Computer Vision and Image Understanding 166 (2018) 41–50

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

VideoLSTM convolves, attends and flows for action recognition

Zhenyang Li[*,a], Kirill Gavrilyuk[a], Efstratios Gavves[a], Mihir Jain[a,b], Cees G.M. Snoek[a,b]

[a] QUVA Lab, University of Amsterdam, Science Park 904, Amsterdam, The Netherlands
[b] Qualcomm Research Netherlands, Science Park 400, Amsterdam, The Netherlands

ARTICLE INFO

Keywords:
Action recognition
Video representation
Attention
LSTM

ABSTRACT

We present *VideoLSTM* for end-to-end sequence learning of actions in video. Rather than adapting the video to the peculiarities of established recurrent or convolutional architectures, we adapt the architecture to fit the requirements of the video medium. Starting from the soft-Attention LSTM, VideoLSTM makes three novel contributions. First, video has a spatial layout. To exploit the spatial correlation we hardwire convolutions in the soft-Attention LSTM architecture. Second, motion not only informs us about the action content, but also guides better the attention towards the relevant spatio-temporal locations. We introduce motion-based attention. And finally, we demonstrate how the attention from VideoLSTM can be exploited for action localization by relying on the action class label and temporal attention smoothing. Experiments on UCF101, HMDB51 and THUMOS13 reveal the benefit of the video-specific adaptations of VideoLSTM in isolation as well as when integrated in a combined architecture. It compares favorably against other LSTM architectures for action classification and especially action localization.

1. Introduction

This paper strives to classify video actions like *shaving, biking* and *punch* in an end-to-end fashion. This challenging task is commonly addressed by learning a video's spatial appearance and motion flow with the aid of deep convolutional neural networks, before temporal pooling, *e.g.*, Simonyan and Zisserman (2014), Feichtenhofer et al. (2016b) and Wang et al. (2016). Different from these modern architectures, which all learn a global video representation without considering local spatial structures over time, we prefer to embed the spatio-temporal nature of the action into our representation learning, with the added benefit that we obtain the action's location for free.

We draw inspiration from saliency, a classical topic in computer vision (Itti et al., 1998) that was recently shown to emerge from recurrent neural network architectures as well, *e.g.*, Xu et al. (2015). The first to use such visual attention for action recognition in video is the work by Sharma et al. (2015); 2016). They extend the soft-Attention model of Xu et al. (2015), intended for image captioning, to action recognition in video. First, the appearance of individual video frames is encoded as a feature map tensor derived from the last convolutional layer of a GoogLeNet (Szegedy et al., 2015). Then, the model propagates the vectorized tensor through an LSTM (Hochreiter and Schmidhuber, 1997) and predicts the attention at the next frame in addition to the action label. Their model is very effective for action classification, despite the fact that it leverages image appearance only, while completely ignoring the motion inside a video. We propose a new LSTM architecture that integrates attention, appearance and motion of a video to arrive at better action classification and localization.

The standard LSTM (Hochreiter and Schmidhuber, 1997) treats all incoming data as a vector, even though it is well known that an image has a spatial correlation that is better preserved by convolutions than inner products (LeCun et al., 1998). Moreover, convolutional neural networks ensure better shift, scale and distortion invariance by leveraging local receptive fields, shared weights, and pooling, especially when they are very deep (He et al., 2015; Simonyan and Zisserman, 2015; Szegedy et al., 2015). Shi et al. (2015) introduced convolutional structures, rather than vectors, in both the input-to-state and state-to-state transitions of a standard LSTM for radar map forecasting. Ballas et al. (2016) stack multiple convolutional recurrent units in a deep hierarchical architecture. Their model achieves better performance than a single-layered recurrent neural network. In modeling videos, however, convolutions with a deep architecture alone do not suffice and attention must also be considered, as we will show in the experiments. Hence, to reckon the spatio-temporal nature of the video when using an LSTM, we must hardwire the LSTM network with convolutions and spatio-temporal attentions.

In this work we advocate and experimentally verify that in order to model videos accurately with LSTMs, we must adapt the LSTM architecture to fit the video medium, not vice versa. Such a model must address the video properties, *i.e.* appearance, motion and attention,

* Corresponding author.
E-mail address: zhenyangli@uva.nl (Z. Li).

https://doi.org/10.1016/j.cviu.2017.10.011
Received 12 May 2017; Received in revised form 4 October 2017; Accepted 18 October 2017
Available online 31 October 2017

