Action Recognition with Dynamic Image Networks

Hakan Bilen, Basura Fernando, Efstratios Gavves, and Andrea Vedaldi

Abstract—We introduce the concept of *dynamic image*, a novel compact representation of videos useful for video analysis especially when convolutional neural networks (CNNs) are used. The dynamic image is based on the rank pooling concept and is obtained through the parameters of a ranking machine that encodes the temporal evolution of the frames of the video. Dynamic images are obtained by directly applying rank pooling on the raw image pixels of a video producing a single RGB image per video. This idea is simple but powerful as it enables the use of existing CNN models directly on video data with fine-tuning. We present an efficient and effective approximate rank pooling operator, speeding it up orders of magnitude compared to rank pooling. Our new approximate rank pooling CNN layer allows us to generalize dynamic images to dynamic feature maps and we demonstrate the power of our new representations on standard benchmarks in action recognition achieving state-of-the-art performance.

Index Terms—human action classification, video classification, motion representation, deep learning, convolutional neural networks.

arXiv:1612.00738v1 [cs.CV] 2 Dec 2016

1 Introduction

Videos comprise a large majority of the visual data in existence, surpassing by a wide margin still images. Therefore understanding the content of videos accurately and on a large scale is of paramount importance. The advent of modern learnable representations such as deep convolutional neural networks (CNNs) has improved dramatically the performance in many image understanding tasks. Since videos are composed of a sequence of still images, some of these improvements have been shown to transfer to videos directly. However, it remains unclear *how videos could be optimally represented*. For example, one can look at a video as a sequence of still images, perhaps enjoying some form of temporal smoothness, or as a subspace of images or image features, or as the output of a neural network encoder.

Early work [8], [22], [54] in the understanding of motion considered video frames as dynamic sequences of images. Doretto *et al.* [8] define a sequence of frames as (linear) dynamic textures, where the pixel intensities can be reconstructed as a weighted summation of spatial filters, associated to an auto-regressive moving average process. Wang *et al.* [54] represent the motion in a video as the moments of a generative distribution (mixture of gaussians specifically) of temporally local, flow-based appearance variations. The underlying assumption of these earlier works that there exists a generative process that produces the various, simple or complex, motion patterns.

More recent methods [14], [34], [45] approach the problem of understanding motion by considering videos as stacked frames, whose appearance and flow is defined by the types of actions present in the videos. The majority of these methods rely on convolutional or recurrent neural network architectures and learn spatio-temporal filters, such that they maximize the predictive capabilities of the final system. These methods produce the highest accuracies in action recognition tasks, as their primary purpose is to model the action classes and not the motion itself.

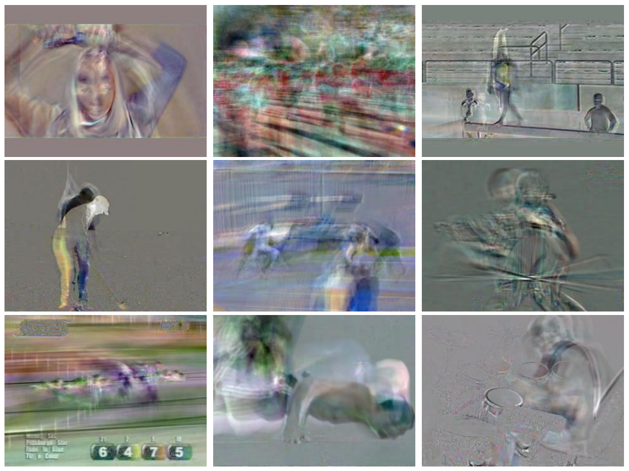


Fig. 1: Dynamic images summarizing the actions and motions that happen in images in standard 2d image format. *Can you guess what actions are visualized just from their dynamic image motion signature?* [1]

In this paper, we build on the former intuition, namely there exists an underlying generative process in videos, whose product is the motion as a temporal transition of textures and appearances. We postulate that by encoding this underlying generative process in a general and action class-agnostic manner, we can arrive at long-term, robust motion representation relevant not only to action classification, but other temporal modelling tasks. In this work we propose to aggregate the motion present in the frames of a video into single images, which we coin *dynamic images*. We show that *dynamic images* are novel, powerful, efficient, and yet

- *Hakan Bilen and Andrea Vedaldi are with the University of Oxford. E-mail: {hbilen,vedaldi}@robots.ox.ac.uk*
- *Basura Fernando is with the ACRV, Research School of Engineering, The Australian National University, ACT 2601, Australia. E-mail: basura.fernando@anu.edu.au*
- *Efstratios Gavves is with the QUVA Lab, University of Amsterdam. E-mail: egavves@uva.nl*

1. From left to right and top to bottom: “blowing hair dry”, “band marching”, “balancing on beam”, “golf swing”, “fencing”, “playing the cello”, “horse racing”, “doing push-ups”, “drumming”.

simple representation of videos in the context of deep learning.

We consider the problem of recognizing human actions in short video sequences. Recent works such as [10], [11], [12], [16], [42] pointed out that long term dynamics and temporal patterns are very important cues for the recognition of actions. However, representing complex long term dynamics is challenging, particularly if one seeks compact representations that can be processed efficiently. Several efficient representations of long term dynamics have been obtained by *temporal pooling* of image features in a video. Temporal pooling has been done using temporal templates [3] or using ranking functions for video frames [11] or subvideos [16] or by more traditional pooling operators [42].

We propose a new temporal pooling operator which is simple, efficient, compact, and powerful in the context of neural networks. Since a CNN provides a whole hierarchy of image representations, one for each intermediate layer, the first question is where temporal pooling should take place. For example, one could use a method such as rank pooling [11] to pool the output of the fully-connected layers of a standard CNN architecture pre-trained on still images and applied to individual frames. A downside of this solution is that the CNN itself is unaware of the lower level dynamics of the video. Alternatively, one can model the dynamics of the response of some intermediate network layer. In this case, the lower layers are still computed from individual frames, but the upper layers can reason about the overall dynamics in the video. An extreme version of this idea is to capture the video dynamics *directly at the level of the image pixels*, considering those as the first layer of the architecture.

Our first contribution (section 3) is to introduce the notion of a *dynamic image*, *i.e.*a RGB still image that summarizes, in a compressed format, the gist of the dynamics in a whole video (sub)sequence (fig. 1). The dynamic image of a video can be viewed as the generative process that produced the motion in the video, since it can produce novel video segments by reordering by perturbations subsets of the video frames. The dynamic image is obtained as a ranking classifier that, similarly to [11], [12], sorts video frames temporally; the difference is that we compute this classifier directly at the level of the image pixels instead of using an intermediate feature representation.

There are three keys advantages to this idea. First, the new RGB image can be processed by a CNN architecture which is nearly identical to architectures used for still images, while still capturing the long-term dynamics in a video that relate to the long term dynamics therein. This enables use of a standard CNN to learn suitable "dynamic" features from the videos. The second advantage of this method is its remarkable efficiency: the extraction of the dynamic image is extremely simple and efficient. It allows us to reduce the video classification task to an image classification task which is solved using a standard CNN architecture. The third advantage is the compression factor, as a whole video is summarized by an amount of data equivalent to a single frame.

As a second contribution we provide a fast approximation of the ranker in the construction of the dynamic image, which we refer to as *approximate rank pooling* (ARP). This fast ranking approximation replaces the learning-to-rank problem with a linear machine that represents the video as a weighted summation of the video frames. The weights of approximate rank-pooling are pre-determined and shared across all videos of the same length, which makes this an extremely efficient operation. As ARP is a linear function of video frames, we consider ARP as a temporal pooling operator over any σ-ordered, N-length sequences, $\sigma = [I_{\sigma(1)}, \ldots, I_{\sigma(N)}]$. Unlike other commonly used temporal pooling operators like max- or average-pooling that are orderless, and therefore, time/sequence invariant, ARP is sensitive to the permutation order σ. To the best of our knowledge, we are the first to propose a *temporal* pooling layer for neural network architectures that aggregates order-specific information of the samples within a batch. We show that ARP can be seamlessly integrated into the end-to-end training of a CNN for video data. We also show that in a similar manner it is possible to apply the concept of dynamic image via approximate rank pooling to the intermediate layers of a CNN representation, allowing for a multi-scale temporal pooling in a neural network.

As a third contribution, we propose a static/dynamic neural network architecture (section 4) which can perform end-to-end training from videos combining both static appearance information from still frames, as well as short and long term dynamics from the whole video. We show that these static/dynamic architecture allows for efficient and accurate classification of actions in videos, obtaining the state-of-the-art in standard benchmarks in the area (section 5).

This work is an extension of a prior publication [2], compared to which we list the following differences:

- a more extensive overview and comparison of the related literature,
- more detailed formulation of the proposed pooling operations in section 3.3,
- a new temporal pooling method, *parametric pooling* that can learn to pool, easily be integrated into a deep neural network and can be trained end-to-end in section 3.4,
- a thorough evaluation of the proposed ARP when it is applied to intermediate layers of a CNN instead of RGB pixels which further improves action classification performance in different benchmarks in section 5,
- a detailed analysis of various design choices such as temporal window length, sampling rate, temporal pooling strategies in section 5.

The remainder of the paper is as follows, section 2 gives an extensive overview of the related work. section 3 describes formulations of the proposed pooling methods. Section 4 proposes different deep neural network architectures and explains how the proposed pooling operators can be integrated into standard CNN models. Section 5 provides a rigorous analysis of design choices and evaluates the performance of the proposed method in standard human action recognition benchmarks. Section 6 concludes the paper and discusses possible future directions.

