion patterns in a video, usually latent sequential models are not efficient. To cope with this problem, representations in the form of temporal pyramids

[15], [32] or sequences of histograms of visual features [14] are introduced. A method that aims at comparing two sequences of frames in the frequency domain using fast Fourier analysis called circulant temporal aggregation is presented in [47] for event retrieval. Different from the above, we explicitly model video-wide, video level dynamics using a principled learning paradigm. Moreover, contrary to [14], our representation does not require manually annotated atomic action units during training.

Recurrent neural networks have also been extensively studied in the context of sequence generation and sequence classification [20], [60]. In [57] the state of the LSTM encoder after observing the last input frame is used as a video representation [57]. A hierarchical recurrent neural network for skeleton based action recognition is presented in [8]. An LSTM model that uses CNN features for action recognition is presented in [73]. Typically, recurrent neural networks are trained in a probabilistic manner to maximize the likelihood of generating the next element of the sequence. They are conditional loglinear models. In contrast, the proposed rank pooling uses a support vector based approach to model the elements in the sequence. Rank pooling uses empirical risk minimization to model the evolution of the sequence data. Furthermore, in comparison to RNN-LSTM-based methods, Rank pooling is efficient both during training and testing, and effective even for high dimensional input data.

2.3 Functional representations

Our work has some conceptual similarity to the functional representations used in geometric modeling [41], which are used for solid and volume modeling in computer graphics. In this case an object is considered as a point set in a multidimensional space, and is defined by a single continuous real-valued function of point coordinates of the nature $f(x_1, x_2, ..., x_n)$ which is evaluated at the given point by a procedure traversing a tree structure with primitives in the leaves and operations in the nodes of the tree. The points with $f(x_1, x_2, ..., x_n) \geq 0$ belong to the object, and the points with $f(x_1, x_2, ..., x_n) < 0$ are outside of the object. The point set with $f(x_1, x_2, ..., x_n) = 0$ is called an isosurface. Similarly, in our approach the ranking function has to satisfy chronological order constraints on frame feature vectors and we use the ranking function as a representation of that video.

Since we use the parameters of a linear function as a new representation of a particular sequence, our work also bears some similarity to the exemplar SVM concept [37], [74]. Differently, our objective is to learn a representation for the relative ordering of a set of frames in a video. At the same time we do not need to rely on negative data to learn the representation, as is the case for exemplar SVM.

Meta-representation has the ability to represent a higher-order representation with a lower-order representation embedding. It is the capacity to represent a representation. Our rank pooling representation can also be considered as a meta-representation. The parameters of the ranking function in fact represent a lower dimensional embedding of chronological structure of the frames. In the learning to rank paradigm, these ranking functions are trained to order data. Our hypothesis is that this parametric embedding of sequence data can be used to represent their dynamics.

This paper extends the work of [12]. Compared to the conference version, this paper gives a more precise account of the internals of rank pooling. First, we provide an extended discussion of related work, covering better the recent literature. From a technical point of view, we generalize the concept of rank pooling to a framework that uses functional parameters as a new video representation. We hypothesize that any stable and robust parametric functional mapping that maps frame data to the time variable can be used for modeling the video dynamics. Furthermore, we analyze the types of non-linear kernels that best capture video evolution. We provide some empirical evidence to demonstrate the capabilities of rank pooling. Finally, we combine rank pooling with convolutional neural network features to further boost the state-of-the-art action recognition performance.

Recently, Fernando *et al* extend rank pooling to encode higher order dynamics of a video sequence in a hierarchical manner in [10] and in [3] Bilen *et al* introduced dynamic image networks which allows us to learn dynamic representation using CNNs and rank pooling.

3 Video Representations

In this section we present our *temporal pooling* method, which encodes dynamics of video sequences and, more specifically captures the video-wide temporal evolution (VTE) of the appearance in videos. First, we present the main idea in Section 3.1 where we propose to use parameters of suitable functions to encode the dynamics of a sequence. Then, in Section 3.2 we present how to formulate these specific functions using rankers. Next, in Section 3.3 we analyse the generalization capacity of the proposed rank pooling. Finally, in Section 3.4 we describe how to use functional parameters of other parametric models as a temporal representation and compare traditional temporal pooling methods with rank pooling.

3.1 Functional parameters as temporal representations

We assume that each frame of a given video is represented by a vector $\mathbf{x}$. Then the video composed of n frames is a sequence of vectors, $X = [\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}]$. A frame at discrete time step t is denoted by a vector $\mathbf{x_t} \in \mathbb{R}^D$. Given this sequence of vectors, we first smooth the sequence X to a more general form to obtain a new sequence $V = [\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}]$. We discuss how to obtain smoothed sequences in Section 4. For the rest of the analysis we use smoothed sequences V, unless otherwise specified. Last, we use the notation $\mathbf{x}_{1:t}$ or $\mathbf{v}_{1:t}$ to denote a sub-sequence from time step 1 to t.

Our goal is to encode the temporal evolution of appearances, or, in other words the dynamics $\mathcal{D}$ of the sequence V. At an abstract level, dynamics $\mathcal{D}$ reflect the way the vector valued input changes from time t to $t+1$ for all t. Assuming that the sequence V is sufficiently smooth, we can encode the dynamics of V using a linear function $\Psi_u = \Psi(V; \mathbf{u})$ parametrized by $\mathbf{u}$, such that Ψ approximates $\mathcal{D}$, namely

$$\arg\min_{u} ||\mathcal{D} - \Psi_u||. \tag{1}$$

For a given definition of dynamics $\mathcal{D}$ (see below), there exists a family of functions Ψ. Different videos from the same action category will have different (yet similar) appearances and will be characterized by different (yet similar) appearance dynamics. For each video $V_i(\cdot)$, we learn a different dynamics function $\Psi_i(\cdot; \mathbf{u_i})$ parametrized by $\mathbf{u_i}$. Given stability and robustness guarantees of

the family of functions Ψ, different videos from the same action category then result in similar dynamics functions $\Psi_i(\cdot; \mathbf{u_i})$.

As the family of functions Ψ for modeling the dynamics is the same for all videos, what characterizes the dynamics of each specific video is the parametrization $\mathbf{u_i}$. We propose to use the parameters $\mathbf{u_i} \in \mathbb{R}^D$ of Ψ_i as a new video representation, capturing the specific *appearance evolution of the video*. Thus we obtain a functional representation, where the functional parameters $\mathbf{u_i}$ serve as the representation, capturing a vital part of the video-wide temporal information.

As a first concrete case, in the next section, we present how we could learn such functional representations using the learning-to-rank paradigm.

3.2 Rank pooling

One way to understand dynamics $\mathcal{D}$ is to consider them as the driving force for placing frames in the correct order. Indeed, in spite of the large variability in speed, between different videos and even within a single video, the relative ordering is relatively preserved. To capture such dynamics for video sequence V_i, we consider the learning-to-rank [36] paradigm, which optimizes ranking functions of the form $\Psi(t, \mathbf{v_{1:t}}; \mathbf{u})$. We can either employ a point-wise [55], a pair-wise [36] or a sequence-based ranking machine [11]. Then, we can use the parameters of these ranking machines as our new video representation in a process that we coin *rank pooling*.

Videos are ordered sequences of frames, where the frame order also dictates the evolution of the frame appearances. We focus on the relative orderings of the frames. If $\mathbf{v_{t+1}}$ succeeds $\mathbf{v_t}$ we have an ordering denoted by $\mathbf{v_{t+1}} \succ \mathbf{v_t}$. As such, we end up with order constraints $\mathbf{v_n} \succ \ldots \succ \mathbf{v_t} \succ \ldots \succ \mathbf{v_1}$. We exploit the transitivity property of video frames to formulate the objective as a pairwise learning-to-rank problem *i.e.* (if $\mathbf{v_a} \succ \mathbf{v_b}$ and $\mathbf{v_b} \succ \mathbf{v_c} \implies \mathbf{v_a} \succ \mathbf{v_c}$).

To model the video dynamics with pair-wise rank-pooling, we solve a constrained minimization *pairwise-learning-to-rank* [36] formulation, such that it satisfies the frame order constraints. Pair-wise linear ranking machines learn a linear function $\psi(\mathbf{v}; \mathbf{u}) = \mathbf{u}^T \cdot \mathbf{v}$ with parameters $\mathbf{u} \in \mathbb{R}^D$. The ranking score of $\mathbf{v_t}$ is obtained by $\psi(\mathbf{v_t}; \mathbf{u}) = \mathbf{u}^T \cdot \mathbf{v_t}$ and satisfies the pairwise constraints $(\mathbf{v_{t+1}} \succ \mathbf{v_t})$ by a large margin, while avoiding over-fitting. As a result we aim to learn a parametric vector $\mathbf{u}$ such that it satisfy all constraints $\forall t_i, t_j$, $\mathbf{v_{t_i}} \succ \mathbf{v_{t_j}} \iff \mathbf{u}^T \cdot \mathbf{v_{t_i}} > \mathbf{u}^T \cdot \mathbf{v_{t_j}}$.