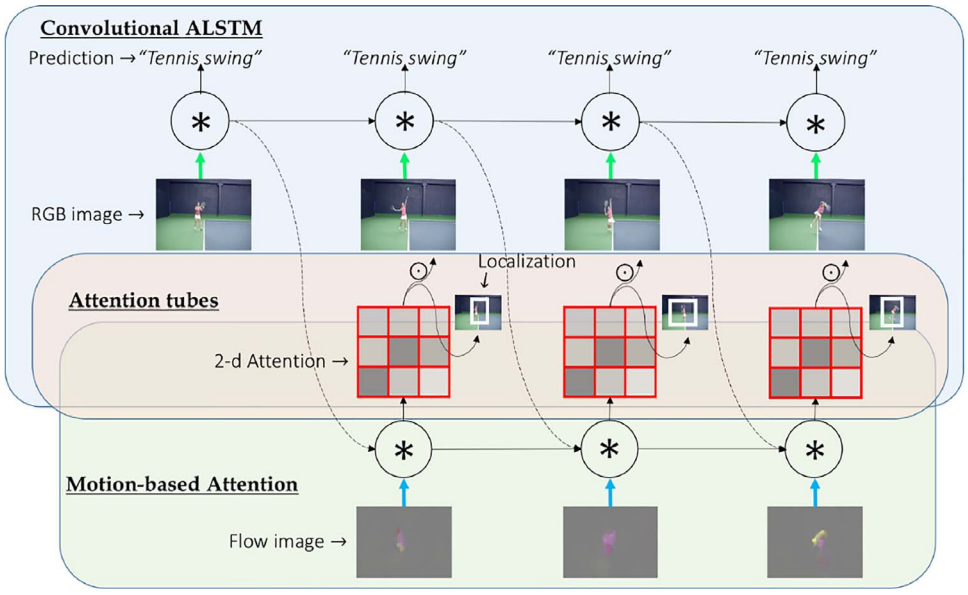

Fig. 1. The proposed *VideoLSTM* network. The blue container details the *Convolutional ALSTM* (Section 3.1), the green container details the motion-based attention (Section 3.2), while in the pale red container we rely on attention maps for action localization (Section 4). VideoLSTM learns to classify actions in an end-to-end manner and localizes the action from an action class label only. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

simultaneously and not in isolation. In fact, our experiments reveal that modeling a subset only of the properties brings little, if any, improvement, while a joint treatment results in consistent gains. We propose *VideoLSTM*, a recurrent neural network architecture intended for action classification and localization. VideoLSTM makes three contributions. First, VideoLSTM recognizes that video frames have a spatial layout. Hence the video frame encoding as well as the attention should be spatial too. We introduce convolutions to exploit the spatial correlation in images and adapt the soft-attention model to ensure the spatial structures are also preserved over time. Second, VideoLSTM incorporates motion-based attention, as motion not only informs us about the action content, but also better guides the attention towards relevant spatio-temporal locations. And finally, we demonstrate how the attention from VideoLSTM can be used for action localization by relying on just the action class label. We summarize our architecture in Fig. 1, and provide a synopsis of related work next.

2. Related work

The literature on action recognition in video is vast and too broad for us to cover here completely. We reckon and value the impact of traditional video representations, *e.g.*, Wang and Schmid (2013), Jain et al. (2013), Sadanand and Corso (2012), Peng et al. (2016); 2014),Fernando et al. (2015) and Lan et al. (2015), and recent mixtures of shallow and deep encodings *e.g.*, Wang et al. (2015a) and Jain et al. (2015). Here we focus on deep end-to-end alternatives that have recently become popular and powerful.

ConvNet architectures. One of the first attempts of using a deep learning architecture for action recognition is by Ji et al. (2013), who propose 3-d convolutional neural networks. A 3-d convolutional network is the natural extension of a standard 2-d convolutional network to cover the temporal domain as a third dimension. However, processing video frames directly significantly increases the learning complexity, as the filters need to model both appearance and motion variations. To compensate for the increased parameter complexity larger datasets are required. Indeed, Tran et al. (2015) demonstrated that 3-d convolutional networks trained on massive sport video datasets (Karpathy et al., 2014) yield significantly better accuracies.

To avoid having to deal with this added complexity of spatio-temporal convolutional filters, Simonyan and Zisserman (2014) proposed a two-stream architecture to learn 2-d filters for the optical flow and appearance variations independently. In order to capture longer temporal patterns, several frames are added as multiple consecutive channels as input to the network. To account for the lack of training data a multi-task setting is proposed, where the same network is optimized for two datasets simultaneously. Similar to Simonyan and Zisserman (2014) we also use optical flow to learn motion filters. Different from Simonyan and Zisserman (2014), however, we propose a more principled approach for learning frame-to-frame appearance and motion transitions via explicit recurrent temporal connections.