2 Related Work

Videos as stack of still images: Existing video representations can be grouped into two categories. The first one, which comprises the majority of the literature on video processing, action and event classification, be it with shallow [11], [12], [33], [54] or deep architectures [34], [45], consider videos either as a stream of still images [34] or as a short and smooth transition between similar frames [45]. It is shown that considering the video as bag of static frames perform reasonably well [34], as the context of an action typically correlates with the action itself (*e.g.*, "playing basketball" takes place usually in a basketball court).

Videos as spatio-temporal volumes: The second category considers videos as a 3D dimensional volume rather than a collection of 2D images. In pre-deep era, several authors [38], [40], [44] propose learning spatio-temporal templates from the spatio-temporal volumes. More recent work [19], [51] extends spatial 2D CNNs to a third, temporal dimension [19] substituting 2D filters with 3D ones. While the extension brings a more natural feature representation for videos, it leads to a significant increase in the number of parameters to learn and thus requires more training data. Tran *et al.* [51] show that 3D convolutional networks perform well in the presence of large amount of annotated videos. Simonyan *et al.* [45] show an alternative way of exploiting spatio-temporal coherence in videos by training a deep neural networks on pre-computed optical flow rather than raw RGB frames and report significant improvement over previous state-of-the-art. Similarly, [14] uses action tubes to to fit a double stream appearance and motion based neural network that captures the movement of the actor.

We can also distinguish two architectural choices in the construction of video CNNs. The first choice is to provide as input to the CNN a sub-video of fixed length, packing a short sequence of video frames into an array of images. The advantage of this technique is that they allow using simple modifications of standard CNN architectures (*e.g.* [23]) by swapping 3D filters for the 2D ones. Examples of such techniques include [19] and [45].

Short and long-term dynamics: While the aforementioned methods successfully capture the local changes within a small time window, they cannot capture longer-term motion patterns associated with certain actions. An alternative solution is to consider a second family of architectures based on recurrent neural networks (RNNs) [7], [50]. RNNs typically consider memory cells [17], which are sensitive to both short as well as longer term patterns. RNNs parse the video frames sequentially and encode the frame-level information in their memory. In [7] LSTMs are used together with convolutional neural network activations to either output an action label or a video description. In [50] an autoencoder-like LSTM architecture is proposed such that either the current frame or the next frame is accurately reconstructed. Finally, the authors of [60] propose an LSTM with a temporal attention model for densely labelling video frames.

Many of the ideas in video CNNs originated in earlier architectures that used hand-crafted features. For example, the authors of [27], [53], [54] have shown that local motion patterns in short frame sequences can capture very well the short temporal structures in actions. The rank pooling idea, on which our dynamic images are based, was proposed in [11], [12] using hand-crafted representation of the frames.

Multi-stream networks: Our static/dynamic CNN uses a multi-stream architecture. Multiple streams have been used in a variety of different contexts. Examples include Siamese architectures for learning metrics for face identification [5] of for unsupervised training of CNNs [6]. Simonyan *et al.* [45] use two streams to encode respectively static frames and optical flow frames in action recognition. The authors of [31] propose a dual loss neural network was proposed, where coarse and fine outputs are jointly optimized. A difference of our model compared to these is that we branch off two streams at arbitrary location in the network, either at the input, at the level of the convolutional layers, or at the level of the fully-connected layers.

Motion information: Motion is a rich source of information for recognizing human actions and kinematic features design is heavily studied in the context of human action recognition. In this regard, techniques such as motion summary methods [3], optical flow [1], [9], [21], [43], and dynamic textures [22] are used to capture motion.

Our work is also related to early work on motion summary techniques such as motion energy image (MEI) and motion history image (MHI) [3]. Given an image sequence, the binary MEIs highlight regions in the image where any form of motion was present. To construct MEIs, the summation of the square of consecutive image differences is used as a robust spatial motion-distribution signal. To encode the motion of an image sequence, the motion history image (MHI) are used. In an MHI, pixel intensity is a function of the motion history at that location, where brighter values correspond to more recent motion. As a matter of fact, we compare the proposed method to MHI quantitatively and qualitatively in section 5.

Optical-flow based methods estimate the optical-flow between successive frames and then summarize the motion using principle components [1], [43]. In some instances the optical flow is computed on sub-volumes of the whole video using integral videos [21], or surrounding the central motion [9]. However, normally, the optical-flow, provides only the local dynamics and aggregation of local motion is performed using simple summarization methods.

Spatio-temporal dynamics: Dynamic texture [8] uses auto-regressive moving average process which estimates the parameters of the model using sequence data. Dynamic textures methods evolved from techniques originally designed for recognizing textures in 2D images [8], where they were extended to time-varying dynamic textures [22] for sequence recognition tasks. The Local Binary Patterns (LBP) [35], for example, use short binary strings to encode the micro-texture centered around each pixel. A whole 2D image is represented by the frequencies of these binary strings. In [22], [62] the LBP descriptor was extended to 3D video data and successfully applied to facial expression recognition tasks. Subspace-based methods are used in [28]. These methods captured some time-varying information for sequence classification tasks.

Even though these techniques [1], [3], [22], [43] provides a solution to capture motion of video sequences, none of them use *learning strategy* based on optimization to summarize the motion dynamics of video sequence as our method. Moreover, we are the first to use a motion summary images to train CNNs for human action recognition. Our motion summary concept is based on rank pooling and can be applied at different levels of CNN architecture.

Learning to rank videos: More recently the rank pooling [12] method is extended in [10] to increase the capacity of rank pooling using a hierarchical approach. In [13], an end-to-end video representation learning method is proposed using CNNs and rank-pooling. Our method is also based on rank pooling [12], however, compared [12] we learn the video representations end-to-end while being more efficient than [13]. The end-to-end video classification method [13] relies on computing the exact gradient of the rank pooling operator where as we argue that it is a good compromise to approximate the gradient of the rank pooling function considering exact method of [13] has to rely on bi-level optimization [15]. In this paper, we only take the first gradient step of the rank pooling operator which allows us to obtain a reasonable solution to the initial optimization problem. To the best of our knowledge such effective optimization trick has not been tried before in the context of CNN-based learning.

The impact of objects in action recognition is studied in [18]. Fisher vector [37] and VLAD descriptor based action recognition has shown promising results along with hand-crafted features [26],

[29], [36]. Attributes [30], action-parts [39], [56], hierarchy [25], [48], [57], trajectory pooled deep features [55], human pose [4], [59] and the context [32] also have been used for action recognition. Overall, the good practices in action recognition is described in [58].

3 Dynamic Images

In this section we introduce the concept of *dynamic image*, which is a standard RGB image that summarizes the appearance and dynamics of a whole video sequence (section 3.1). Then, we show how dynamic images can be used to train dynamic-aware CNNs for action recognition in videos (section 3.2). We propose a fast approximation to accelerate the computation of dynamic images (section 3.3). Finally, we parametrize the derived approximation and show that we can learn its weights jointly during training (section 3.4).

3.1 Constructing dynamic images

While CNNs can learn automatically powerful data representations, they can only operate within the confines of a specific hand-crafted architecture. In designing a CNN for video data, in particular, it is necessary to think of how the video information should be presented to the CNN. As discussed in section 2, standard solutions include encoding sub-videos of a fixed duration as multi-dimensional arrays or using recurrent architectures. Here we propose an alternative and more efficient approach in which the video content is summarized by a single still image which can then be processed by a standard CNN architecture such as AlexNet [23].

Summarizing the video content in a single still image may seem difficult. In particular, it is not clear how image pixels, which already contain appearance information in the video frames, could be overloaded to reflect dynamic information as well, and in particular the long-term dynamics that are important in action recognition.

We show here that the construction of Fernando *et al.* [11], [12] can be used to obtain exactly such an image. The idea of their work is to represent a video as a *ranking function* for its frames $I_1, \dots, I_T$. In more detail, let $\psi(I_t) \in \mathbb{R}^d$ be a representation or feature vector extracted from each individual frame I_t in the video. Let $V_t = \frac{1}{t}\sum_{\tau=1}^{t} \psi(I_\tau)$ be time average of these features up to time t. The ranking function associates to each time t a score $S(t|\mathbf{d}) = \langle \mathbf{d}, V_t \rangle$, where $\mathbf{d} \in \mathbb{R}^d$ is a vector of parameters. The function parameters $\mathbf{d}$ are learned so that the scores reflect the rank of the frames in the video. Therefore, later times are associated with larger scores, *i.e.* $\forall \{q,t\}$ s.t. $q > t \implies S(q|\mathbf{d}) > S(t|\mathbf{d})$. Learning $\mathbf{d}$ is posed as a convex optimization problem using the RankSVM [47] formulation:

$$\begin{aligned} \mathbf{d}^* &= \rho(I_1, \dots, I_T; \psi) = \operatorname*{argmin}_{\mathbf{d}} E(\mathbf{d}), \\ E(\mathbf{d}) &= \frac{\lambda}{2}\|\mathbf{d}\|^2 + \\ & \frac{2}{T(T-1)} \times \sum_{q>t} \max\{0, 1 - S(q|\mathbf{d}) + S(t|\mathbf{d})\}. \end{aligned} \tag{1}$$

The first term in this objective function is the usual quadratic regularizer used in SVMs. The second term is a hinge-loss soft-counting how many pairs $q > t$ are *incorrectly* ranked by the scoring function. Note in particular that a pair is considered correctly ranked only if scores are separated by at least a unit margin, i.e. $S(q|\mathbf{d}) > S(t|\mathbf{d}) + 1$.