Using the structural risk minimization and max-margin framework, the constrained learning-to-rank objective is

$$\begin{aligned} \arg\min_{\mathbf{u}} \ & \frac{1}{2}\|\mathbf{u}\|^2 + C \sum_{\forall i,j \mathbf{v_{t_i}} \succ \mathbf{v_{t_j}}} \epsilon_{ij} \\ s.t.\ & \mathbf{u}^T \cdot (\mathbf{v_{t_i}} - \mathbf{v_{t_j}}) \geq 1 - \epsilon_{ij} \\ & \epsilon_{ij} \geq 0. \end{aligned} \tag{2}$$

As the parameters $\mathbf{u}$ define the frame order of frames $\mathbf{v_t}$, they represent how the frames evolve with regard to the appearance of the video. Hence, the appearance evolution is encoded in the parameter $\mathbf{u}$. The above optimization objective is expressed on the basis of RankSVM [25], however, any other linear learning-to-rank method can be employed. For example, in point-wise rank pooling we seek a direct mapping from the input time dependent vectors v_t to the time variable t based on the linear parameters $\mathbf{u}$. Namely, we have that

$$\begin{aligned} g(\mathbf{v_t}; \mathbf{u}) &\mapsto t \\ \mathbf{u}^* = \arg\min_{\mathbf{u}} \sum_t & |t - \mathbf{u^T} \cdot \mathbf{v_t}|. \end{aligned} \tag{3}$$

The support vector regression (SVR) [55] formulation is a robust extension of equation 3 and thus, one can use SVR parameters to encode the dynamics. Support vector regression is known to be a point-wise ranking formulation [36]. The solution of SVR would also satisfy the order constraints $g(\mathbf{v_q}; \mathbf{u}) > g(\mathbf{v_j}; \mathbf{u})$ if $\mathbf{v_q} \succ \mathbf{v_j}$ because of the direct mapping of the form $g(\mathbf{v_t}; \mathbf{u}) \mapsto t$.

In summary, to represent dynamics $\mathcal{D}$ of a video V using rank pooling, we use the parameter vector $\mathbf{u}$ as a video representation. The vector $\mathbf{u}$ is a temporal encoding of the input vector sequence $\mathbf{v_n} \succ \ldots \succ \mathbf{v_t} \succ \ldots \succ \mathbf{v_1}$. The video representation $\mathbf{u}$ can be learnt either using a pair-wise ranking machine as in equation 2 or using the direct mapping as in equation 3, *i.e.* SVR [55] (our default setting). Modeling the temporal evolution via rankers displays several advantages. First, in videos in the wild we typically observe a large variability in speed at which actions are performed. This is not an issue for ranker functions that are oblivious to the pace at which the frames appear and only focus on their accurate relative ordering. Second, a powerful advantage of linear ranking machines is that their function parameters reside in the same space as the input sequence data V.

3.3 Generalization capacity

As explained above, we use the parameters of learnt ranking functions to model the temporal dynamics of the specific video. All functions from all videos will belong to the same parametric family of models. However, as the different videos will differ in appearance and their dynamics, each function will be characterized by a different set of parameters. It remains to be answered whether different videos that contain the same action category will be characterized by similar parameters or not.

For action recognition we make the basic assumption that similar actions in different videos will have similar dynamics ($\mathcal{D}$). Namely, we assume there is a theoretical probability density function $p_{\mathcal{D}}$ based on which different instances of video-wide temporal evolutions are sampled for an action type. Naturally, different videos of the same action will be different and generate different ranking functions, so each linear ranker will have a different parametric representation vector ψ. Therefore, a rightful question is to what extent learning the ψ per video generalizes well for different videos of the same action.

As we cannot know the theoretical probability density function $p_{\mathcal{D}}$ of dynamics in real world videos, it is not possible to derive a strict bound on the generalization capacity of the functional parameters u_i. However, the sensitivity risk minimization framework gives us a hint of this generalization capacity of $\mathbf{u_i}$ when the input for the training is slightly perturbed. More specifically, Bousquet *et al.* [4] showed on a wide range of learning problems, *e.g.* SVM, SVR and RankSVM, that the difference of the generalization risk R from the leave one out error R_i in the training set is bounded by

$$|R - R^{/i}| \leq E_r[|l(A_S, r) - l(A_{S^{/i}}, r)|] \leq \beta, \tag{4}$$

where A_S is a ranking algorithm with uniform stability β learned from the set of samples S. The expectation of the loss over the distribution r is denoted by $E_r[l]$ where l is a bounded loss

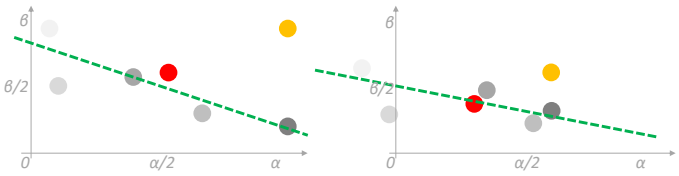


Fig. 2: Various pooling operations given data plotted on a 2d feature space (gray circles stands for data, red circles for average pooling and yellow circles for max pooling, whereas the green dashed lines stand for rank pooling). The green dashed hyperplanes returned by our rank pooling not only describe nicely the latent data structure, but also are little affected by the random data noise. In contrast the average and max pooling represenations are notably disturbed. In fact, max pooling even creates "ghost" circles in areas of the feature space where no data exist.

function such that $0 \leq l(A_S, r) \leq M$; (M is a sufficiently small number).

Given a certain video, eq. (4) implies that a slight change (ignoring smoothing of sequences) during training will learn a ranking function $\psi_{/i}$ with an error no larger than β compared to the ψ learned when all frames are available. Although eq. (4) does not give a strict answer for what happens when the training input changes significantly from video to video, it hints that since the temporal evolution of similar actions should be similar, this should also be the case for the learned ranking functions of rank pooling denoted by $\mathbf{u}$. This generalization capacity of rank pooling is furthermore supported by our experimental validation.

3.4 Functional parameters as temporal pooling

In the above we described how to encode temporal information from a video sequence using ranking machines. The parameters $\mathbf{u}$ that we learn either from a pair-wise ranking machine or a point-wise ranking machine can be viewed as a principled, data-driven, temporal pooling method, as they summarize the data distributions over a whole sequence. The use of ranking functional parameters as temporal pooling contrasts with other standard methods of pooling, such as *max* pooling or *sum* pooling, which are typically used either in convolutional neural networks [27] or for aggregating Fisher vectors [43].

First, as rank pooling is regularized, it is much less susceptible to the local noise in the observations *i.e.* robust. See for example the left picture in Fig. 2, where max pooling is notably affected by stochastic perturbations in the feature space of the sequence data. Second, given some latent structure, temporal structure in our case, rank pooling fits the data trend by minimizing the respective loss function. Max pooling and sum pooling, on the other hand, are operators that do not relate to the underlying temporal data distribution. As such, max and sum pooling might aggregate the data by creating artificial, *ghost* samples, as shown in the right picture of Fig. 2. In contrast, rank pooling transits the problem to a dual parameter space, in which the aggregation point is the one that optimally represents the latent data structure, as best expressed by the respective parametric model.

Next, we extend further the idea of using functional parameters as representations with different parameteric models. Assume a function which learns a projection of the video frames into a subspace. Also, assume that we have enriched the frame representations so that they are more correlated with the time arrow, as we will discuss in Section 4. Then, another way to capture the video temporal evolution of appearances and the dynamics of the video would be to fit a function that reconstructs the time-sensitive appearance of all frames.

To reconstruct the time-sensitive appearance of all frames in a video sequence V, we need to fit a function $\Psi(t, \mathbf{v_{1:t}}; \mathbf{u})$, such that

$$u^* = \arg\min_u \|V - uu^T V\|^2, \tag{5}$$

where $\mathbf{u} \in \mathbb{R}^{D \times d}$, where d is the new subspace dimensionality. In equation (5) we minimize the reconstruction error after a linear projection. One can solve the above minimization using principal component analysis, namely by singular value decomposition

$$V = U\Sigma U'^T \tag{6}$$

The singular value decomposition returns two orthonormal matrices $U \in \mathbb{R}^{D \times D}, U' \in \mathbb{R}^{T \times T}$, who contain the eigenvectors of the covariance matrices $\hat{C} = E(VV^T)$ and $\hat{C}' = E(V^T V)$ respectively, where T is the number of frames in the video sequence $V \in \mathbb{R}^{D \times T}$.

The straightforward way of defining the subspace u is by selecting the k first eigenvectors from U. However, more often than not the number of frames is smaller than the dimensionality of the frame features, $T < D$. Hence, the matrix $\hat{C}$, which is the expected value of the real but unknown covariance matrix C, is an unreliable estimate. To obtain a more robust subspace projection, we can instead consider

$$u = U'(V^T)^{-1} \tag{7}$$

Since U' is obtained from the more robust estimate $\hat{C}'$, the subspace projection u from eq. 7 is a more reliable representation of the temporal evolution of the appearances in V. We can therefore use u from eq. 7 to represent the video V. Naturally, one can maintain only the first d principal components of U' to control the final dimensionality of u.

Given that frame features should ideally be correlated with the time variable, the first principal eigenvector contains the highest variance of the video appearance as it evolves with time. Therefore, one can use the first principal component of a video as a temporal representation given that video frames are pre-processed to indirectly correlate with time (see Section 4). We refer to the above functional parameter pooling as ***subspace pooling***. Subspace pooling is robust, as also shown in [13]. Moreover, the subspace pooling has a close relationship to dynamic texture [7] which uses auto-regressive moving average process which estimates the parameters of the model using sequence data. Subspace pooling is also related to dynamic subspace angles [35] which compares videos by computing subspaces and then measuring the principal angle between them.