LSTM architectures. LSTM networks (Hochreiter and Schmidhuber, 1997) model sequential memories both in the long and in the short term, which makes them relevant for various sequential tasks (Donahue et al., 2015; Jia et al., 2015; Karpathy and Fei-Fei, 2015; Wu et al., 2015; Yue-Hei Ng et al., 2015). Where early adopters used traditional features as LSTM input (Baccouche et al., 2010; 2011), more recently both Yue-Hei Ng et al. (2015) and Donahue et al. (2015) propose LSTMs that explicitly model short snippets of ConvNet activations. Ng *et al.*demonstrate that an average fusion of LSTMs with appearance and flow input improves over improved dense trajectories (Wang and Schmid, 2013) and the two-stream approach of Simonyan and Zisserman (2014), be it that they pre-train their architecture on 1 million sports videos. Srivastava et al. (2015) also pre-train on hundreds of hours of sports video, but without using the video labels. Their representation demonstrates competitive results on the challenging UCF101 and HMDB51 datasets. We also rely on an LSTM architecture that combines appearance and flow for action recognition, but without the need for video pre-training to be competitive.

ALSTM architectures. Where the traditional LSTMs for action classification emphasize on modeling the temporal extent of a sequence with the use of spatial ConvNets, Attention-LSTMs (ALSTMs) also take into account spatial locality in the form of attention. While originally proposed for machine translation (Bahdanau et al., 2015), it was quickly recognized that a soft-Attention mechanism instead of a fixed-length vector is beneficial for vision problems as well (Xu et al., 2015). The attention turns the focus of the LSTM to particular image locations, such that the predictive capacity of the network is maximized. Sharma et al. (2015); 2016) proposed the first ALSTM for action classification, which proved to be an effective choice. However, by staying close to the soft-Attention architecture for image captioning by Xu et al. (2015), they ignore the motion inside a video. Moreover, rather than vectorizing an image, for vision it is more beneficial to rely on convolutional structures (LeCun et al., 1998; Shi et al., 2015). We incorporate convolutions and motion-based attention into the ALSTM, which is not only important for the action classification, but also results in better attention for action localization.

Action localization. In action localization two approaches are dominant. One approach first relies on unsupervised action proposals and then classifies each one with the aid of box annotations, *e.g.*, Jain et al. (2014) and van Gemert et al. (2015). The other is to rely on supervised detection of action regions using box annotations and then learning to link these over time, again using box supervision, *e.g.*, Saha et al. (2016) and Weinzaepfel et al. (2015). In general, the more box supervision, the better the action localization result. Recently, Mettes et al. (2016) showed how unsupervised proposals in combination with point supervision can be competitive with box supervision. Our end-to-end solution for action classification and localization learns from just the action class label, without the need for any box- or point-supervision, while still being able to exploit and predict the most salient action location.

3. VideoLSTM for action classification

VideoLSTM starts from the soft-Attention LSTM model of Xu et al. (2015) (or ALSTM) and introduces two novel modules, the *Convolutional ALSTM* and the *Motion-based Attention* networks.

Notation and terminology. We denote 1-d vector variables with lowercase letters, while 2-d matrix or 3-d tensor variables are denoted with uppercase letters. Unless stated otherwise, all activation functions ($\sigma(\cdot)$, $\tanh(\cdot)$) are applied on an element-wise manner and $\odot$ is an element-wise multiplication. When implementing a particular architecture containing multiple copies of the network, *e.g.*, in unrolled networks, we refer to each of the networks copies as unit. For instance, the unrolled version of an LSTM network, see Fig. 1, is composed of several LSTM units serially connected.

We start with a video composed of a sequence of T frames and obtain the image representation $X_{1:T} = \{X_1, X_2, ..., X_T\}$ for each frame using a ConvNet (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; Szegedy et al., 2015). Unlike previous studies (Donahue et al., 2015; Yue-Hei Ng et al., 2015) that use features from the last fully connected layer, we choose the convolutional feature maps, which retains spatial information of the frames. Therefore, a feature map X_t at each timestep t has a dimension of $N \times N \times D$, where $N \times N$ is the spatial dimentionality of the convolutional feature map and D equals to the number of convolutional filters that produce the feature map. We also define $\widetilde{X}_t$ as the attention weighted image representation by combining image features X_t and the attention map A_t at each frame t. A_t is a $N \times N$ matrix composed of attention weights at all spatial locations.