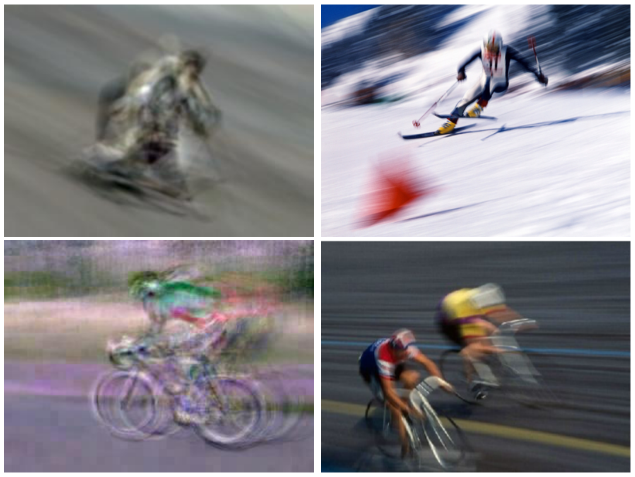


Fig. 2: Left column: dynamic images. Right column: motion blur. Although fundamentally different both methodologically, as well as in terms of applications, they both seem to capture time in a similar manner.

The optimizer to eq. (1) is written as a function $\rho(I_1, \dots, I_T; \psi)$ that maps a sequence of T video frames to a single vector $\mathbf{d}^*$. Since this vector contains enough information to rank all the frames in the video, it aggregates information from all of them and can be used as a video descriptor. The process of constructing $\mathbf{d}^*$ from a sequence of video frames is known as *rank pooling* [12].

In [11] the map $\psi(\cdot)$ used in this construction is set to be the Fisher Vector coding of a number of local features (HOG, HOF, MBH, TRJ) extracted from individual video frames. Here, we propose to apply rank pooling *directly to the RGB image pixels* instead. While this idea is simple, in the next several sections we will show that it has remarkable advantages.

The $\psi(I_t)$ is now an operator that stacks the RGB components of each pixel in image I_t on a large vector. Alternatively, $\psi(I_t)$ may incorporate a simple component-wise non-linearity, such as the square root function $\sqrt{\cdot}$ (which corresponds to using the Hellinger's kernel in the SVM). In all cases, the descriptor $\mathbf{d}^*$ is a real vector that *has the same number of elements as a single video frame*. Therefore, $\mathbf{d}^*$ can be interpreted as standard RGB image. Furthermore, since this image is obtained by rank pooling the video frames, it summarizes information from the whole video sequence.

A few examples of dynamic images are shown in fig. 1. Several observations can be made. First, interestingly the dynamic images tend to focus mainly on the acting objects, such as humans or other animals such as horses in the "horse racing" action, or objects such as drums in the "drumming" action. On the contrary, background pixels and background motion patterns tend to be averaged away. Hence, the pixels in the dynamic image appear to focus on the identity and motion of the salient actors in videos, indicating that they may contain the required information to perform action recognition.

Second, we observe that dynamic images behave differently for actions of different speeds. For slow actions, like "blowing hair dry" in the first row of fig. 1, the motion seems to be dragged over many frames. For faster actions, such as "golf swing" in the second row of fig. 1, the dynamic image reflects key steps in the action such as preparing to swing and stopping after swinging. For longer term actions such as "horse riding" in the third row of fig. 1, the dynamic image reflects different parts of the video; for instance, the rails that appear as a secondary motion contributor are superimposed on top of the horses and the jockeys who are the main actors. Such observations were also made in [12].

Last, it is interesting to note that dynamic images are reminiscent of some other imaging effects that convey motion and time, such as *motion blur* or *panning*, an analogy is illustrated in fig. 2. While motion blur captures the time and motion by integrating over subsequent pixel intensities defined by the camera shutter speed, dynamic images capture the time by integrating and reordering the pixel intensities over time within a video.

3.2 Using dynamic images

Given that the dynamic images are in the format of standard RGB images, they can be used to apply any method for still image analysis, and in particular any state-of-the-art CNN, to the analysis of video. In particular, we experiment with two usage scenarios.

Single Dynamic Image (SDI). In the first scenario, a dynamic image summarizes an entire video sequence. By training a CNN on top of such dynamic images, we implicitly capture the temporal patterns contained in the video. However, since the CNN is still applied to images, we can start from a CNN *pre-trained for still image recognition*, such as AlexNet pre-trained on the ImageNet ILSVRC data, and fine-tune it on a dataset of dynamic images. Fine-tuning allows the CNN to learn features that capture the video dynamics without the need to train the architecture from scratch. This is an important benefit of our method because training large CNNs require millions of data samples which may be difficult to obtain for videos.

Multiple Dynamic Images (MDI). While fine-tuning does not require as much annotated data as training a CNN from scratch, the domain gap between natural and dynamic images is sufficiently large that an adequately large fine-tuning dataset of dynamic images may be appropriate. However, as noted above, in most cases there are only a few videos available for training. Use of MDI also provides robustness to abrupt motion.

In order to address this potential limitation, in the second scenario we propose to generate multiple dynamic images from each video by breaking it into segments. In particular, for each video we extract multiple partially-overlapping segments of duration τ and with stride s. In this manner, we create multiple video segments per video, essentially multiplying the dataset size by a factor of approximately T/s, where T is the average number of frames per video. This can also be seen as a data augmentation step, where instead of mirroring, cropping, or shearing images we simply take a subset of the video frames. From each of the new video segments, we can then compute a dynamic image to train the CNN, using as ground truth class information of each subsequence the class of the original video.

3.3 Fast dynamic image computation

Computing a dynamic image entails solving the optimization problem of eq. (1). While this is not particularly slow with modern solvers, in this section we propose an approximation to rank pooling which is much faster and works as well in practice. Later, this technique, which we call *approximate rank pooling* (ARP), will be critical in incorporating rank pooling in intermediate layers of a deep CNN and to allow back-prop training through it.

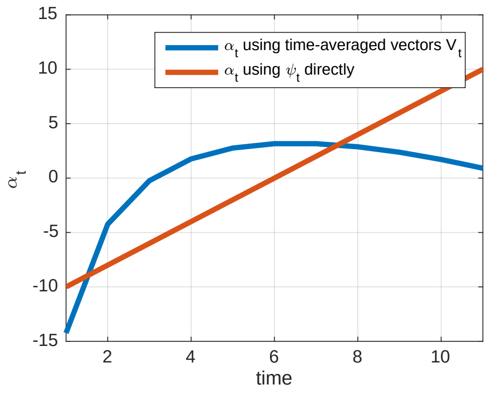


Fig. 3: The graph compares the approximated rank pooling weighting functions α_t (for $T = 11$ samples) of eq. (3) using time-averaged feature frames V_t to the variant eq. (2) that ranks directly the feature frames ψ_t as is.

The derivation of approximate rank pooling is based on the idea of considering the first step in a gradient-based optimization of eq. (1). Starting with $\mathbf{d} = \vec{0}$, the first approximated solution obtained by gradient descent is $\mathbf{d}^* = \vec{0} - \eta\nabla E(\mathbf{d})|_{\mathbf{d}=\vec{0}} \propto -\nabla E(\mathbf{d})|_{\mathbf{d}=\vec{0}}$ for any $\eta > 0$, where

$$\begin{aligned}\nabla E(\vec{0}) &\propto \sum_{q>t} \nabla \max\{0, 1 - S(q|\mathbf{d}) + S(t|\mathbf{d})\}|_{\mathbf{d}=\vec{0}} \\ &= \sum_{q>t} \nabla\langle \mathbf{d}, V_t - V_q\rangle = \sum_{q>t} V_t - V_q.\end{aligned}$$

We can further expand $\mathbf{d}^*$ as follows

$$\mathbf{d}^* \propto \sum_{q>t} V_q - V_t = \sum_{t=1}^{T} \beta_t V_t$$

where β_t are scalar coefficients. By expanding the sum

$$\begin{aligned}\sum_{q>t} V_q - V_t =&(V_2 - V_1) \\ &+(V_3 - V_1) + (V_3 - V_2) \\ &\vdots \\ &+(V_T - V_1) + (V_T - V_2) + \ldots + (V_T - V_{T-1}).\end{aligned}$$

one can simply see that each V_t with positive or negative sign occurs $(t-1)$ and $(T-t)$ times respectively. Now we can write β_t in terms of time and video length:

$$\beta_t = (t-1) - (T-t) = 2t - T - 1. \tag{2}$$

The time average vectors V_t can be written in terms of feature vectors ψ_t and $\mathbf{d}^*$ can be written as a linear combination of ψ_t

$$\mathbf{d}^* \propto \beta_t V_t = \alpha_t \psi(I_t)$$

where the coefficients α_t are given by

$$\alpha_t = 2(T-t+1) - (T+1)(H_T - H_{t-1}), \tag{3}$$

where $H_t = \sum_{i=1}^{t} 1/t$ is the t-th Harmonic number and $H_0 = 0$. The α_t coefficients can be derived from the observation that each

ψ_t occurs $\sum_{i=t}^{T} \beta_i H_i$ times in the sum. Hence the rank pooling operator reduces to

$$\hat{\rho}(I_1, \ldots, I_T; \psi) = \sum_{t=1}^{T} \alpha_t \psi(I_t). \quad (4)$$

which is a weighted combination of the data points ($\psi(I_t)$). In particular, the dynamic image computation reduces to accumulating the video frames after pre-multiplying them by α_t. The function α_t is illustrated in fig. 3.