Support vector ranking machines and principal component decomposition are robust models which we can use for temporal pooling. However, other learning algorithms can also be considered to be used as functional parameter representation. As a case study, in this paper we use two more popular choices: *(a)* Hidden Markov Models (HMMs), and *(b)* Neural Networks (NN). For details, we refer to the experimental results section. We explore the different possibilities experimentally.

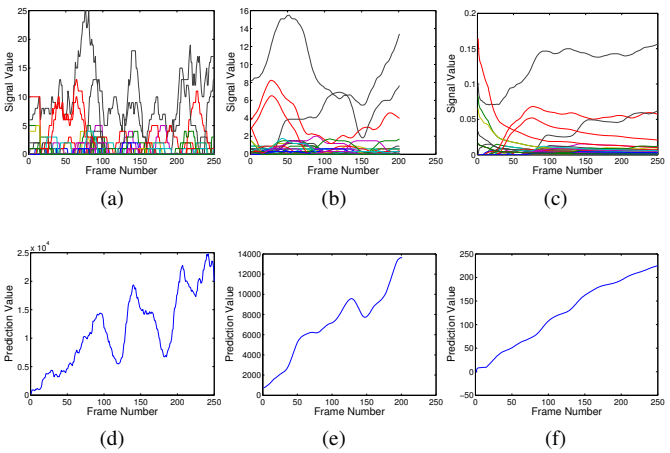


Fig. 3: Using ranking machines for modeling the video temporal evolution of appearances, or alternatively, the video dynamics. We see in (a) the original signal of independent frame representation, (b) the signal obtained by moving average, (c) the signal obtained by time varying mean vector (different colors refer to different dimensions in the signal $\mathbf{v_t}$). In (d), (e) and (f) we plot the predicted ranking score of each frame obtained from signal (a), (b) and (c) respectively after applying the ranking function (predicted ranking value at t, $\mathbf{s_t} = \mathbf{u}^T \cdot \mathbf{v_t}$).

4 Frame Representations

Even in a noise-free world video data would still exhibit high degrees of variability. To reduce the effect of noise and violent abrupt variations, we smooth the original video signal (*i.e.* the frame representation $\mathbf{x_t}$). In this section we discuss three methods to obtain smoothed robust signals $\mathbf{v_t}$ from frame data $\mathbf{x_t}$.

4.1 Independent Frame Representation

The most straightforward representation for capturing the evolution of the appearance of a video is to use independent frames $\mathbf{v_t} = \frac{\mathbf{x_t}}{\|\mathbf{x_t}\|}$. This approach has two disadvantages. First, the original signal can vary significantly, see Figure 3(a), often leading the ranking machines to focus on undesirable temporal patterns. At the same time independent frames might generate ranking functions with high ranking errors during training time. Second, independent frame representations are characterized by a weak connection between $\mathbf{v_t}$ and t. Given this weak correlation between the $\mathbf{v_t}$ and time t, see Figure. 3 (a), the ranking function may not learn the appearance evolution over time properly. As a result, plotting the predicted score $\mathbf{s_t} = \mathbf{u_i}^T \cdot \mathbf{v_t}$ for each of the frames in the video is not as smooth as one would desire (see Figure 3 (d)).

4.2 Moving Average (MA)

Inspired by the time series analysis literature, we consider the moving average with a window size T as video representation at time t. In other words we consider locally smoothed signals. For MA, we observe two facts. First, the output signal is much smoother, see Figure 3(b). Second, $\mathbf{v_t}$ maintains a temporally local dependency on the surrounding frames around t, namely the frames $[t, t+T]$. Unlike the independent frames representation, however, the moving average model forges a connection between $\mathbf{v_t}$ and t. Plotting these two variables for a window T=50 in Figure 3(b), we observe a smoother relation between the dimensions of $\mathbf{v_t}$ and the frame number which equals to the time variable. As such, the video-wide temporal information is captured well in the predicted score s_t, see Figure 3(e).

Although the moving average representation allows for capturing the appearance evolution of a video better, we still witness a general instability in the signals. Furthermore, we note that the moving average representation introduces undesirable artifacts. For one, window size T has to be chosen, which is not always straightforward as actions often take place in different tempos. Moreover, due to boundary effects, v_t is undefined for the last time stamps t of the video.

4.3 Time Varying Mean Vectors

To deal with the limitations of the independent frames representation and the moving average, we propose a third option, the *time varying mean vectors*.

Let us denote the mean at time t as $\mathbf{m_t} = \frac{1}{t} \times \sum_{\tau=1}^{t} \mathbf{x}_\tau$. Then, $\mathbf{v_t}$ captures only the direction of the unit mean appearance vector at time t, *i.e.* ($\mathbf{v_t} = \frac{\mathbf{m_t}}{\|\mathbf{m_t}\|}$). Thus the ranking function ψ learns the evolution of the normalized mean appearance at time t. We plot the relationship between $\mathbf{v_t}$ and t in Figure 3(c) and the prediction score s_t in Figure 3(f). We observe that, as desired, the output is smooth, almost resembling a monotonically increasing function. Different from the independent frames representation, the time varying mean vectors introduce a better dependency between the input v_t and the target t.

By construction *time varying mean vectors* capture only the temporal information from the forward flow of the video with respect to the time. This is because the video progresses from the past to the future frames. However, there is no reason why the mean vectors should not be considered also in the reverse order, starting from the future frames and traversing backwards to the past frames of a video. To this end we generate the exact same objective, as in eq. 2, playing the video in reverse order, however. We shall refer to appearance evolution captured by forward flow as *forward rank pooling* (**FDRP**), whereas reverse flow as *reverse rank pooling*.

4.4 Non-linear rank pooling

In section 3.2, we considered only linear machines to obtain rank pooling based video representations. To incorporate non-linearities we resort to non-linear feature maps [64] applied on each $\mathbf{v_t}$ of V, thus allowing for employing effective [17] linear ranking machines in their primal form.

A popular technique to include non-linearities is to pre-process and transform the input data by non-linear operations. Let us denote a point-wise non-linear operator $\Phi(\cdot)$ which operates on the input x so that the output $\Phi(x)$ is a non-linear mapping of x. We use such non-linear feature maps to model non-linear dynamics of input video data. Given the time varying mean vector $\mathbf{v_t}$, to obtain non-linear representation $\mathbf{u}$ of input video X, we map $\mathbf{v_t}$ to $\Phi(\mathbf{v_t})$ using the non-linear operation before learning the ranking machines. Next we describe an interesting non-linear feature map that is useful particularly for real data such as Fisher vectors. In our experiments we also demonstrate the advantage of capturing non-linear dynamics via non-linear feature maps which we coined non-linear rank pooling.

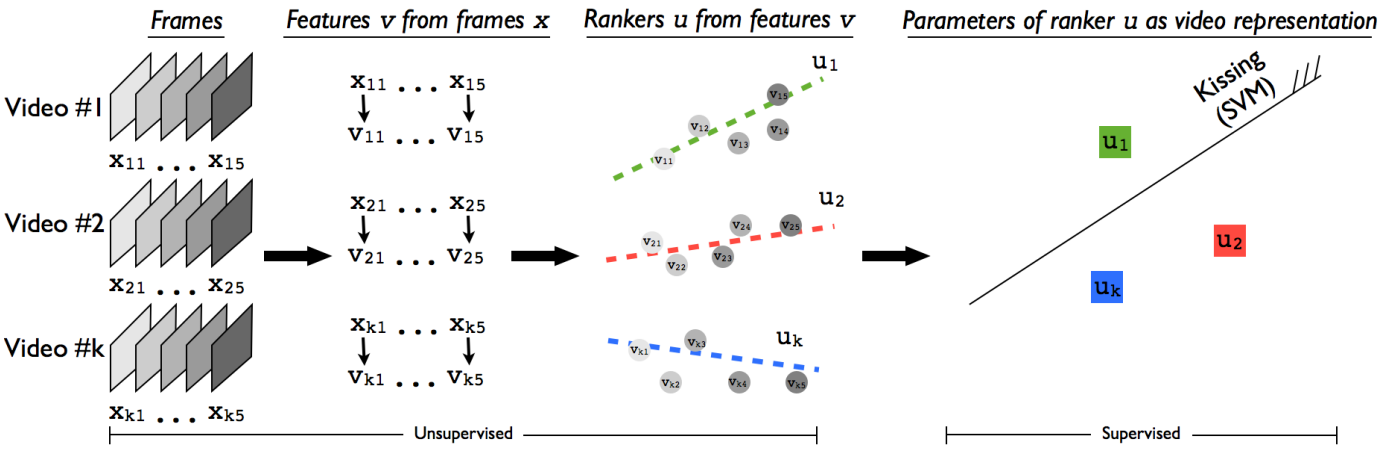


Fig. 4: Processing steps of rank pooling for action recognition. First, we extract frames $\mathbf{x_1} \dots \mathbf{x_n}$ from each video. Then we generate feature $\mathbf{v_t}$ for frame t by processing frames from $\mathbf{x_1}$ to $\mathbf{x_t}$ as explained in section 4. Afterwards, using ranking machines we learn the video representation $\mathbf{u}$ for each video. Finally, video specific $\mathbf{u}$ vectors are used as a representation for action classification.