3.1. Convolutional ALSTM

A video naturally has spatial and temporal components. However, standard LSTM and ALSTM networks make use of full connections and treat the input as linear sequences by vectorizing the feature maps or using the fully connected activations from a deep convolutional network. This results in a major drawback for handling spatio-temporal data like videos, since no spatial information is encoded. It is an important property of images that nearby pixels are more strongly correlated than distant pixels. This property suggests local connectivity is preferred rather than full connectivity in images. In order to preserve the spatial structure of the frames over time, we propose to replace the fully connected multiplicative operations in an LSTM unit with convolutional operations. As research on LSTM has progressed, different LSTM variants have been proposed. We utilize the LSTM unit as described in Donahue et al. (2015), a slight simplification derived from the original LSTM unit proposed in Hochreiter and Schmidhuber (1997). The LSTM updates at time step t given inputs $\widetilde{X}_t$ as follows:

$$I_t = \sigma(W_{xi} * \widetilde{X}_t + W_{hi} * H_{t-1} + b_i) \quad (1)$$

$$F_t = \sigma(W_{xf} * \widetilde{X}_t + W_{hf} * H_{t-1} + b_f) \quad (2)$$

$$O_t = \sigma(W_{xo} * \widetilde{X}_t + W_{ho} * H_{t-1} + b_o) \quad (3)$$

$$G_t = \tanh(W_{xc} * \widetilde{X}_t + W_{hc} * H_{t-1} + b_c) \quad (4)$$

$$C_t = F_t \odot C_{t-1} + I_t \odot G_t \quad (5)$$

$$H_t = O_t \odot \tanh(C_t), \quad (6)$$

where * represents the convolutional operation, $W_{x\sim}$, $W_{h\sim}$ are 2-d convolutional kernels. The gates I_t, F_t, O_t, the candidate memory G_t, memory cell C_t, C_{t-1}, and hidden state H_t, H_{t-1} are 3-d tensors and retain spatial dimensions as well.

Different from LSTM and ALSTM who rely on a multi-layer perceptron (MLP) to generate the attention weights, we employ a shallow ConvNet with no fully connected layers conditioned on the previous hidden state and the current feature map. More specifically, the attention map is generated by convolving the previous hidden state H_{t-1} and the current input feature map X_t,

$$Z_t = W_z * \tanh(W_{xa} * X_t + W_{ha} * H_{t-1} + b_a). \quad (7)$$

By replacing the inner products with convolutions, Z_t is a 2-d score map now. From Z_t we can compute the normalized spatial attention map:

$$A_t^{ij} = p(att_{ij} | X_t, H_{t-1}) = \frac{\exp(Z_t^{ij})}{\sum_i \sum_j \exp(Z_t^{ij})}, \quad (8)$$

where A_t^{ij} is the element of the attention map at position (i, j). Instead of taking the expectation over the features from different spatial locations as in ALSTM ($\tilde{x}_t = \sum_{i=1}^{N^2} a_t^i x_t^i$), we now preserve the spatial structure by weighting the feature map locations only without taking the expectation. Formally, this is simply equivalent to an element-wise product between each channel of the feature map and the attention map:

$$\widetilde{X}_t = A_t \odot X_t. \quad (9)$$

Effectively the attention map suppresses the activations from the spatial locations that have lower attention saliency.

Between the ALSTM and the Convolutional ALSTM models we spot three differences, see Fig. 2 for an illustration. First, by replacing the inner products all the state variables, I_t, F_t, O_t, G_t, C_t, H_t, of the Convolutional ALSTM model retain a spatial, 2-d structure. As such, the video is now reckoned as a spatio-temporal medium and the model can hopefully capture better the fine idiosynchracies that characterize particular actions. As an interesting side note, given that the state variables are now 2-d, we could in principle visualize them as images, which would add to the understanding of the internal workings of the LSTM units. Second, Convolutional ALSTM resembles essentially a deep ConvNet, whose layers have recurrent connections to themselves. Last, we should emphasize that the Convolutional ALSTM architecture can receive as input and process any data with a spatial nature. In this work we experiment with RGB and flow frames.

3.2. Motion-based attention

In the Convolutional ALSTM network the attention is generated based on the hidden state of the previous ALSTM unit. The regions of interest in a video, however, are highly correlated to the frame locations where significant motion is observed. This is especially relevant when attention driven recurrent networks are considered. It is reasonable, therefore, to use motion information to help infer the attention in the ALSTM and Convolutional ALSTM network. Specifically, we propose to add another layer with bottom-up connection with the Convolutional ALSTM layer. This layer corresponds to the bottom row of Convolutional LSTM units of Fig. 1 in the green container, which is updated as follows:

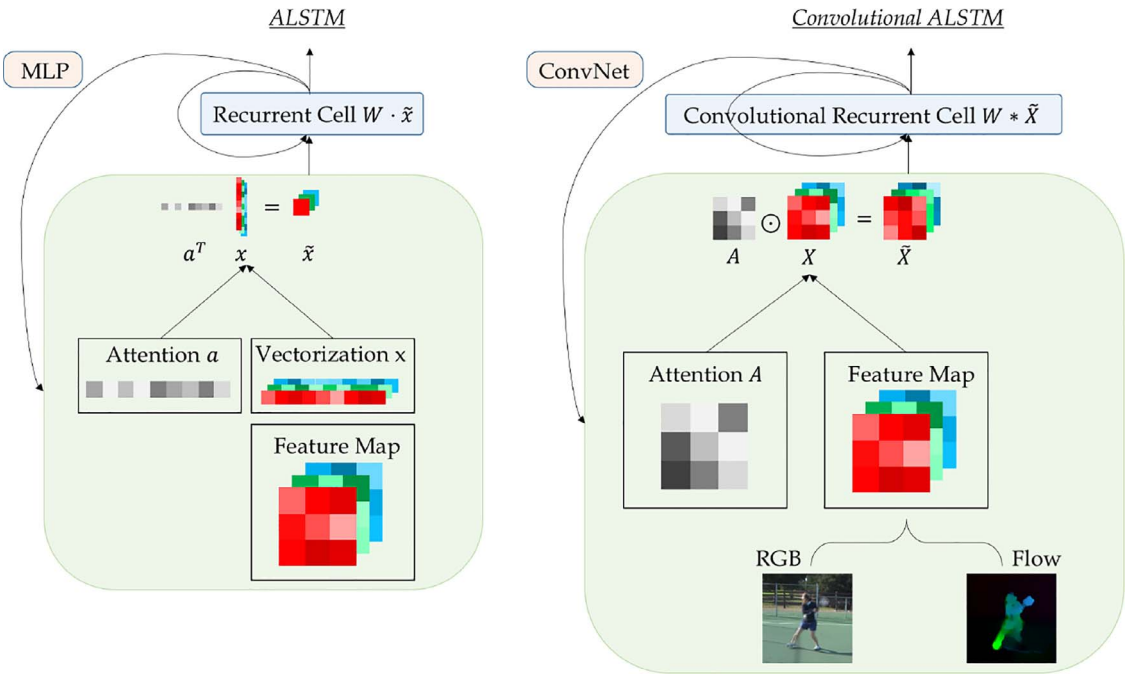

Fig. 2. To the left is an ALSTM model, which weighs the input vector dimensions based on attention and outputs a D-dimensional vector. To the right the proposed Convolutional ALSTM network, which acknowledges the two-dimensional, spatial input, performs a convolution operation and returns a $N \times N \times D$-dimensional tensor preserving the spatial structure of a video.

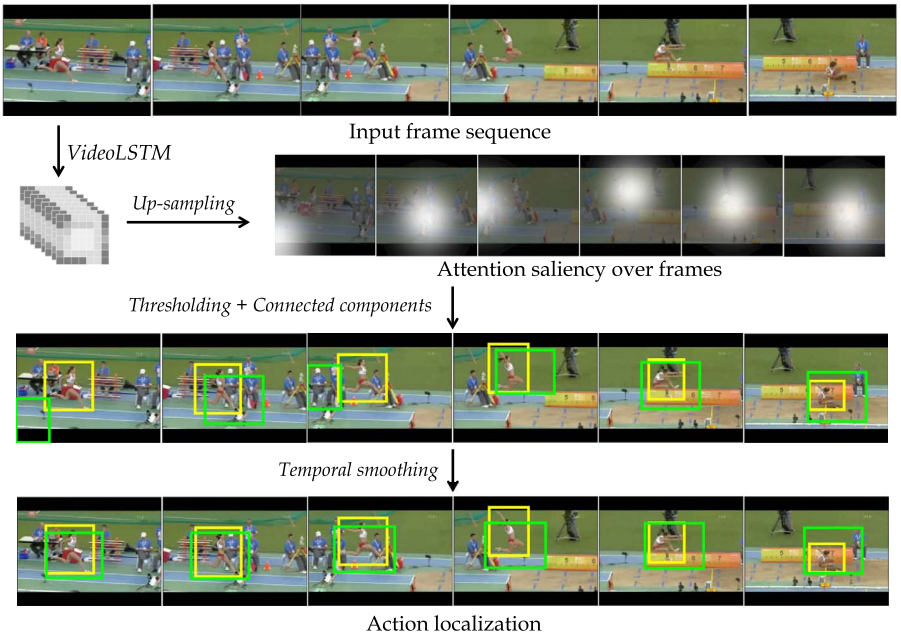

Fig. 3. VideoLSTM generates attention feature maps for an input sequence of frames from an action class label only. These maps are then first up-sampled and smoothed with a Gaussian filter into saliency maps, shown superimposed over frames. Attention saliency maps are then thresholded and processed to localize the action (green box). With local temporal smoothing the boxes are temporally better aligned and lead to better localization of action compared to the ground-truth (yellow box). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$I_t^m = \sigma(W_{xi}^m * M_t + W_{hi}^m * H_{t-1}^m + W_{ei}^m * H_{t-1} + b_i^m) \quad (10)$$