An alternative construction of the rank pooling does not compute the intermediate average features $V_t = (1/t)\sum_{q=1}^{T} \psi(I_q)$, but uses directly individual video features $\psi(I_t)$ in the definition of the ranking scores (1). In this case, the derivation above results in a weighting function of the type

$$\alpha_t = 2t - T - 1 \quad (5)$$

which is linear in t. The two scoring functions eq. (3) and eq. (2) are compared in fig. 3 and in the experiments.

3.4 Parametric Pooling

As explained in the previous section, the rank pooling can be approximated as a linear combination of data points and α_t coefficients can be computed by using the eq. (4). While α_t coefficients provide a good approximation to a ranking function, these parameters can also be learnt in a discriminative manner such that they are optimised only for the target task (video classification) but not for ranking. We call this setting *parametric pooling*. Similarly to the rank pooling, parametric pooling takes a number of frames or feature tensors from a video as input and pools them into a single frame/feature tensor. In contrast to rank pooling, parametric pooling requires a fixed dimensional parameterization or fixed video length.

In practice, parametric pooling can be implemented as a sub-network which is composed of a number of convolutional layers followed by non-linear operators. In case of a single convolution and ReLU layers, it can be formulated as

$$\phi(I_1, \ldots, I_T; \psi) = \sigma(\sum_{t=1}^{T} \alpha_t \psi(I_t) + b_t)$$

where σ is the non-linear ReLU function and b_t is a bias parameter. As a matter of fact, the scalar multiplication and sum with (α_t, b_t) is implemented as a temporal fully-connected convolution over time. In section 5, we experiment with different numbers of convolutional layers and evaluate different the design choices.

4 Dynamic Maps Networks

In the previous section we have introduced the concept of dynamic image as a method to pool the information contained in a number of video frames into a single RGB image. Here, we notice that every layer of a CNN produces as output a *feature map* which, having a spatial structure similar to an image, can be used in place of video frames in this construction. We call the result of applying rank pooling to such features a *dynamic feature map*, or *dynamic map* in short. In the rest of the section we explain how to incorporate this construction as a rank-pooling layer in a CNN (section 4.1) and how to accelerate it significantly and perform back-propagation by using approximate rank pooling (section 4.2).

4.1 Dynamic maps

We illustrate three architecture designs for dynamic map networks fig. 4. In the case seen so far (fig. 4.(a)), rank pooling is applied at the level of the input RGB video frames, which we could think of as layer zero in the architecture. We call this architecture a **dynamic image network**. By contrast, a **dynamic map network** (fig. 4.(b)) moves rank pooling higher in the hierarchy, by applying one or more layers of feature computations to the individual feature frames and applying the same construction to the resulting feature maps.

In particular, let $\mathbf{a}_1^{(l-1)}, \ldots, \mathbf{a}_T^{(l-1)}$ denote the feature maps computed at the $l-1$ layers of the architecture, one for each of the T video frames. Then, we use the rank pooling equation (1) to aggregate these maps into a single dynamic map,

$$\mathbf{a}^{(l)} = \rho(\mathbf{a}_1^{(l-1)}, \ldots, \mathbf{a}_T^{(l-1)}). \quad (6)$$

Note that, compared to eq. (1), we dropped the term ψ; since networks are already learning feature maps, we set this term to the identity function. The dynamic image network is obtained by setting $l = 1$ in this construction.

Rank pooling layer (RankPool) & backpropagation. In order to train a CNN with rank pooling as an intermediate layer, it is necessary to compute the derivatives of eq. (6) for the backpropagation step. We can rewrite eq. (6) as a linear combination of the input data $V_1, \ldots, V_T$, namely

$$\mathbf{a}^{(l)} = \sum_{t=1}^{T} \beta_t(V_1, \ldots, V_T) V_t \quad (7)$$

In turn, V_t is the temporal average of the input features and is therefore a linear function $V_t(\mathbf{a}_1^{(l-1)}, \ldots, \mathbf{a}_t^{(l-1)})$. Substituting, we can rewrite $\mathbf{a}^{(l)}$ as

$$\mathbf{a}^{(l)} = \sum_{t=1}^{T} \alpha_t(\mathbf{a}_1^{(l-1)}, \ldots, \mathbf{a}_T^{(l-1)}) \mathbf{a}_t^{(l-1)}. \quad (8)$$

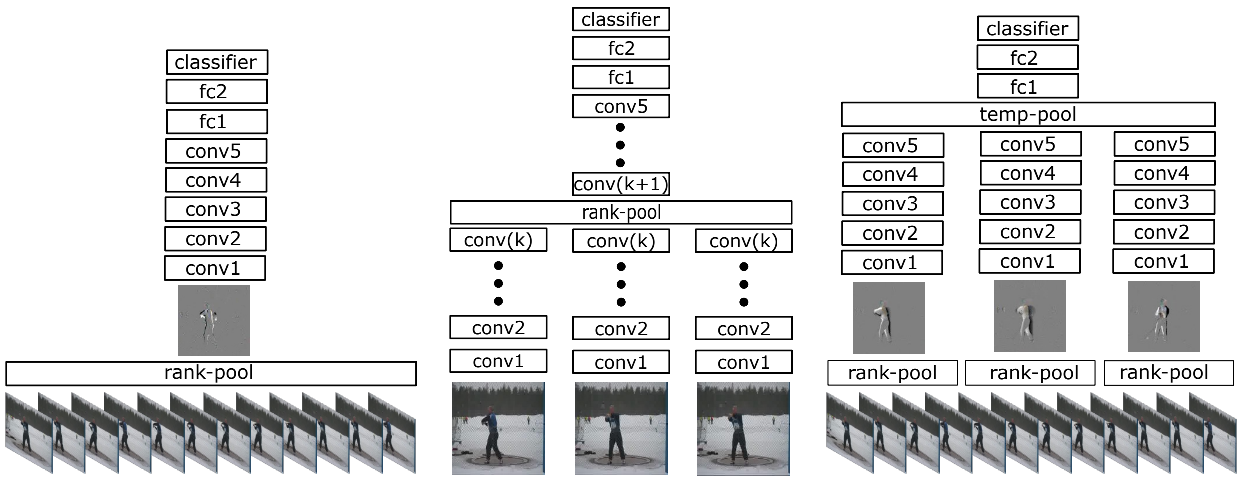
Unfortunately, we observe that due to the non-linear nature of the optimization problem of equation (1), the coefficients β_t, α_t depend on the data $\mathbf{a}_t^{(l-1)}$ themselves. Computing the gradient of $\mathbf{a}^{(l)}$ with respect to the per frame data points $\mathbf{a}_t^{(l-1)}$ is a challenging derivation. Hence, using dynamic maps and rank pooling directly as a layer in a CNN is not straightforward.

We note that the rank pooling layer (RankPool) constitutes a new type of portable convolutional network layer, just like a max-pooling or a ReLU layer. It can be used whenever dynamic information must be pooled across time.

4.2 Approximate dynamic maps.

Constructing the precise dynamic maps, or images, is in theory optimal, but not necessarily practical. On one hand computing the precise dynamic maps via an optimization is computationally inefficient. This is especially important in the context of CNNs, where efficient computations are extremely important for training on large datasets, and the optimization of eq. (6) would be slow compared to other components of the network. On the other hand, computing the gradients would be non trivial.

To this end we replace once again rank pooling with approximate rank pooling. With the approximate rank pooling we significantly accelerate the computations, even by a factor of 45 as



(a) Single Dynamic Image (SDI) (b) Single Dynamic Map (SDM) (c) Multiple Dynamic Image (MDI)

Fig. 4: Illustration of various dynamic image/map network architectures.

we show later in the experiments. Secondly, and more importantly, the approximate rank pooling is also a linear combination of frames, where the per frame coefficients are given by eq. (3). These coefficients are independent of the frame features V_t and $\psi(I_t)$. Hence, the derivative of the approximate rank pooling is much simpler and can be easily computed as the vectorized coefficients of eq. (3), namely

$$\frac{\partial \operatorname{vec} \mathbf{a}^{(l)}}{\partial (\operatorname{vec} \mathbf{a}_t^{(l-1)})^\top} = \alpha_t I \tag{9}$$

where I is the identity matrix. Interestingly, we would obtain the same expression for the derivative if α_t in eq. (8) would be constant and did not depend on the video frames.

We conclude that using approximate rank pooling in the context of CNNs is not only practical, but also necessary for the optimization through backpropagation.

4.3 Single/Multiple Dynamic Image/Map Networks

So far we assume a single dynamic image/map (SDI/SDM) per video (fig. 4.(a-b)). However, this may not be the optimal choice given videos at different lengths. Therefore we propose a new network architecture (fig. 4.(c)) that takes a sequence of frames from a video as input and splits them into fixed length subsequences and generate a dynamic image/map for each one subsequence. Finally after the last convolutional layer (*ie.*) a "temporal pool" layer pools the dynamic images/maps into one. We evaluate different temporal pooling choices in the experiments section. Please note that the multiple dynamic image (MDM) network can easily be adapted into a multiple dynamic map (MDM) by moving the approximate rank pooling layer at higher layers.