A popular kernel in visual recognition tasks is the Hellinger kernel

$$K_{hell}(x, y) = \sqrt{x}^T \sqrt{y}. \tag{8}$$

The Hellinger kernel introduces non-linearities to the kernel machines, while maintaining separability, thus allowing for solving the optimizations in their primal form. The Hellinger kernel copes well with the frequently observed feature burstiness [2]. When eq. (8) is applied directly, then we obtain a complex kernel, as the negative features turn into complex numbers, namely we have that $\sqrt{x} = \sqrt{x^+} + i\sqrt{x^-} = \hat{x}^+ + i\hat{x}^-$, where $\hat{x}^+ = \sqrt{x^+}$ and $\hat{x}^- = \sqrt{x^-}$ refer to the positive and negative parts of the feature x, namely $x_i^+ = x_i, \forall x_i > 0$ and 0 otherwise, while $x_i^- = -x_i, \forall x_i < 0$ and 0 otherwise. Then the Hellinger kernel equals to

$$\begin{aligned} K_{hell}(x, y) &= (\hat{x}^+ + i\hat{x}^-)^T(\hat{y}^+ + i\hat{y}^-) \\ &= (\hat{x}^+\hat{y}^+ - \hat{x}^-\hat{y}^-) + i(\hat{x}^-\hat{y}^+ + \hat{y}^-\hat{x}^+) \end{aligned} \tag{9}$$

To avoid any complications with using complex numbers, we focus on the real part of K_{hell}. Using the real part of the Hellinger kernel, we effectively separate the positive and negative parts of the features, easily deriving that

$$\begin{aligned} K_{\mathrm{Re}\{hell\}} &= [\hat{x}^+, \hat{x}^-][\hat{y}^+, \hat{y}^-]^T \\ &= K_{hell}(x^*, y^*), \end{aligned} \tag{10}$$

where $x^* = [x^+, x^-]^T$ is the *expanded* feature, which is double in dimensionality compared to x and is composed of only positive elements. Comparing eq. (10) with eq. (8), we observe that we have practically doubled the dimensionality of our feature space, as all $x, \hat{x}^+, \hat{x}^-$ have the same dimensionality, allowing for more sophisticated learning. The first half of the feature space relates to the positive values only $\hat{x}^+$, while the second part relates to the negative ones $\hat{x}^-$. We refer to this feature map as *posneg* feature map and, to the respective kernel as *posneg* kernel. Unless stated otherwise, in the remainder of the text we use the posneg feature maps.

5 Overview

Next, we will briefly describe the pipeline for applying rank pooling for the task of action classification in videos.

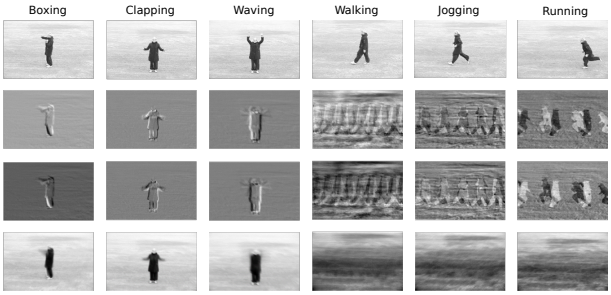


Fig. 5: Examples from the six action categories in the KTH action recognition dataset [31]. From left to right you see the actions *boxing*, *clapping*, *waving*, *walking*, *jogging* and *running*. From top to bottom you see an example frame from a random video, the forward rank pooling, the reverse rank pooling and the result after the standard mean pooling. The rank pooling as well as the mean representations are computed on the image pixels. We observe that the forward and reverse rank pooling indeed capture some of the crisp, temporal changes of the actions, whereas the mean representations lose the details.

5.1 Action classification from A to Z

The action classification pipeline is illustrated in detail in Figure 4. First, for each video X_i the video frames $x_{ij}, j = 1, \dots M$ are processed individually, so that frame feature encodings v_{ij} are extracted and their frame location in the video is recorded. A popular choice to date would be to first extract HOG, HOF, MBH, TRJ features computed on improved trajectories per frame together with the frame location, then compute the per frame Fisher vector or the Bag-of-Words feature encodings (first two columns in Figure 4). Next, given the smooth frame features obtained from time varying mean vectors or any of the other frame representations discussed in Section 4, we apply a parametric pooling step. For each of the videos X_i we fit one of the parametric models discussed in Section 3 (third column in Figure 4). We then use the parameters u_i of the parametric model as the video representation. Last, after having computed all u_i for every video X_i, we run a standard supervised classification method on our dataset denoted by $D_{train} = \{\mathbf{u_i}, y_i\}, i = 1, \dots, N$ where N is

the number of videos in our training set and y_i is the class label of the i^{th} video. We use non-linear SVM classifiers such as χ^2 feature maps [63] applied on feature vectors $\mathbf{u_i} \in \mathbb{R}^D$.

We summarize some of the advantages of using parameters of a function that is trained to map or correlate input data to time variable as a video representation. First, no supervised information is needed as video order constraints can be obtained directly from the sequence of video frames. Second, by minimizing eq. (1) rank pooling captures the evolution of appearance of a video in a principled manner, either by minimizing a ranking objective or by minimizing the reconstruction error of video appearances over time. Third, such a parametric representation does not require negative data to be added explicitly during the learning of the video representations. Fourth, since rank pooling encapsulates the changes that occur in a video, it captures useful information for action recognition.

5.2 Visualising dynamics of videos

In this section we demonstrate a visual inspection of what our rank pooling method learns. For simplicity of visualization we use sample video sequences from the KTH action recognition dataset [31]. As features we use the raw RGB values vectorized per frame as features. In this visualization experiment we do not extract any trajectories or other more sophisticated features and we use independent frame representations. We apply forward and reverse rank pooling on the video sequences of the first row. To obtain the visualization, given a frame image we first transform it to a D-dimensional gray-scaled vector. Then we apply the rank pooling method to obtain the parameters $\mathbf{u}$. Afterwards, we reshape the vector $\mathbf{u}$ to the original frame image size and project back each pixel value to be in the range of 0-255 using linear interpolation by min-max normalization.

We use example videos provided in the dataset which consists of six action classes, namely boxing, hand clapping, hand waving, walking, jogging and running. Samples from this dataset are shown in the first row of Figure 5. For each of the 6 actions in the KTH dataset we present a sample sequence in each column of Figure 5 (from left to right we have *boxing*, *clapping*, *waving*, *walking*, *jogging* and *running*). In the second and third row we show the forward and reverse rank pooling video representation respectively (namely the computed u_i), illustrating the captured temporal motion information. In the last row we show the result of the standard average pooling. When the motion of the action is apparent, rank pooling method seems to capture this well. What is more interesting is that, not only rank pooling separates running in one direction from the other direction, but also seems to capture the periodicity of the motion to an extent, see the last column of Figure 5 that depicts *running*.

6 Experiments

Now we present a detailed experimental evaluation of rank pooling.

Datasets. As the proposed methodology is not specific to an action type or class of actions, we present experiments in a broad range of datasets. We follow exactly the same experimental settings per dataset, using the same training and test splits and the same features as reported by the state-of-the-art methods.

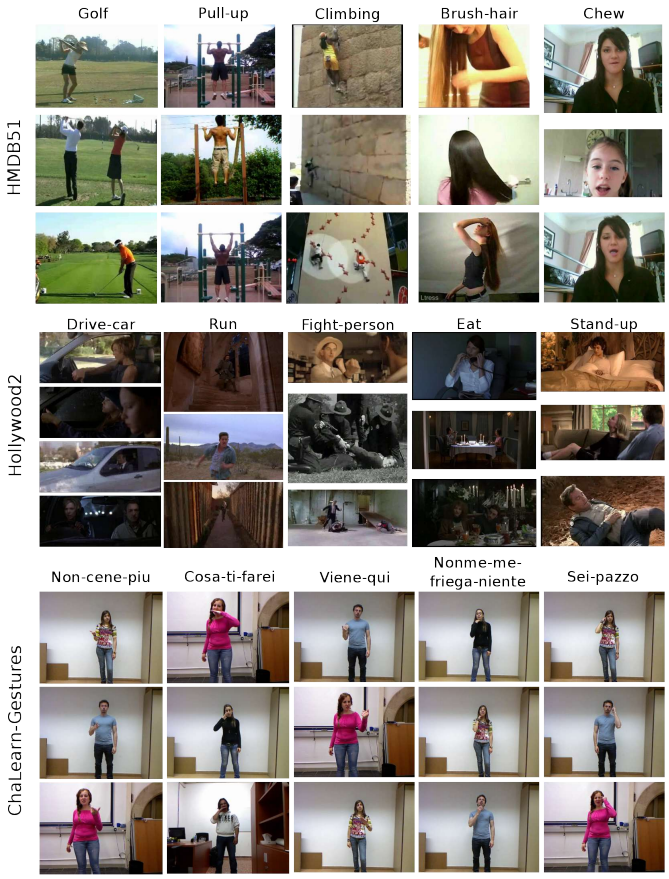


Fig. 6: Some example frames from the top performing categories of the HMDB51, Hollywood2, and ChaLearn-Gestures dataset, respectively.

HMDB51 dataset [28]. This is a generic action classification dataset composed of roughly 7,000 clips divided into 51 action classes. Videos and actions of this dataset are subject to different camera motions, viewpoints, video quality and occlusions. As done in the literature we use a one-vs-all classification strategy and report the mean classification accuracy over three standard splits provided by the authors in [28]. Some example frames from this challenging dataset are shown in Figure 6.