$$F_t^m = \sigma(W_{xf}^m * M_t + W_{hf}^m * H_{t-1}^m + W_{ef}^m * H_{t-1} + b_f^m) \quad (11)$$

$$O_t^m = \sigma(W_{xo}^m * M_t + W_{ho}^m * H_{t-1}^m + W_{eo}^m * H_{t-1} + b_o^m) \quad (12)$$

$$G_t^m = \tanh(W_{xc}^m * M_t + W_{hc}^m * H_{t-1}^m + W_{ec}^m * H_{t-1} + b_c^m) \quad (13)$$

$$C_t^m = F_t^m \odot C_{t-1}^m + I_t^m \odot G_t^m \quad (14)$$

$$H_t^m = O_t^m \odot \tanh(C_t^m), \quad (15)$$

where the previous hidden state from top layer H_{t-1} is also given as input to the bottom layer, and M_t is the feature map extracted from optical flow image at timestep t. W_x^m, W_h^m are 2-d convolutional kernels operating on the inputs and hidden states from the bottom layer, while convolutional kernels W_e^m are applied to the hidden states from the top layer. As shown in Fig. 1, the whole network is built by stacking two layers of convolutional LSTMs, each of which has its individual input streams (RGB or flow images). Based on the updated LSTM cell the attention map is now conditioned on the current hidden state H_t^m from bottom layer. This contrasts to Eq. (8), and standard LSTM architectures also, where the attention is conditioned on the H_{t-1} from top layer. Namely, with the updated LSTM cell the attention at frame t depends on the hidden state from the same frame t, instead of the previous frame $t-1$.

Moreover, note that the motion-based attention map is applied on

the input feature map X_t from the top layer. Therefore, the bottom layer (green container in Fig. 1) only helps to generate the motion-based attention and does not provide any direct information to the top layer for the final classification.

VideoLSTM. As a last step, we define our VideoLSTM as the two layer motion-based Convolutional ALSTM architecture in Fig. 1.

4. VideoLSTM for action localization

By design the VideoLSTM provides a sequence of attention maps A_t, see Eq. (8), which effectively represent the appearance and motion saliency at each frame. Each of these attention maps are first up-scaled with 2-dimensional interpolation assuming affine transform and then smoothed with a Gaussian filter, resulting into saliency maps, $\{S_1, S_2, \cdots, S_T\}$ for the T video frames. See Fig. 3 for an example input video frame sequence. Given saliency maps $\{S_t\}$, our goal is to sample promising spatio-temporal tubes or sequences of spatial regions from the video, which are likely to bound the performed action spatially and temporally. Formally, the pth action proposal is given by $\alpha_p = \{\alpha_p^t\}$, where α_p^t is a bounding-box enclosing a region from the tth frame.

One can think of a variety of complex models exploiting the saliency maps to formulate this problem. We take a naive, greedy approach that selects regions in frame t by simply applying a threshold on saliency map S_t. Each connected component thus obtained forms a region, leading to a set of candidate boxes B_t in S_t,

$$B_t = CC_{box}(S_t \geq \theta_t) \tag{16}$$

where $CC_{box}()$ is a function generating 8-connected components enclosed by rectangular boxes and θ_t is threshold for S_t.

With $|B_t|$ number of boxes for frame t, there are $\Pi_t|B_t|$ possible action proposals. Here, it is possible to generate a large number of action proposals and encode them with state-of-the-art motion features similar to Jain et al. (2014), van Gemert et al. (2015) and Weinzaepfel et al. (2015). However, we choose to avoid using bounding-box ground-truth and keep our approach end-to-end by generating only *a single detection*. In practice, we set high enough threshold such that we rarely get more than one box in a frame. In case of multiple boxes we select the one that has better IoU (intersection over union) with the preceding selected box. Thus, unlike other approaches (van Gemert et al., 2015; Mettes et al., 2016; Saha et al., 2016; Weinzaepfel et al., 2015) in action detection that rely on a large number of proposals, we output *only one* proposal per video; hence dropping the subscript p, we have:

$$\alpha^t = \underset{b \in B_t}{\text{argmax}} \frac{b \cap \alpha^{t-1}}{b \cup \alpha^{t-1}} \tag{17}$$

Sometimes, α^t, are sampled from the background, away from the action or actor. This is expected as the attentions from VideoLSTM are directed to focus on the locations that are discriminative for classification and context information for certain actions can be helpful. Continuing the *Long Jump* example from Fig. 3, the running track distinguishes it from several other actions. Yet, most of the attention does fall on the action/actor and we apply temporal smoothing on the sequence of boxes, α^t, so that sudden deviations from the action can be minimized. We use locally weighted linear regression (first degree polynomial model) to smooth the boxes, such that they do not jitter much from frame to frame.