5 Experiments

In this part, we first give details for the video classification benchmarks (section 5.1) and experimental setup (section 5.2) used in the paper. Then we evaluate and compare SDI and MDI architectures, ARP, dynamic maps, parametric pooling in the following sections. Finally we show that our results are on par and complementary to the previous state-of-the-art.

5.1 Datasets

We benchmark the proposed models on two state-of-the-art datasets used for evaluating neural network based models for action recognition, namely *UCF101* [49] and HMDB51 [24]. Examples from these datasets are shown in Fig. 5.

UCF101. The UCF101 dataset [49] comprises of 101 human action categories, like "Apply Eye Makeup" and "Rock Climbing" and spans over $13,320$ videos. The videos are realistic and relatively clean. They contain little background clutter and a single action category. Also the videos are trimmed, thus almost all frames relate to the action in the video. The standard evaluation is average accuracy over three data splits provided by the authors.

HMDB51. The HMDB51 dataset [24] consists of 51 human action categories, such as "backhand flip" and "swing baseball bat" and spans over $6,766$ videos. The videos are realistic, downloaded from Youtube, each containing a single human action. This dataset is split in three parts and accuracy is averaged over all three parts, similar to *UCF101*.

5.2 Implementation details

Network: While more powerful networks exist, we choose here a good performing model which is at the same time reasonably efficient to train and evaluate, namely BVLC reference model (CaffeNet) [20]. This network is pre-trained on ImageNet [41] images and it is used as a starting point to train our dynamic image networks. We fine-tune all the layers with the learning rate to be 10^{-4}, decrease it to 10^{-5} after 15 epochs and stop the training after 20 epochs.

Dynamic Images: For generating dynamic images, we follow the pipeline suggested in Fernando *et al.* [11]: i) apply the non-linear operator, ii) square root the RGB pixel values ($\psi(\cdot)$), iii) use a time varying mean representation of $\sqrt{\cdot}$, iv) learn ranking hyperplanes for each channel, v) scale the computed dynamic images into $[0, 255]$ range again. The dynamic images are pre-computed and fed into a CNN as input in the experiments.

Approximate Dynamic Images/Maps: In contrast to dynamic images, approximate ones are computed on the fly as a part of CNN computation and RGB images are used as input. For

Fig. 5: Examples from the UCF101 (top row) and HMDB51 (bottom row) datasets. Both the datasets contain challenging real world videos for different human action categories.

Method	Accuracy (%)
Mean Image	52.6
Max Image	48.0
Dynamic Image	**57.2**

TABLE 1: Comparing the performance of various single image video representation methods on split-1 the UCF101 dataset in terms of mean class accuracy (%).

generating approximate dynamic images, we use an ARP layer (defined in Section 3.3) as the first layer of the network which is followed by a ℓ_2 normalization and scale layers to bring the output range between $[0, 255]$. Differently from approximate dynamic images, generating dynamic maps requires intermediate feature maps rather than RGB images. To do so, we place an ARP and ReLU layer after an existing ReLU layer and connect its output with the rest of the network.

Sharing code, models. We use MatConvNet toolbox [52] in all our experiments and share our code and models in the following address[2].

5.3 Rank pooling

First, we evaluate the "single image per video" setting, namely extracting a single dynamic image per video sequence. We compare the single dynamic image with the two most popular pooling methods, that is computing a single mean and max pooled image per video. For the mean and the max pooled images we first compute the single images per video offline, then apply the pooling operator. For dynamic images we use SVR [47]. After computing all three variants we fine-tune a CaffeNet model (see fig. 4.(a)) per variant. The fine tuning is performed on the first train/test split of the UCF101 dataset and results are shown in Table 1. We observe that dynamic images achieve the highest accuracy and conclude that the SDI model is better in pooling a single image representations from video as compared to the mean and max image models.

5.4 Approximating rank pooling

Next, we compare the rank pooling (RP) and approximate rank pooling (ARP) in terms of speed (frames per second) and pair-wise ranking accuracy, which is a common measure for evaluating learning-to-rank methods. We train on a subset of 10 videos that contain different actions and evaluate on a new set of 10 videos with the same type of actions respectively. We report results with the mean and the standard deviations in Table 2.

2. https://github.com/hbilen/dynamic-image-nets

Method	fps	Ranking Acc.	Classification Acc.
RP	131	99.5 ± 0.1	57.9
ARP	5920	96.5 ± 0.9	55.2

TABLE 2: Approximate rank pooling vs rank pooling in terms of speed, ranking accuracy (%) and classification accuracy (%)

We observe that RP takes one second to learn a dynamic image from 131 frame-length video in average which is approximately five times slower than a forward pass of these frames through our CNN model. ARP is approximately $45\times$ faster than RP, while obtaining similar ranking performance, while it brings a negligible additional cost to the CNN computation. Furthermore, we plot the score distributions for RP and ARP in Figure 6. We observe that their score profiles are also similar. We conclude that approximate rank pooling is a good approximation to rank pooling, while being two magnitudes faster as it involves no optimization.

Moreover, we also evaluate ARP in the first split of the UCF101 dataset and the single dynamic image representation with ARP obtains 55.2% accuracy which is 2.7 points lower than RP, however, it is much faster and outperforms "mean" and "max" pooling. Please note that when we refer to dynamic images from this point on, we assume that they are approximate rank pooled in the rest of the paper, unless it is explicitly stated.

5.5 Single vs multiple dynamic image networks

So far, we use a single dynamic image to represent each video and employ the network in Figure 4.(a). Here we show that splitting a video into multiple sub-sequences and encoding each of them as a dynamic image achieves better classification accuracy than using a single dynamic image. To do so, we use the multiple dynamic image (MDI) network in Figure 4.(c). After applying the ARP on the sub-sequences, we use an additional temporal max-pooling layer to merge the resulting convolutional features into one (see `temp-pool` layer in Figure 4.(c)). First we evaluate the influence of window size (τ) and stride (s) which determine the sub-sequence length and sampling frequency of frames respectively. While using video-length window size is equivalent

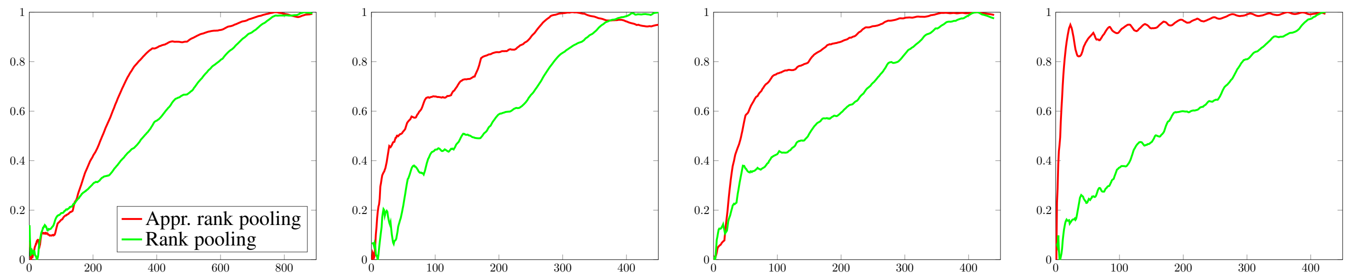


Fig. 6: Comparison between four score profiles of ranking functions for approximate rank pooling and rank pooling. Generally the approximate rank pooling follows the trend of rank pooling.

window	stride	Accuracy
5	3	63.9
10	3	66.9
10	6	**67.4**
20	6	60.7
20	12	58.9
video length	-	55.2

TABLE 3: Evaluating the effect of window size (τ) and stride (s) in terms of multi-class classification accuracy in the first split of the UCF101.

average	max	ARP
66.2	**68.3**	65.2

TABLE 4: Evaluating temporal pooling after the last convolutional layer in terms of multi-class classification accuracy.

to computing a single dynamic image per video, a single frame window corresponds to using RGB frames as input. In fact, we observe in Table 3 that using a mid-range window length (10) with a 40% overlap yields the best classification performance.

Furthermore we visualize several dynamic images which is computed over different window sizes and study effect of this factor in Figure 7. The top, middle and bottom rows show dynamic images with window lengths of 10, 50 and whole video length. As we use more and more frames to generate dynamic images, we observe that more pixels are activated to represent longer range of motion. For instance, in fig. 7.(c) and (d) we see more revolutions of hula-hoops and wheels respectively towards the bottom row. We notice that the dynamic image representation is unable to cope with complex and overlapping motion in fig. 7.(e), as the number of frames increases.

In addition to the rank pooling that generates dynamic images, we use a second temporal pooling layer to pool over the last convolutional features of multiple dynamic images. We investigate different pooling strategies (mean, max and approximate rank) for this layer and report the results in table 4. We observe that max pooling gives the best result. This can be explained with the fact that max pooling is invariant to the temporal position of the target action instance and does not require any alignment of start and end of action instances across different videos.