Hollywood2 dataset [32] This dataset has been collected from 69 different Hollywood movies that include 12 action classes. It contains 1,707 videos in total where 823 videos are used for training and 884 are used for testing. Training and test videos are selected from different movies. The performance is measured by mean average precision (mAP) over all classes, as in [32].

MPII cooking activities dataset [48]. This dataset was created to evaluate fine-grained action classification. It is composed of 65 different actions that take place continuously within 8 hours of recordings. As the kitchen remains the same throughout the recordings, the classification focuses mainly on the content of the actions and cannot benefit from potentially discriminative background information (*e.g.* driving a car always takes place inside a car). We compute per class average precision using the same procedure as in [48] and report the final mAP.

ChaLearn Gesture Recognition dataset [9]. This dataset

contains 23 hours of Kinect data of 27 persons performing 20 Italian gestures. The data includes RGB, depth, foreground segmentation and Kinect skeletons. The data is split into train, validation and test sets, with in total 955 videos each lasting 1 to 2 minutes and containing 8 to 20 non-continuous gestures. As done in the literature, we report precision, recall and F1-score measures on the validation set.

Rank pooling and baselines. In Sec. 6.1 and 6.3 we compare different variants of rank pooling. As a first baseline we use the state-of-the-art trajectory features (*i.e.* improved trajectories and dense trajectories) and pipelines as in [65], [66]. As this trajectory-based baseline mainly considers *local temporal information* we refer to this baseline as *local*. We also compare with temporal pyramids (*TP*), by first splitting the video into two equal size sub-videos, then computing a representation for each of them like spatial pyramids [33]. For these baselines, at frame level we apply non-linear feature maps (*i.e.* power normalization for Fisher vectors and chi-squared kernel maps for bag-of-words-based methods). We also compare different versions of rank pooling, we denote the forward rank pooling by *FDRP*, the reverse & forward rank pooling by *RFDRP*, the non-linear forward rank pooling by *NL-FDRP* and the non-linear reverse & forward rank pooling by *NL-RFDRP*.

Implementation details. In principle there is no constraint on the type of linear ranking machines we employ for learning rank pooling. We have experimented with state-of-the-art ranking implementation RankSVM [25] and SVR [55]. Both these methods can be used to solve learning to rank problems formulated in equation 2. We observe that both methods capture evolution of the video appearances equally well. As for SVR the learning convergence is notably faster, we will use the SVR solver of Liblinear in this paper (C = 1).

For HMDB51 and Hollywood2 datasets we use state-of-the art improved trajectory features [66] with Fisher encoding [43]. As done in the literature, we extract HOG, HOF, MBH, and trajectory (TRJ) features from the videos. We create GMMs of size 256 after applying PCA with a dimensionality reduction of factor 0.5 on each descriptor. As done in [66], we also apply the square-root trick on all descriptors except for TRJ.

In order to compute non-linear rank pooling, we apply features maps *(posneg)* followed by a L2-normalization on individual Fisher vectors extracted from each video frame. For linear rank pooling, we just use Fisher vectors without any power normalization.

For MPII cooking dataset we use the features provided by the authors [48], that is bag-of-words histograms of size 4000 extracted from dense trajectory features [65] (HOG, HOF, MBH and TRJ). As we use bag-of-words for this dataset, in order to compute non-linear rank pooling, we apply χ^2-kernel maps on individual bag-of-words histograms after the construction of the vector valued function as explained in section 4.

For the ChaLearn Gesture Recognition dataset we start from the body joints [51]. For each frame we calculate the relative location of each body joint w.r.t. the torso joint. Then, we scale these relative locations in the range [0,1]. We use a dictionary of 100 words to quantize these skeleton features. Similar to MPII cooking dataset, in order to compute non-linear rank pooling and for all baselines we use chi-squared kernel maps.

We train non-linear SVM classifiers with feature kernel maps for the final classification. Whenever we use bag-of-words representation we compute χ^2-kernel maps over the final video representation and then L2 normalize them. We use this strategy for both baselines and rank pooling. Similarly, when Fisher vectors are used, we use posneg feature map and L2 normalization for the final video representation. The C parameter of SVM is cross-validated over the training set using two-fold cross-validation to optimize the final evaluation criteria (mAP, classification accuracy or F-score). When features are fused (combined) we use the average kernel strategy. We provide code for computing rank pooling in a public website [1].

Execution time. Rank pooling takes about 0.9 ± 0.1 sec per video on the Hollywood2 dataset excluding the Fisher vector computation. The proposed algorithm is linear on the length of the video.

6.1 Rank pooling: Frame representations & encodings

We first evaluate the three options presented in Section 4 for the frame representation, *i.e. independent frame*, *moving average* and *time varying mean vector* representations. We perform the experiments with Fisher vectors on the Hollywood2 dataset and summarize the results in Table 1. Similar trends were observed with dense trajectory features, bag-of-words and other datasets.

From the comparisons, we make several observations that validate our analysis. First, applying ranking functions directly on the Fisher vectors from the frame data captures only a moderate amount of the temporal information. Second, *moving average* applied with ranking seems to capture video-wide temporal information better than applying ranking functions directly on the frame data. However, the *time varying mean vector* consistently outperforms the other two representations by a considerable margin and for all features. We believe this is due to two reasons. First, *moving average* and *time varying mean vector* methods smooth the original signal. This reduces the noise in the signal. Therefore, it allows the ranking function to learn meaningful VTE. Secondly, the appearance information of the *time varying mean vectors* is more correlated with the time variable. The ranking function exploits this correlation to learn the evolution of the appearance over time in the video signal.

We conclude that *time varying mean vectors* are better for capturing the video-wide evolution of appearance of videos when applied with rank pooling. In the rest of the experiments we use the time varying mean vectors.

Last, we evaluate the contribution of the time-varying mean vectors when used along with other pooling methods such as average pooling. We perform an experiment on Hollywood2 using the MBH features. The average pooling on top of time-varying mean vectors gives an improvement of 0.5% (relative to average pooling on FV directly) only, indicating that for average pooling, there is no advantage of time varying mean vectors.

6.2 Action classification

Next, we present a detailed analysis of the action classification results in HMDB51, Hollywood2, MPII Cooking and ChaLearn Gesture recognition datasets (see Table 2, 3, 4 and 5 respectively).

1. The code for computing rank pooling, as well as scripts for running experiments for the different datasets can be found in http://bitbucket.org/bfernando/videodarwin.

	HOG	HOF	MBH	TRJ	Comb.
Independent frames	41.6	52.1	54.4	43.0	57.4
Moving average (T=20)	42.2	54.6	56.6	44.4	59.5
Moving average (T=50)	42.2	55.9	58.1	46.0	60.8
Time varying mean vectors	**45.3**	**59.8**	**60.5**	**49.8**	**63.6**

TABLE 1: Comparison of different video representations for rank pooling. Results reported in mAP on the Holywood2 dataset using FDRP with Fisher vectors. As also motivated in Sec. 4, the time varying mean vector representation captures better the video-wide temporal information present in a video.

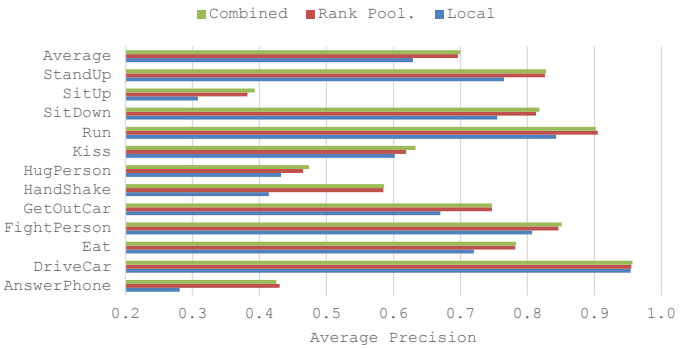


Fig. 7: Per class AP in the Hollywood2 dataset. The AP is improved significantly for all classes, with an exception of "Drive car", where context already provides useful information.

Rank pooling obtains better results in comparison to local temporal methods. Accurately modeling the evolution of appearance and motion, allows to capture more relevant information for a particular action. These results confirm our hypothesis that what makes an action most discriminative from other actions is mostly the video-wide evolution of appearance and motion information of that action. The forward and reverse rank pooling variant reports consistent improvement over forward-only rank pooling, further improved when non-linear rank pooling is employed. It is interesting to see that this trend can be observed in all four datasets too. Overall, local methods combined with rank pooling bring a substantial absolute increase over local methods ($+6.6\%$ for HMDB51, $+7.1\%$ for Holywood2, $+8.6\%$ for MPII Cooking, $+9.3\%$ for ChaLearn).

Analysis of action classification results. Looking at the individual results for the Hollywood2 dataset shown in Figure 7, we observe that almost all actions benefit the same, about a 7% average increase. Some notable exceptions are "answer phone", which improves by 14% and "handshake", which improves by 17%. For "drive car" there is no improvement. The most probable cause is that the car context already provides enough evidence for the classification of the action, also reflected in the high classification accuracy of the particular action. Our method brings improvements for periodic actions such as "run, handshake" as well as non-periodic actions such as "get-out-of-car".

For the case of the ChaLearn dataset (Table 10), we see that rank pooling is able to achieve superior results without requiring to explicitly define task-specific steps, e.g. hand-posture or hand-trajectory modeling [39].