Other than the attention, VideoLSTM also provides classification scores for each class and each frame. We average these scores over frames to obtain confidence scores of the single proposal for each action class of interest and perform action localization. This is all done for the sake of simplicity. We believe a more sophisticated approach, *e.g.*, considering multiple scales and ratios or using contour information of objects, will further improve action localization performance. Still two key features of our approach that most of the existing action detectors do not have are: (1) *we do not require bounding box* ground-truth for training and (2) *we do not need to see the whole video at once*, frames are processed as they are received, allowing for applications that need to process live video streams.

5. Experiments

5.1. Datasets

UCF101 (Soomro et al., 2012). This dataset is composed of about 13,000 realistic user-uploaded video clips and 101 action classes. The database is particularly interesting, because it comprises of several aspects of actions in video such as various types of activities, camera motion, cluttered background and objects/context. It also provides a relatively large number of samples that is needed for training ConvNet/LSTM networks and hence has been popular among approaches based on deep learning. There are 3 splits for training and testing, following other recent works on end-to-end learning, *e.g.*, Simonyan and Zisserman (2014) and Donahue et al. (2015) we report averaged results on all the splits. Classification accuracy is used as evaluation measure.

HMDB51 (Kuehne et al., 2011). Composed of 6766 video clips from various sources, the HMDB51 dataset has 51 action classes. The dataset has two versions, the original and the one with motion stabilization, we use the more challenging original one. It has 3 train/test splits each with 3570 training and 1530 test videos. Following the common practice in end-to-end learning we evaluate with classification accuracy averaged over the 3 splits.

THUMOS13 localization (Idrees et al., 2017; Jiang et al., 2013). This dataset is a subset of UCF101 with 24 classes (3207 videos) and is provided with bounding-box level groundtruth for action localization. The dataset is quite challenging and is currently one of the largest datasets for action localization that has a rich variety of actions. The localization set consists of mostly trimmed videos with the actor mostly visible along with a few untrimmed videos also. Following the previous works (van Gemert et al., 2015; Mettes et al., 2016; Weinzaepfel et al., 2015; Yu and Yuan, 2015), we use the first split and report mean average precision (mAP) over all 24 classes.

5.2. Implementation details

As reported in the literature (Feichtenhofer et al., 2016b; Wang et al., 2016), end-to-end architectures are sensitive to hyperparameters, layer designs, different optical flow variations, training styles etc. These optimal settings typically vary per architecture, which makes comparisons on fair grounds hard. For this reason in our internal experiments –where we study Convolutional ALSTM and motion-based attention and compare with traditional architectures – we purposefully reconfigure *all* architectures and training styles to be similar for fair assessment. However, in the experiments where we compare against the state-of-the-art, we rely on our optimal network settings and compare with the original numbers published in the related literature. All code and models will be made available for future reference.

ConvNet architectures. We choose the VGG-16 (Simonyan and Zisserman, 2015) architecture which consists of 13 convolutional layers to train the appearance network and optical flow network. For both networks, we choose the pre-trained ImageNet model as initialization for training. The input of the optical flow network is a single optical flow image which has two channels by stacking the horizontal and vertical flow fields. The optical flow is computed from every two adjacent frames using the algorithm of Zach et al. (2007). We discretize the values of flow fields by linearly rescaling them to [0, 255] range. We then extract the image features from the last fully connected layer (*i.e., fc7*) or the last pooling layer (*i.e., pool5*) of each VGG-16 network. Those features are fed into our LSTM architectures.

LSTM, ALSTM and VideoLSTM architectures. All our LSTM models have a single layer with 512 hidden units with input feature

vectors from *fc*7. For ALSTM and VideoLSTM, we use the convolutional feature maps extracted from the *pool*5 layer with size $7 \times 7 \times 512$. Although the convolutional features from the earlier layers (*e.g., conv*5, *pool*4) could also be used or combined (Ballas et al., 2016), we consider *pool*5 only in our work for the sake of efficiency. Convolutional kernels for input-to-state and state-to-state transitions are of size 3×3, while 1×1 convolutions are used to generate the attention map in Eq. (7). All these convolutional kernels have 512 channels. The hidden representations from the LSTM, ALSTM and VideoLSTM layer are then fully connected to an output layer which has 1024 units. A dropout (Srivastava et al., 2014) is also applied on the output before fed to the final softmax classifier with a ratio of 0.7.