5.6 Dynamic images vs maps

So far we use ARP as the first layer in a CNN that is fed into the input frames. Here we apply the rank pooling after different

	dynamic image	dynamic map		
depth	conv0	conv1	conv2	conv3
UCF101	70.9	68.5	**73.3**	67.6
HMDB51	37.2	**38.0**	**38.0**	37.4

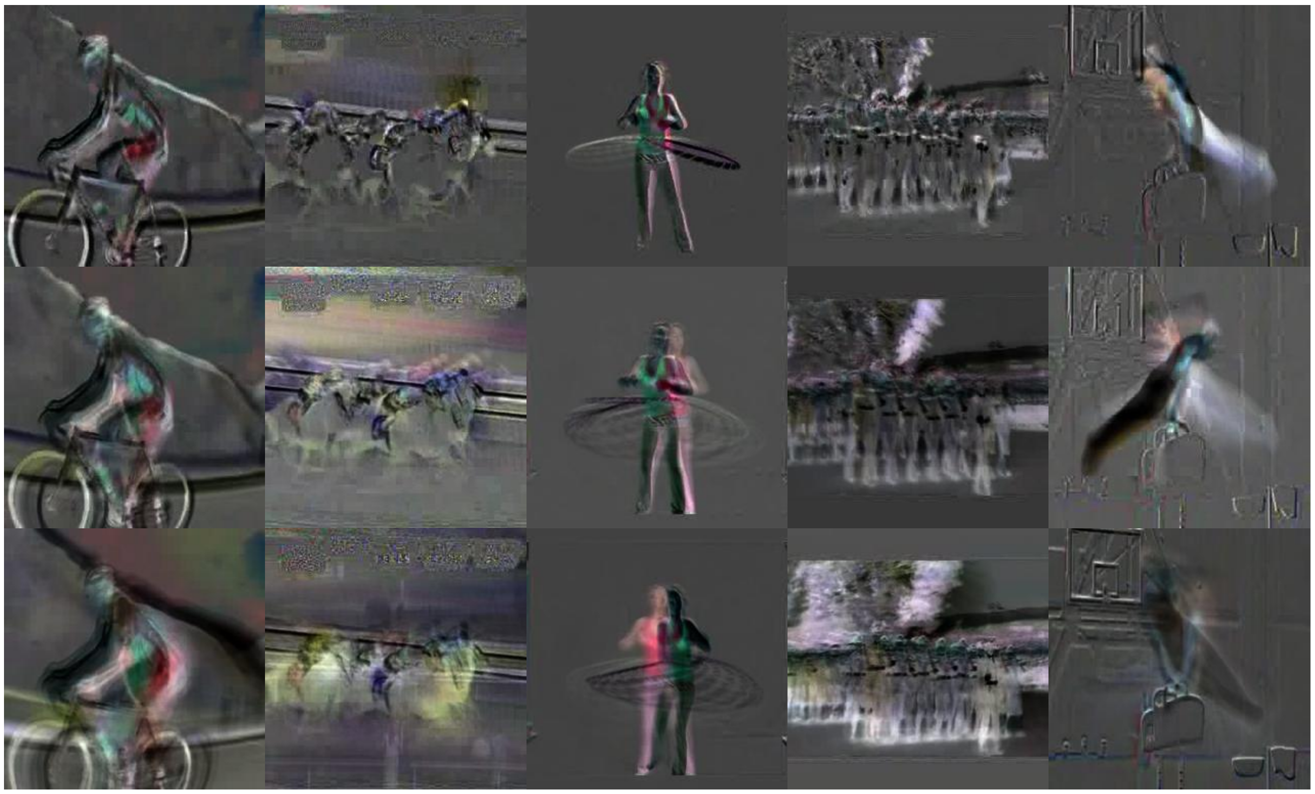
TABLE 5: Classification accuracy (%) for dynamic image and map networks at different depths on the UCF101 and HMDB51 datasets. "conv0" corresponds to placing the ARP as the first layer of network.

layers on feature maps rather than the input images. We report the mean class accuracy on the HMDB51 and UCF101 datasets in Table 5. "conv0" corresponds to approximate dynamic images where ARP is at the level of input images. Starting from "conv1", the rank pooling layer is positioned after the linear rectification layer (ReLU) following the specified convolutional layer and it is also followed by an additional ReLU. Each network is trained in end-to-end manner with multiple dynamic maps (see the network architecture in Figure 4.(b)). Please note that such an end-to-end training is possible, as we can compute the derivative dynamic images with respect to the input of ARP layer. We see that pooling after the second convolutional layer ("conv2") performs slightly better than "conv0" and "conv1' and the performance starts degrading after this layer. The degradation can be explained with the fact that the intermediate features in the earlier layers are more useful to express the motion and dynamics of a video.

5.7 Parametric Pooling

We also evaluate the performance of the proposed parametric pooling (see Section 3.4) on the UCF101 and HMDB51 datasets and denote it as "PPX" for two settings in Table 6. Similar to the dynamic map experiments, we apply the parametric pooling after the ReLU layer following the specified convolutional layer and parametric pooling is followed by an additional ReLU. The parameters of the networks are trained in end-to-end fashion with the pooling coefficients. The results ("parametric1") show that learning the pooling coefficients is beneficial and this method improves over the rank pooling.

The single layer parametric pooling (PP1) is implemented as a 10×1 temporal convolutional layer over 10 frames that belong to the same video. We fix the number of frames from each video to 10 frames (see table 3) and thus use a 10×1 convolution. PP1 can be extended to a mini multi-layer network by adding more temporal convolutions and non-linearities. As a matter of fact, we design a two layer parametric pooling with a 10×10 and following a 10×1 temporal convolution and denote it as "PP2" in table 6. While the



(a) Biking (b) HorseRace (c) HulaHoop (d) MilitaryParade (e) PommelHorse

Fig. 7: Visual analysis of different window sizes (τ) on dynamic images. The top, middle and bottom rows depict dynamic images for $\tau = 10$, $\tau = 50$ and $\tau =$(whole video length) respectively.

pooling	dataset	conv0	conv1	conv2	conv3
ARP	UCF101	70.9	68.5	73.3	67.6
	HMDB51	37.2	38.0	38.0	37.4
PP1	UCF101	67.6	73	71.9	70.9
	HMDB51	34.7	36.6	38.4	37.4
PP2	UCF101	68.5	73.9	70.0	69.1
	HMDB51	36.0	36.9	37.3	37.4

TABLE 6: Classification accuracy (%) for approximate rank pooling and parametric pooling at different depths on the UCF101 and HMDB51 datasets. “conv0” corresponds to placing the ARP or PPX as the first layer of network. PP1 and PP2 correspond to 1 and 2 layer sub-networks respectively.

two layer alternative performs comparable to the single layer one for “conv0”, “conv1”, it performs worse when it is applied on the raw video frames and after “conv3”.

5.8 Combining dynamic images with static images

Next, we evaluate how complementary dynamic image networks and static RGB frame networks are. For both networks we apply max pooling on the per video activations at pool5 layer and train each network separately. In test time we simply average the output scores of two networks and report the results in Table 7. As expected, we observe that static appearance information appears to be equally important to the dynamic appearance in the context of convolutional neural networks. A combination of the two, however, brings a noticeable 6% increase in accuracy. We conclude that the two representations are complementary to each other.

Method	HMDB51	UCF101
Static	36.0	68.5
Dynamic Image	37.2	70.9
Dynamic Image + Static	**39.7**	**76.5**

TABLE 7: Evaluating complementarity of dynamic images with static images.

Method	HMDB51	UCF101
TRJ [54]	60.0	86.0
Dynamic Image + Static + TRJ [54]	65.2	89.1

TABLE 8: Combining with trajectory features brings a noticeable increase in accuracy.

Furthermore, we investigate whether dynamic images are complementary to state-of-the-art features, like improved trajectories [54], relying on late fusion. Results are reported in Table 8. We obtain a significant improvement of 5.2% over trajectory features alone on HMDB51 dataset and 3.1% on UCF101 dataset.

5.9 State-of-the-art comparisons

We compare with the state-of-the-art techniques in UCF101 and HMDB51 in Table 9, where we make a distinction between deep and shallow architectures. Note that similar to us, almost all methods, be it shallow or deep, obtain their accuracies after combining their methods with improved trajectories [54] for optimal results.

Considering deep learning methods, our method performs on par and is only outperformed from [61]. [61] makes use of the very

	Method	HMDB51	UCF101
deep	This paper	65.2	89.1
	Zha et al. *[61]*	–	89.6
	Simonyan et al. *[45]*	59.4	88.0
	Yue-Hei-Ng et al. *[34]*	–	88.6
shallow	*Wu* et al. *[58]*	56.4	84.2
	Fernando et al. *[11]*	63.7	–
	Hoai et al. *[16]*	60.8	–
	Lan et al. *[26]*	65.4	89.1
	Peng et al. *[36]*	66.8	–

TABLE 9: Comparison with the state-of-the-art. Despite being a relatively simple representation, the proposed method is able to obtain results on par with the state state-of-the-art.

Class	Acc(%)	Class	Acc(%)
HandstandWalking	12.0	HorseRace	98.0
Shotput	26.1	IceDancing	96.7
Hammering	26.1	BasketballDunk	96.5
ShavingBeard	27.2	Billiards	96.0
CricketShot	29.1	Bowling	94.6

TABLE 10: Best and worst performing classes on the UCF101.

deep VGGnet [46], which is a more competitive network than that the Alexnet architecture we rely on. Hence a direct comparison is not possible. Compared to the shallow methods the proposed method is also competitive. We anticipate that combining the proposed dynamic images with sophisticated encodings [26], [36] will benefit the accuracies further.