To gain further insight we investigate the mean similarity computed over classes on MPII cooking dataset with BOW-based MBH features. We construct the dot product kernel matrix using all the samples and then compute the mean similarity between classes, see Figure 8. The rank pooling kernel matrix (Figure 8 (right)) appears to be more discriminative than the one with max-pooled features (Figure 8 (left)). The action "smell" (#41) seems very difficult to discriminate either using max-pooling or rank pooling method. Actions "sneeze" (#44) and "stamp" (#45) seem to be very similar in-terms of appearances, however with rank pooling we can discriminate them better. Actions like "take & put in cupboard" (#47), "take & put in drawer"(#48), "take & put in fridge" (#49) and "take & put in oven" (#50) seem to be the most confused ones for rank pooling. These actions differ in the final instrument, but not in the dynamics of the action.

	HOG	HOF	MBH	TRJ	Combined
Local	39.2	48.7	50.8	36.0	55.2
TP	40.7	52.2	53.5	37.0	57.2
FDRP	39.2	52.7	53.0	37.0	57.9
RFDRP	41.6	53.3	54.6	39.1	59.1
NL-FDRP	44.2	54.7	55.2	37.7	61.0
NL-RFDRP	**46.6**	**55.7**	**56.7**	**39.5**	**61.6**
Local + FDRP	42.4	53.7	54.3	39.7	59.3
Local + RFDRP	42.7	53.9	54.9	40.0	59.4
Local + NL-FDRP	45.6	56.2	56.2	41.0	61.3
Local + NL-RFDRP	**47.0**	**56.6**	**57.1**	**41.3**	**61.8**

TABLE 2: One-vs-all accuracy on HMDB51 dataset [28]

	HOG	HOF	MBH	TRJ	Combined
Local	47.8	59.2	61.5	51.2	62.9
TP	52.0	61.1	63.6	52.1	64.8
FDRP	45.3	59.8	60.5	49.8	63.6
RFDRP	50.5	63.6	65.5	**55.1**	67.9
NL-FDRP	52.8	60.8	62.9	50.2	65.6
NL-RFDRP	**56.7**	**64.7**	**66.9**	54.5	**69.6**
Local + FDRP	50.2	62.0	64.4	53.6	66.7
Local + RFDRP	52.7	64.3	66.2	55.9	68.7
Local + NL-FDRP	54.7	62.9	64.9	54.4	67.6
Local + NL-RFDRP	**57.4**	**65.2**	**67.3**	**56.1**	**70.0**

TABLE 3: Results in mAP on Hollywood2 dataset [38]

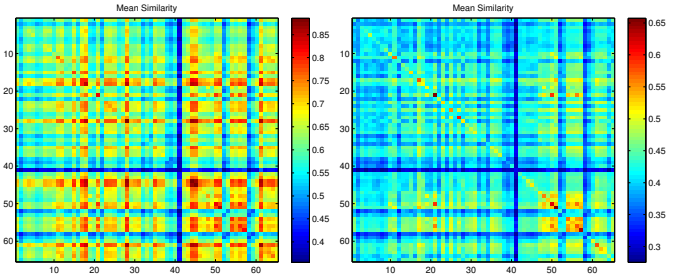


Fig. 8: Mean class similarity obtained with (left) max-pooling and (right) rank pooling on MPII Cooking activities dataset using BOW-based MBH features extracted on dense trajectories. Non-linear forward rank pooling are used for our method.

6.3 Rank pooling analysis

Stability to dropped frames We analyze the stability of rank pooling compared to average pooling and temporal pyramids. For this experiment we use Hollywood2 dataset and MBH features with Fisher vectors. We gradually remove 5%, 10%, ... 25% of random frames from each video from both train and test sets and then measure the change in mean average precision.

	HOG	HOF	MBH	TRJ	Combined
Local	49.4	52.9	57.5	50.2	63.4
TP	**55.2**	56.5	61.6	**54.6**	64.8
FDRP	50.7	53.5	58.0	48.8	62.4
RFDRP	53.1	55.2	61.4	51.9	63.5
NL-FDRP	52.8	**60.8**	**62.9**	50.2	**65.6**
NL-RFDRP	50.6	53.8	56.5	50.0	62.7
Local + FDRP	61.4	65.6	69.0	62.7	71.5
Local + RFDRP	**63.7**	**65.9**	**69.9**	**63.0**	71.7
Local + NL-FDRP	63.5	65.0	68.6	61.0	71.8
Local + NL-RFDRP	64.6	65.7	68.9	61.2	**72.0**

TABLE 4: Results in mAP on MPII Cooking fine grained action dataset [48].

	Precision	Recall	F-score
Local	65.9	66.0	65.9
TP	67.7	67.7	67.7
FDRP	60.6	60.4	60.5
RFDRP	65.5	65.1	65.3
NL-FDRP	69.5	69.4	69.4
NL-RFDRP	**74.0**	**73.8**	**73.9**
Local + FDRP	71.4	71.5	71.4
Local + RFDRP	73.9	73.8	73.8
Local + NL-FDRP	71.8	71.9	71.8
Local + NL-RFDRP	**75.3**	**75.1**	**75.2**

TABLE 5: Detailed analysis of precision and recall on the ChaLearn gesture recognition dataset [9]

We present in Figure 9 the relative change in mAP after frame removal. Typically, we would expect the mAP to decrease. Interestingly, removing up-to 20% of the frames from the video does not significantly change the results of rank pooling; in-fact we observe a slight relative improvement. This is a clear indication of the stability of rank pooling and an advantage of learning-based temporal pooling. As expected, the mAP decreases for both average pooling method and the temporal pyramids method as the number of frames that are removed from videos increases. For average pooling mAP seems to drop almost in an exponential manner. However, it should be noted that 25% of the video frames is a significant amount of data. We believe the results illustrate the stability of rank pooling.

Effect of video length. In this experiment we analyse how the length of the video influences the testing performance. We train rank pooling-based classifiers as before using the entire training set and then partition the test set into three segments. Then, we compare the action classification accuracies obtained with different video lengths. Results are shown in Figure 10. Interestingly, the longer the video, the better our method seems to perform. This is not as surprising, since longer videos are more likely to contain more dynamic information compared to shorter videos. Also, for longer videos averaging will likely be more affected by outliers. What is more noteworthy is the relative difference in accuracy between very long and very short videos, approximately 6%. We conclude that our method is capable of capturing the dynamics of short videos as well as of long videos.

The impact of feature maps on Fisher Vectors. In this section we evaluate the effect of different feature maps during ranker function construction and final video classification. We use MBH features as the representation and evaluate the activity recognition performance on Hollywood2 dataset. Results are reported in Table 6.

We observe that the combination of posneg feature maps both for computing rank pooling, as well as computing the final classification kernel, outperforms all other alternatives. The closest competitor is when we use the posneg kernel for computing the rank pooling features. In general, we observe that for the classification kernel the different combinations perform somewhat similarly, given a fixed rank pooling feature map. We conclude the highest accuracies with rank pooling are obtained when we apply the posneg feature map, irrespective to the classification kernel.

Functional parameters as temporal pooling. In this experiment we evaluate several parametric models in which we can use the parameters to represent a video. More specifically, we evaluate rank pooling using SVR [55] and RankSVM [25] subspace pooling using the first principal eigenvector only as the video

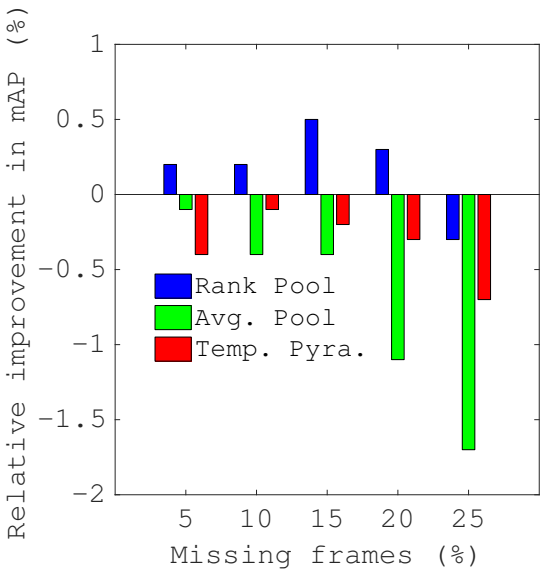
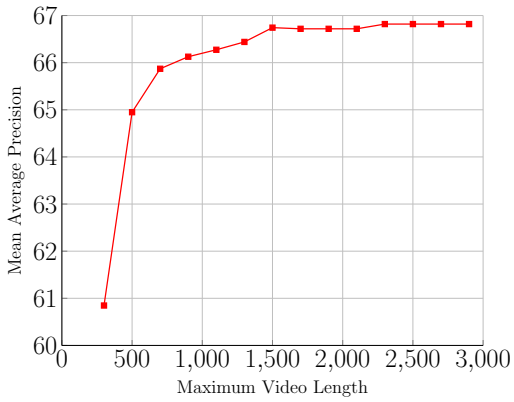


Fig. 9: Comparison of action recognition performance after removing some frames from each video randomly on Hollywood2. rank pooling appears to be stable even when up to 20% of the frames are missing.



(a)

Fig. 10: Hollywood2 action recognition performance with respect to the length of the video using our rank pooling method.