We conclude that while being in the context of CNNs a simple and efficient video representation, dynamic images allow for state-of-the-art accuracy in action recognition.

5.10 Further analysis

Best/Worst Performing Classes: We first investigate the best and worst performing classes for the UCF101 dataset and report the accuracies of these categories in table 10 and show dynamic image examples from these categories in fig. 8.

Static vs Dynamic Networks: We, furthermore, perform a per class analysis between static rgb networks and MDI based networks. We list in Table 11 the top 5 classes based on the relative performances for each method. MDI performs better for "PullUps" and "PushUps", where motion is dominant and discriminating between motion patterns is important. RGB static models seems to work better on classes such as "CricketShot" and "Drumming", where context is already quite revealing. We conclude that dynamic images are useful for actions where there exist characteristic motion patterns and dynamics.

Motion History Images: An early work which also keeps record of spatio-temporal change in a video and represents a video by generating a frame is Motion History Images (MHI) [3]. An MHI is a scalar-valued image where more recently moving pixels are brighter and thus pixel intensity is a function of temporal history of motion at that point. We conduct a qualitative comparison between approximate dynamic images and MHIs in Figure 9. The figure shows a representative frame for a given video at the top row, corresponding MHI and approximate dynamic image in the middle and last row respectively. We first see that dynamic images provide more detailed representation of videos, as the range of intensity values are not limited with the number frames as in MHIs. Second dynamic images are more robust to moving viewpoint and background. Finally MHIs can only represent the motion gradient in object boundaries in contrast to dynamic images.

Class	Diff(%)	Class	Diff(%)
SoccerJuggling	+38.5	CricketShot	-47.9
CleanAndJerk	+36.4	Drumming	-25.6
PullUps	+32.1	PlayingPiano	-22.0
PushUps	+26.7	PlayingFlute	-21.4
PizzaTossing	+25.0	Fencing	-18.2

TABLE 11: Class by class comparison between static and MDI networks, where the difference in scores between two are reported. A positive difference is better for MDI, a negative difference better for RGB.

MHIs are originally designed to be used for action recognition task. The authors extract moment-based descriptors from a set of MHIs, learn a distance metric over each action category and perform classification using the computed metrics. As such a methodology is not competitive enough for the modern datasets, similar to our method, we compute multiple MHIs for a given a video, use them as input to the proposed MDI architecture and train it on the UCF101 dataset. For this representation, we obtain 61.1% accuracy in the first split of the UCF101 dataset which is 7.2% lower than what we obtain with dynamic images. This shows that the aforementioned advantages in the qualitative analysis transfers to better classification performance.

6 Conclusion

We present *dynamic images*, a powerful and new, yet simple video representation in the context of deep learning that summarizes videos into single images. As such, dynamic image networks can be training using existing CNN architectures allowing for end-to-end video representation learning for action recognition. We introduce a novel temporal pooling layer coined `Approximate-RankPool` which allows us to extend the principle of "dynamic images" to the CNN feature maps and back propagate the gradients through this temporal pooling layer. We also invent a novel parametric pooling layer which encapsulate dataset specific global temporal information through dynamic image principle. Experiments on state-of-the-art action recognition datasets demonstrate the descriptive power of dynamic images, despite their conceptual simplicity. A visual inspection outlines the richness of dynamic images in describing complex motion patterns as simple 2d images.

Dynamic images have some notable limitations as well. Even though they are good at capturing smooth dynamics, it is unable to handle abrupt motion in very complex video sequences. Second, the RGB colour space and the pixel dynamics are highly correlated in the spatial, temporal domain. Perhaps it is more effective to first obtain decorrelated spatio temporal volumes and then to model temporal dynamics. Third, dynamic images operates at single level of abstraction. In the future it is beneficial to explore dynamic encoding at multiple levels of abstraction by allowing the network to adapt according to the complexity of temporal data. We plan to address these limitations of dynamic images in future work.

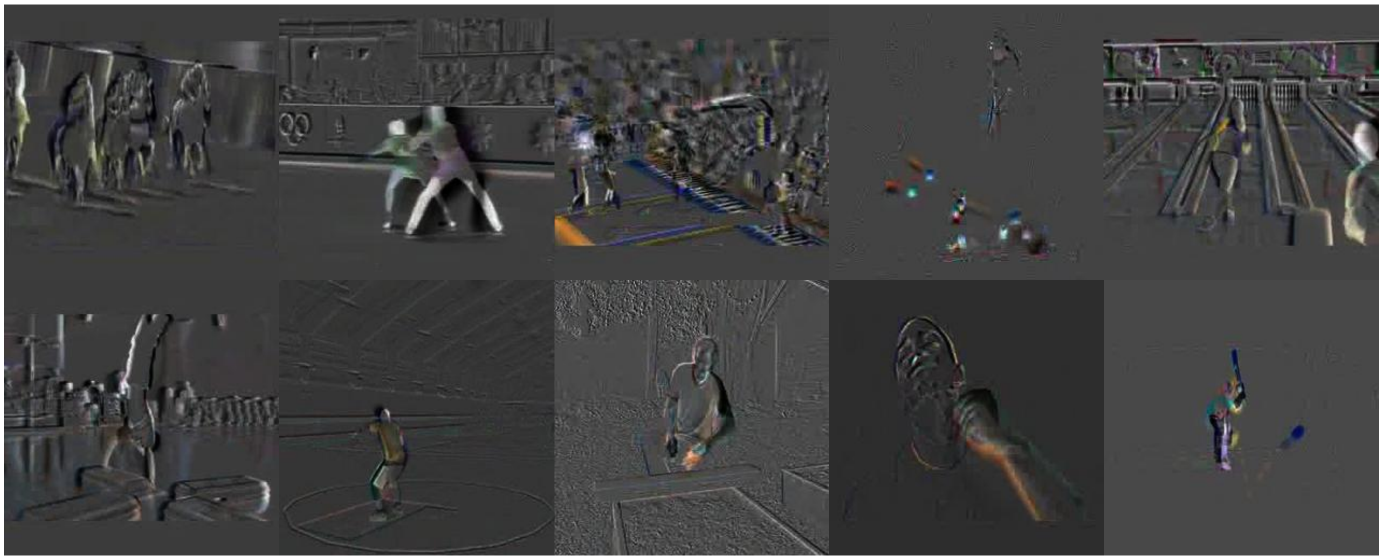


Fig. 8: Dynamic image examples from the best and worst performing classes in the top and bottom rows respectively on the UCF101.

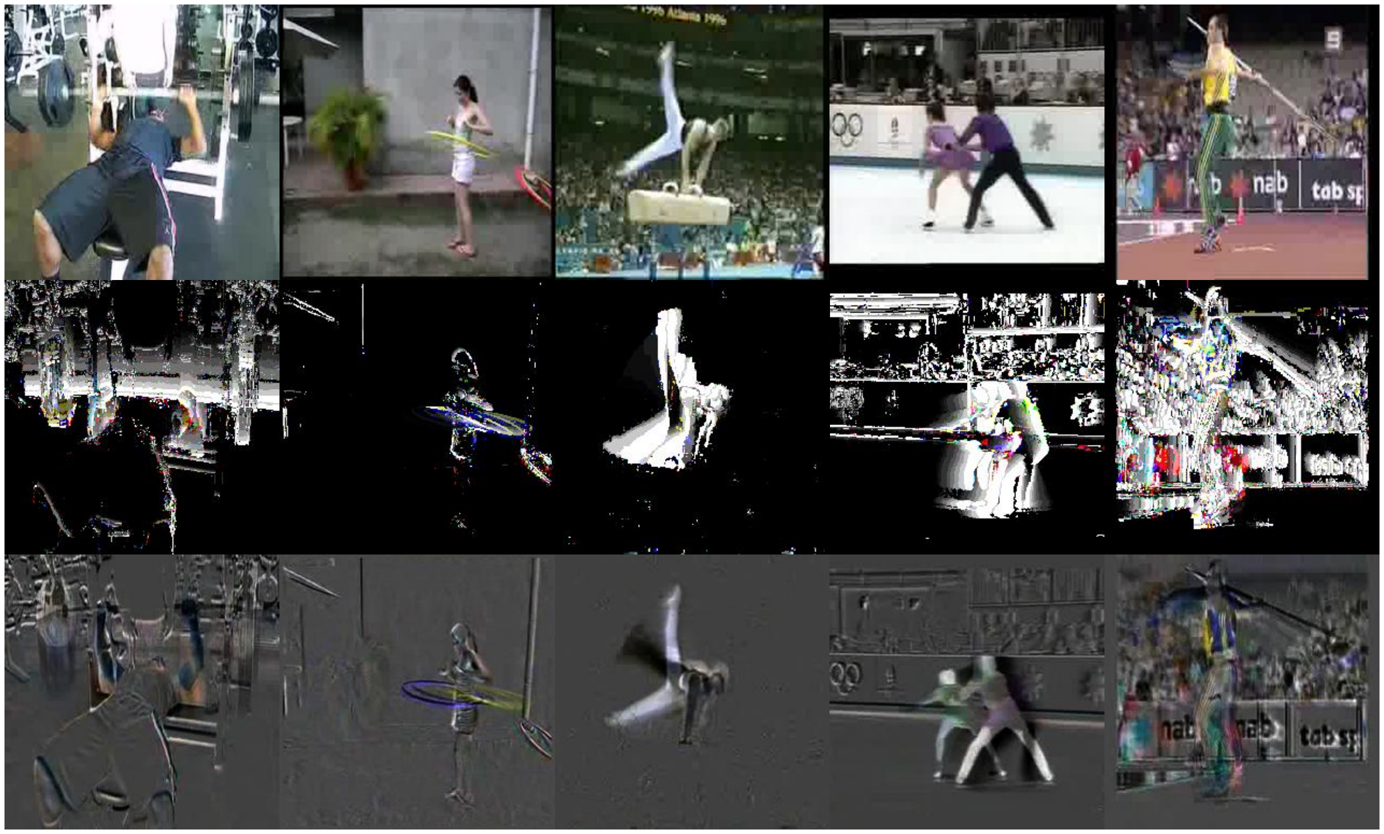


Fig. 9: Comparing Dynamic Images (DI) to Motion History Images (MHI) [3].The top row shows representative frames from different videos, middle and bottom rows depict MHI and DI of corresponding videos respectively. While both methods can represent the evolution of pixels along time, our method produces more interpretable images which are more robust to long-range and background motion.

Acknowledgments

This work acknowledges the support of the EPSRC grant EP/L024683/1, the ERC Starting Grant IDIU and the Australian Research Council Centre of Excellence for Robotic Vision (project number CE140100016).

References

[1] S. Ali and M. Shah, “Human action recognition in videos using kinematic features and multiple instance learning,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 2, pp. 288–303, 2010.

[2] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould, “Dynamic image networks for action recognition,” in *CVPR*, 2016.

[3] A. F. Bobick and J. W. Davis, “The recognition of human movement using temporal templates,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 3, pp. 257–267, 2001.

[4] G. Chéron, I. Laptev, and C. Schmid, “P-cnn: Pose-based cnn features for action recognition,” *arXiv preprint arXiv:1506.03607*, 2015.

[5] S. Chopra, R. Hadsell, and Y. LeCun, “Learning a similarity metric discriminatively, with application to face verification,” in *CVPR*, 2005.

[6] C. Doersch, A. Gupta, and A. A. Efros, “Unsupervised visual representation learning by context prediction,” in *CVPR*, 2015.

[7] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” in *arXiv preprint arXiv:1411.4389*, 2014.

[8] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto, “Dynamic textures,” *International Journal of Computer Vision*, vol. 51, no. 2, pp. 91–109, 2003.

[9] A. A. Efros, A. C. Berg, G. Mori, and J. Malik, “Recognizing action at a distance,” in *ICCV*. IEEE, 2003, pp. 726–733.

[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, J. Oramas, A. Ghodrati, and T. Tuytelaars, “Modeling video evolution for action recognition,” in *CVPR*, 2015.

[12] ——, “Rank pooling for action recognition,” *PAMI*, vol. PP, no. 99, pp. 1–1, 2016.

[13] B. Fernando and S. Gould, “Learning end-to-end video classification with rank-pooling,” in *ICML*, 2016.

[14] G. Gkioxari and J. Malik, “Finding action tubes,” in *CVPR*, June 2015.

[15] S. Gould, B. Fernando, A. Cherian, P. Anderson, R. S. Cruz, and E. Guo, “On differentiating parameterized argmin and argmax problems with application to bi-level optimization,” *arXiv preprint arXiv:1607.05447*, vol. 1, no. 1, p. 1, July 2016.

[16] M. Hoai and A. Zisserman, “Improving human action recognition using score distribution and ranking,” in *ACCV*, 2014.

[17] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] 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.

[19] S. Ji, W. Xu, M. Yang, and K. Yu, “3d convolutional neural networks for human action recognition,” *TPAMI*, 2013.

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” *arXiv preprint arXiv:1408.5093*, 2014.

[21] Y. Ke, R. Sukthankar, and M. Hebert, “Efficient visual event detection using volumetric features,” in *ICCV*, vol. 1. IEEE, 2005, pp. 166–173.

[22] V. Kellokumpu, G. Zhao, and M. Pietikäinen, “Human activity recognition using a dynamic texture based method.” in *BMVC*, vol. 1, 2008, p. 2.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012.

[24] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, “Hmdb: a large video database for human motion recognition,” in *ICCV*, 2011.

[25] T. Lan, Y. Zhu, A. R. Zamir, and S. Savarese, “Action recognition by hierarchical mid-level action elements,” in *ICCV*, 2015.

[26] Z.-Z. Lan, M. Lin, X. Li, A. G. Hauptmann, and B. Raj, “Beyond gaussian pyramid: Multi-skip feature stacking for action recognition.” in *CVPR*, 2015.

[27] I. Laptev, “On space-time interest points,” *IJCV*, vol. 64, pp. 107–123, 2005.

[28] 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.

[29] Y. Li, W. Li, V. Mahadevan, and N. Vasconcelos, “Vlad3: Encoding dynamics of deep features for action recognition,” in *CVPR*, 2016.

[30] J. Liu, B. Kuipers, and S. Savarese, “Recognizing human actions by attributes,” in *CVPR*, 2011.

[31] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” *CVPR*, 2015.

[32] M. Marszałek, I. Laptev, and C. Schmid, “Actions in context,” in *CVPR*, 2009.

[33] M. Mazloom, E. Gavves, and C. G. M. Snoek, “Conceptlets: Selective semantics for classifying video events,” *IEEE TMM*, December 2014.

[34] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,” in *CVPR*, 2015.

[35] T. Ojala, M. Pietikäinen, and T. Mäenpää, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *TPAMI*, vol. 24, no. 7, pp. 971–987, 2002.

[36] X. Peng, C. Zou, Y. Qiao, and Q. Peng, “Action recognition with stacked fisher vectors,” in *ECCV*, 2014.

[37] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, “Large-scale image retrieval with compressed fisher vectors,” in *CVPR*, 2010.

[38] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, “Bilinear classifiers for visual recognition,” in *Advances in neural information processing systems*, 2009, pp. 1482–1490.

[39] M. Raptis, I. Kokkinos, and S. Soatto, “Discovering discriminative action parts from mid-level video representations,” in *CVPR*, 2012.

[40] M. D. Rodriguez, J. Ahmed, and M. Shah, “Action mach a spatio-temporal maximum average correlation height filter for action recognition,” in *CVPR*, June 2008, pp. 1–8.

[41] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “Imagenet large scale visual recognition challenge,” *Int. J. Comput. Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[42] M. S. Ryoo, B. Rothrock, and L. Matthies, “Pooled motion features for first-person videos,” in *CVPR*, 2015.

[43] K. Schindler and L. Van Gool, “Action snippets: How many frames does human action recognition require?” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[44] E. Shechtman and M. Irani, “Space-time behavior based correlation,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1. IEEE, 2005, pp. 405–412.

[45] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” *CoRR*, vol. abs/1406.2199, pp. 1–8, 2014. [Online]. Available: http://arxiv.org/abs/1406.2199

[46] ——, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[47] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Statistics and computing*, vol. 14, pp. 199–222, 2004.

[48] Y. Song, L.-P. Morency, and R. Davis, “Action recognition by hierarchical sequence summarization,” in *CVPR*, 2013.

[49] K. Soomro, A. R. Zamir, and M. Shah, “Ucf101: A dataset of 101 human actions classes from videos in the wild,” *CoRR*, 2012.

[50] N. Srivastava, E. Mansimov, and R. Salakhudinov, “Unsupervised learning of video representations using lstms,” in *ICML*, 2015.

[51] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” *arXiv preprint arXiv:1412.0767*, 2014.

[52] A. Vedaldi and K. Lenc, “Matconvnet – convolutional neural networks for matlab,” in *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.

[53] 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.

[54] H. Wang and C. Schmid, “Action recognition with improved trajectories,” in *ICCV*, 2013.

[55] L. Wang, Y. Qiao, and X. Tang, “Action recognition with trajectory-pooled deep-convolutional descriptors,” in *CVPR*, 2015, pp. 4305–4314.

[56] Y. Wang and G. Mori, “Hidden part models for human action recognition: Probabilistic versus max margin,” *PAMI*, vol. 33, pp. 1310–1323, 2011.

[57] D. Wu and L. Shao, “Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition,” in *CVPR*, 2014.

[58] J. Wu, Y. Zhang, and W. Lin, “Towards good practices for action video encoding,” in *CVPR*, 2014.

[59] A. Yao, J. Gall, G. Fanelli, and L. Van Gool., “Does human action recognition benefit from pose estimation?” in *BMVC*, 2011.

[60] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei, "Every moment counts: Dense detailed labeling of actions in complex videos," *ArXiv e-prints*, 2015.

[61] S. Zha, F. Luisier, W. Andrews, N. Srivastava, and R. Salakhutdinov, "Exploiting image-trained cnn architectures for unconstrained video classification," in *BMVC*, 2015.

[62] G. Zhao and M. Pietikainen, "Dynamic texture recognition using local binary patterns with an application to facial expressions," *TPAMI*, vol. 29, no. 6, pp. 915–928, 2007.

Hakan Bilen is a post-doctoral research assistant in the Visual Geometry Group at the University of Oxford.

Basura Fernando is a research fellow with the The Australian Research Council Centre of Excellence for Robotic Vision based at the Australian National University.

Efstratios Gavves is an assistant professor with the University of Amsterdam in the Netherlands and Scientific Manager of the QUVA Deep Vision Lab.

Andrea Vedaldi is an associate professor in the Visual Geometry Group at the University of Oxford.