Ranking Feature Map	Classifier Feature Map	mAP
$\sqrt{\lvert x\rvert}$	$\text{sgn}(u)\sqrt{\lvert u\rvert}$	50.0
$\sqrt{\lvert x\rvert}$	$\sqrt{\lvert u\rvert}$	54.0
$\sqrt{x^*}$	$\sqrt{u^*}$	**66.1**
$\sqrt{x^*}$	$\text{sgn}(u)\sqrt{\lvert u\rvert}$	65.4
$\sqrt{x^*}$	$\sqrt{\lvert u\rvert}$	63.7

TABLE 6: Comparison of different features maps for ranking and classification. We use different symbols, x and u, to avoid the confusion, as u refers to the feature encodings (*e.g.*, Fisher vectors) that we use to compute rank pooling, while u refers to the rank pooling features. x^* stands for the the input to the posneg kernel, namely $x^* = [x^+, x^-]^T$.

Parameter pooling	mAP
Rank Pooling with RankSVM	66.0
Rank Pooling with SVR	66.5
Subspace pooling	56.4
Robust Subspace Pooling	64.1
HMM pooling	17.8
Neural Network pooling	21.1

TABLE 7: Pooling parameters as representations from different parametric models. R-PCA stands for the Robust PCA. Experiments were conducted on the Hollywood2 dataset using MBH features.

representation (Sec. 3.4) and robust subspace pooling using the first eigenvector only as described in equation. 7. Additionally, we use the parameters of two layered fully connected neural networks as a video representation. In this case, the neural network consists of one hidden layer (10 hidden units) and the input layer. It is trained to map frame data to the time variable hoping to capture dynamics similar to SVR [55]. Furthermore, we train a Hidden Markov Model using the input video data and then use the transition and observation probability matrix as a video representation. We run the experiment on Hollywood2 dataset with MBH features and show results in Table 7.

We observe that standard PCA-based subspace pooling is less accurate than both the SVR and the RankSVM rank pooling. The robust subspace pooling, which deals better with very low data volume to dimensionality ratios, captures the video-wide temporal evolution reasonably well. However, pooling from ranker SVR machines works best. Interestingly, the neural network and HMM performance is poor. Probably, the neural network overfits easily compared to the SVR machines.

We conclude that for moderately long videos using the parameters of simpler, linear machines as the representation for the sequence data is to be preferred to avoid overfitting. However, we expect that for very long videos or for even richer frame representations more complex dynamics could arise. In these cases higher capacity methods, like neural networks, would likely capture better the underlying dynamics.

CNN features for action classification In this experiment we evaluate our method using the convolutional neural network (CNN)-based features. We use the activations of the first fully connected layer of vgg-16 network [53] to represent each frame in a video. We compare several pooling techniques using Hollywood2 dataset in Table 8. Rank pooling by itself does not perform that well compared to local (average pooling) method (32.2 mAP vs. 39.0 mAP). However, the combination of rank pooling with the local approach improves the results to 46.4 mAP. The CNN features used in this experiment are 4096 dimensional and are not fine tuned for action classification. As the pre-trained features are trained specifically for appearance-based classification, we combine CNN features with MBH features. With the local approach, the combination of CNN and MBH results in 65.6 mAP. The best results are obtained with local pooling of CNN and temporal pooling of MBH. We believe this strategy exploits the advantage of both appearance information and dynamics of videos.

Method	mAP
Local(cnn)	39.0
NL-RFDRP(cnn)	32.2
Local(cnn)+NL-RFDRP(cnn)	46.4
Local(cnn)+Local(MBH)	65.6
Local(cnn)+NL-RFDRP(MBH)	**70.1**
Local(cnn+MBH)+NL-RFDRP(MBH)	69.7
Local(cnn+MBH)+NL-RFDRP(cnn+MBH)	69.5

TABLE 8: Results obtained on Hollywood2 dataset using CNN (vgg-16 network [53]) features.

	HMDB51	Hollywood2	Cooking
Rank pooling+CNN	65.8	**75.2**	–
Rank pooling	63.7	73.7	**72.0**
Hoai et al *[19]*	60.8	73.6	–
Peng et al *[42]*	**66.8**	–	–
Wu et al *[69]*	56.4	–	–
Jain et al *[22]*	52.1	62.5	–
Wang et al *[66]*	57.2	64.3	–
Wang et al *[65]*	46.6	58.2	–
Taylor et al *[62]*	–	46.6	–
Zhou et al *[75]*	–	–	70.5
Rohrbach et al *[48]*	–	–	59.2

TABLE 9: Comparison of the proposed approach with the state-of-the-art methods sorted by reverse chronological order. Results reported in mAP for Hollywood2 and Cooking datasets. For HMDB51 we report one-vs-all classification accuracy.

6.4 State-of-the-art and discussion.

Last, we compare the nonlinear forward and reverse rank pooling combined with the local temporal information with the latest state-of-the-art in action recognition. We summarize the results in Table 9 and Table 10. Note that for Hollywood2 and HMDB51, we use data augmentation by mirroring the videos as in [19], which brings a further 5% improvement, and combine with max-pooled CNN features to capture static appearance information explicitly.

	Precision	Recall	F-score
Rank pooling	**75.3**	**75.1**	**75.2**
Martinez-Camarena et al *[39]*	61.4	61.9	61.6
Pfister et al *[44]*	61.2	62.3	61.7
Yao et al *[72]*	–	–	56.0
Wu et al *[70]*	59.9	59.3	59.6

TABLE 10: Comparison of the proposed approach with the state-of-the-art methods on ChaLearn gesture recognition dataset sorted by reverse chronological order.

By the inspection of Tables 9 and 10, as well as from the results in the previous experiments, we draw several conclusions.

First, rank pooling is useful and robust for encoding video-wide, temporal information. Second, rank pooling is complementary to action recognition methods that compute local temporal features, such as improved trajectory-based features [66]. In fact, fusing rank pooling with the previous state-of-the-art in local motion and appearance, we improve up to 10%. Third, rank pooling is complimentary with static feature representations such as CNN-based max pooled features. Forth, rank pooling is only outperformed on HMDB51 by [42], who combine their second layer Fisher vector features with normal Fisher vectors to arrive at 205K dimensional vectors and a 66.8% accuracy. When using Fisher vectors like rank pooling does, Peng *et al*. [42] obtain 56.2%, which is 10% lower than what we obtain with rank pooling.

7 Discussion and Conclusion

We introduce rank pooling, a new pooling methodology that models the evolution of appearance and dynamics in a video. Rank pooling is an unsupervised, learning based temporal pooling method, which aggregates the relevant information throughout a video via fitting learning-to-rank models and using their parameters as the new representation of the video. We show the regularized learning of the learning-to-rank algorithms, as well as the minimization of the temporal ordering empirical risk, has in fact favorable generalization properties that allow us to capture robust temporal and video dynamics representations. Moreover, we show that the ranking models can be replaced with different parameteric models, such as principal component analysis. However, experiments reveal that learning-to-rank linear machines seem to capture the temporal dynamics in videos best. We demonstrate that a temporal smoothing and sequence pre-processing is important for modelling the temporal evolution in sequences. Last, we show that designing kernels that separate the positive from the negative part of the incoming features has a substantial effect on the final classification using rank pooling. Based on extensive experimental evaluations on different datasets and features we conclude that, our method is applicable to a wide variety of frame-based representations for capturing the global temporal information of a video.

In the current work we focused mainly on exploring rank pooling within an action classification setting on moderately long videos. However, we believe that rank pooling could easily be exploited in other tasks too, such as video caption generation, action detection, video retrieval, dynamic texture and video summarization.

We conclude that rank pooling is a novel and accurate method for capturing the temporal evolution of appearances and dynamics in videos.

8 Acknowledgment

The authors acknowledge the support of FP7 ERC Starting Grant 240530 COGNIMUND, KU Leuven DBOF PhD fellowship, the FWO project *Monitoring of abnormal activity with camera systems*, iMinds High-Tech Visualization project and the Australian Research Council Centre of Excellence for Robotic Vision (project number CE140100016).

References

[1] "https://www.youtube.com/yt/press/statistics.html."
[2] R. Arandjelović and A. Zisserman, "Three things everyone should know to improve object retrieval," in *CVPR*, 2012.
[3] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould, "Dynamic image networks for action recognition," in *CVPR*, 2016.
[4] O. Bousquet and A. Elisseeff, "Stability and generalization," *JMLR*, vol. 2, pp. 499–526, 2002.
[5] G. Chéron, I. Laptev, and C. Schmid, "P-cnn: pose-based cnn features for action recognition," in *ICCV*, 2015, pp. 3218–3226.
[6] R. De Geest and T. Tuytelaars, "Dense interest features for video processing," in *ICIP*, 2014.
[7] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto, "Dynamic textures," *IJCV*, vol. 51, no. 2, pp. 91–109, 2003.
[8] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *CVPR*, 2015.
[9] S. Escalera, J. Gonzàlez, X. Baró, M. Reyes, O. Lopes, I. Guyon, V. Athitsos, and H. Escalante, "Multi-modal gesture recognition challenge 2013: Dataset and results," in *ICMI*, 2013.
[10] B. Fernando, P. Anderson, M. Hutter, and S. Gould, "Discriminative hierarchical rank pooling for activity recognition," in *CVPR*, 2016.
[11] B. Fernando, E. Gavves, D. Muselet, and T. Tuytelaars, "Learning-to-rank based on subsequences," in *ICCV*, 2015.
[12] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars, "Modeling video evolution for action recognition," in *CVPR*, 2015.
[13] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *ICCV*, 2013.
[14] A. Gaidon, Z. Harchaoui, and C. Schmid, "Actom sequence models for efficient action detection," in *CVPR*, 2011.
[15] A. Gaidon, Z. Harchaoui, C. Schmid *et al.*, "Recognizing activities with cluster-trees of tracklets," in *BMVC*, 2012.
[16] E. Gavves, B. Fernando, C. Snoek, A. Smeulders, and T. Tuytelaars, "Local alignments for fine-grained categorization," *IJCV*, vol. 111, no. 2, pp. 191–212, 2014.
[17] E. Gavves, C. G. M. Snoek, and A. W. M. Smeulders, "Convex reduction of high-dimensional kernels for visual classification," in *CVPR*, 2012.
[18] G. Gkioxari and J. Malik, "Finding action tubes," in *CVPR*, June 2015.
[19] M. Hoai and A. Zisserman, "Improving human action recognition using score distribution and ranking," in *ACCV*, 2014.
[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[21] H. Izadinia and M. Shah, "Recognizing complex events using large margin joint low-level event model," in *ECCV*, 2012.
[22] M. Jain, H. Jégou, and P. Bouthemy, "Better exploiting motion for better action recognition," in *CVPR*, 2013.
[23] M. Jain, J. van Gemert, H. Jegou, P. Bouthemy, and C. G. Snoek, "Action localization with tubelets from motion," in *CVPR*, 2014.
[24] M. Jain, J. van Gemert, and C. G. M. Snoek, "What do 15,000 object categories tell us about classifying and localizing actions?" in *CVPR*, 2015.
[25] T. Joachims, "Training linear svms in linear time," in *ICKDD*, 2006.
[26] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014.
[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.
[28] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: a large video database for human motion recognition," in *ICCV*, 2011.
[29] Z. Lan, D. Yao, M. Lin, S.-I. Yu, and A. Hauptmann, "The best of both worlds: Combining data-independent and data-driven approaches for action recognition," *arXiv preprint arXiv:1505.04427*, 2015.
[30] I. Laptev, "On space-time interest points," *IJCV*, vol. 64, pp. 107–123, 2005.
[31] I. Laptev and T. Lindeberg, "Space-time interest points," in *ICCV*, 2003.
[32] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *CVPR*, 2008.
[33] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *CVPR*, 2006.
[34] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *CVPR*, 2011.
[35] B. Li, M. Ayazoglu, T. Mao, O. I. Camps, and M. Sznaier, "Activity recognition using dynamic subspace angles," in *CVPR*. IEEE, 2011, pp. 3193–3200.
[36] T.-Y. Liu, "Learning to rank for information retrieval," *Foundations and Trends in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.
[37] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-svms for object detection and beyond," in *ICCV*, 2011.
[38] M. Marszałek, I. Laptev, and C. Schmid, "Actions in context," in *CVPR*, 2009.

[39] M. Martínez-Camarena, J. Oramas M, and T. Tuytelaars, "Towards sign language recognition based on body parts relations," in *ICIP*, 2015.
[40] M. Mazloom, E. Gavves, and C. G. Snoek, "Conceptlets: Selective semantics for classifying video events," *Multimedia, IEEE Transactions on*, vol. 16, no. 8, pp. 2214–2228, 2014.
[41] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko, "Function representation in geometric modeling: concepts, implementation and applications," *The Visual Computer*, vol. 11, no. 8, pp. 429–446, 1995.
[42] X. Peng, C. Zou, Y. Qiao, and Q. Peng, "Action recognition with stacked fisher vectors," in *ECCV*, 2014.
[43] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, "Large-scale image retrieval with compressed fisher vectors," in *CVPR*, 2010.
[44] T. Pfister, J. Charles, and A. Zisserman, "Domain-adaptive discriminative one-shot learning of gestures," in *ECCV*, 2014.
[45] V. Ponce-López, H. J. Escalante, S. Escalera, and X. Baró, "Gesture and action recognition by evolved dynamic subgestures," in *BMVC*, 2015.
[46] M. Raptis, I. Kokkinos, and S. Soatto, "Discovering discriminative action parts from mid-level video representations," in *CVPR*, 2012.
[47] J. Revaud, M. Douze, C. Schmid, and H. Jégou, "Event retrieval in large video collections with circulant temporal encoding," in *CVPR*, 2013.
[48] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele, "A database for fine grained activity detection of cooking activities," in *CVPR*, 2012.
[49] M. Rohrbach, M. Regneri, M. Andriluka, S. Amin, M. Pinkal, and B. Schiele, "Script data for attribute-based recognition of composite activities," in *ECCV*, 2012.
[50] M. S. Ryoo and J. K. Aggarwal, "Recognition of composite human activities through context-free grammar based representation," in *CVPR*, 2006.
[51] J. Shotton, A. W. Fitzgibbon, M. Cook, T. Sharp, M. Finochio, R.Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *CVPR*, 2011.
[52] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," *CoRR*, vol. abs/1406.2199, pp. 1–8, 2014.
[53] ——, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.
[54] C. Sminchisescu, A. Kanaujia, and D. Metaxas, "Conditional models for contextual human motion recognition," *Computer Vision and Image Understanding*, vol. 104, no. 2, pp. 210–220, 2006.
[55] A. Smola and V. Vapnik, "Support vector regression machines," *Advances in neural information processing systems*, vol. 9, pp. 155–161, 1997.
[56] Y. Song, L.-P. Morency, and R. Davis, "Action recognition by hierarchical sequence summarization," in *CVPR*, 2013.
[57] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms," *CoRR*, vol. abs/1502.04681, 2015.
[58] W.-T. Su, Y.-H. Lu, and A. S. Kaseb, "Harvest the information from multimedia big data in global camera networks," in *IEEE International Conference on Multimedia Big Data*, 2015.
[59] C. Sun and R. Nevatia, "Active: Activity concept transitions in video event classification," in *ICCV*, 2013.
[60] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014, pp. 3104–3112.
[61] K. Tang, L. Fei-Fei, and D. Koller, "Learning latent temporal structure for complex event detection," in *CVPR*, 2012.
[62] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *ECCV*, 2010.
[63] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," 2008.
[64] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *PAMI*, vol. 34, pp. 480–492, 2012.
[65] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *IJCV*, vol. 103, pp. 60–79, 2013.
[66] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *ICCV*, 2013.
[67] Y. Wang and G. Mori, "Hidden part models for human action recognition: Probabilistic versus max margin," *PAMI*, vol. 33, pp. 1310–1323, 2011.
[68] D. Wu and L. Shao, "Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition," in *CVPR*, 2014.
[69] J. Wu, Y. Zhang, and W. Lin, "Towards good practices for action video encoding," in *CVPR*, 2014.
[70] J. Wu, J. Cheng, C. Zhao, and H. Lu, "Fusing multi-modal features for gesture recognition," in *ICMI*, 2013.
[71] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden markov model," in *CVPR*, Jun 1992, pp. 379–385.
[72] A. Yao, L. Van Gool, and P. Kohli, "Gesture recognition portfolios for personalization," in *CVPR*, 2014.
[73] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *CVPR*, 2015.
[74] J. Zepeda and P. Perez, "Exemplar svms as visual feature encoders," in *CVPR*, 2015.
[75] Y. Zhou, B. Ni, S. Yan, P. Moulin, and Q. Tian, "Pipelining localized semantic features for fine-grained action recognition," in *ECCV*, 2014.

Basura Fernando is a Research Fellow at the Australian National University. He received Ph.D. from KU Leuven Belgium in 2015, M.Sc. in 2011 from the European Erasmus Mundus Master program CIMET and B.Sc in 2007 from the University of Moratuwa Sri Lanka. He is a Computer Vision researcher and his research interests includes visual representation learning, data-mining, domain adaptation, object recognition, action recognition, video analysis, and deep learning.

Efstratios Gavves is an Assistant Professor with the University of Amsterdam in the Netherlands. He received his Ph.D. in 2014 at the University of Amsterdam. He was a post-doctoral researcher at the KU Leuven from 2014 - 2015. He has authored several papers in major computer vision and multimedia conferences and journals. His research interests include, but are not limited to, statistical and deep learning with applications on computer vision tasks, like object recognition, image captioning, action recognition, tracking, memory networks and recurrent networks.

Jose Oramas M. is a post-doctoral researcher at the KU Leuven, where he received his Ph.D. in 2015 under the advice of Prof. Tinne Tuytelaars and Prof. Luc de Raedt. He received his Computer Engineering degree from the Escuela Superior Politécnica del Litoral, Ecuador. His research focuses on understanding how groups of elements within images (objects, object-parts, regions, trajectories, etc.) behave and how the relations between them can be exploited to address computer vision problems.

Amir Godrati is a Ph.D. candidate since 2013 at VISICS group at KU Leuven under supervision of Prof. Tinne Tuytelaars. He finished M.Sc. in Computer Science (Artificial Intelligence) at Sharif University of Technology, Tehran, Iran, advised by Prof. Shohreh Kasaei. He received his B.Sc. degree in Computer Engineering from Amirkabir University of Technology, Tehran, Iran. His research interests are in video analysis, action recognition.

Tinne Tuytelaars is a Professor at KU Leuven Belgium. She received a Master of Electrical Engineering from the KU Leuven , Belgium in 1996. Since her graduation, she has been working within ESAT - PSI of the KU Leuven, where she obtained her Ph.D. in 2000. She has been one of the program chairs for ECCV 2014 and one of the general chairs of CVPR 2016. Her research interests are object recognition, action recognition, multimodal analysis and image representations.