For network training, we randomly sample a batch of 128 videos from the training set at each iteration. From each video, a snippet of 30 frames is randomly selected. We do not perform any data augmentation while being aware it will improve our results further. We train all our models by minimizing the cross-entropy loss using back propagation through time and *rmsprop* (Tieleman and Hinton, 2012) with a learning rate of 0.001 and a decay rate 0.9. At test time, we follow (Simonyan and Zisserman, 2014) to sample 25 equally spaced segments from each video with size of 30 frames. To obtain the final video-level prediction, we first sum the LSTM frame-level class predictions over time and then average the scores across the sampled segments. For HMDB51 dataset, as the number of videos for training is very small, we use our pre-trained model on UCF101 to initialize its model. All the models are trained on an NVIDIA GeForce GTX Titan.

5.3. Action classification

Convolutional ALSTM. In the first experiment we compare the Convolutional ALSTM (ConvALSTM) to other architectures using appearance and optical flow input frames. We present the results on UCF101 and HMDB51 first split in Table 1.

We first focus on the appearance frames. A ConvNet (Simonyan and Zisserman, 2015) already brings decent action classification, but inserting a standard LSTM on top, as suggested in Yue-Hei Ng et al. (2015) and Donahue et al. (2015) brings no significant benefit. The reason is that even a single RGB frame can be quite representative of an action. Moreover, since subsequent frames are quite similar, the LSTM memory adds little to the prediction. Adding soft attention (Xu et al., 2015) to the LSTM to arrive at the ALSTM proposed by Sharma et al. (2015); 2016), even deteriorates the action classification performance, while a Convolutional LSTM has a marginal impact as well. However, when considering the proposed Convolutional ALSTM on appearance data, results gain + 2.2% on UCF101 and + 1.1% on HMDB51. A small but important improvement, given that all other LSTM-based architectures fail to bring any benefit over a standard ConvNet.

Before discussing the impact of flow, we first note that both the original ALSTM (Sharma et al., 2016) and ConvLSTM models (Shi et al., 2015) rely on appearance only. A design choice which impacts their action classification potential. With flow frames all the LSTM-based models improve over the ConvNet baseline. We attribute this to the observation that in flow frames the background is not as descriptive and one can better rely on the succession of flows to recognize an action. When employing attention or convolutions independently with an LSTM we obtain a noticeable improvement over the standard LSTM, indicating that more refined LSTM architectures can exploit the flow information better. Once again, when considering the proposed Convolutional ALSTM we observe the most consistent improvement: + 3.8% over LSTM, + 2.6% over ALSTM and + 1.7% over Convolutional LSTM on UCF101. The improvement is even larger on HMDB51.

Table 1
Convolutional ALSTM networks. For both appearance and optical flow input, our proposed ConvALSTM improves accuracy the most.

	UCF101		HMDB51	
	RGB	Flow	RGB	Flow
ConvNet	77.4	75.2	42.2	41.8
LSTM	77.5	78.3	41.3	46.0
ALSTM	77.0	79.5	40.9	49.2
ConvLSTM (This paper)	77.6	80.4	41.8	48.2
ConvALSTM (This paper)	79.6	82.1	43.3	52.6

Table 2
Motion-based attention further improves both the ALSTM model and ConvALSTM.

Attention	UCF101		HMDB51	
	ALSTM	ConvALSTM	ALSTM	ConvALSTM
Appearance-based	77.0	79.6	40.9	43.3
Motion-based	78.6	79.9	42.6	44.8

Motion-based Attention. Next we evaluate the impact of motion-based attention, by explicitly modeling motion saliency using optical flow to help generate the attention maps while using RGB appearance input. We present the results in Table 2 on UCF101 and HMDB51 first split. For ALSTM models motion-based attention obtains obvious improvements for classification, outperforming appearance-based attention by +1.6% on UCF101 and +1.7% on HMDB51, while for Convolutional ALSTMs we obtain also +1.5% improvement on HMDB51. On UCF101 the improvement is smaller, because the background context is already a very strong indicator of the action class. We expect that for datasets where the background context is less indicative of the action class, motion-based attention will boost accuracy further.

Fig. 4 shows the attention maps generated from our proposed VideoLSTM using RGB raw frames as input and optical flow images as input. We can see that the model does not always attend to the actor when using RGB frames, although the video is correctly classified. The first video is classified as *Long Jump*, however, the running track distinguishes it from other actions. In the second video, the model tends to look at the diving board and predicts the action *Diving*. The contextual cues in the background are already very strong indicators of some action classes. By contrast, the attention generated using optical flow images is more likely to focus consistently on the person performing the action. Therefore, we use the attention maps generated from optical flow images in our action localization experiments.

Comparison with other LSTM architectures. We list in Table 3 the accuracies from other LSTM architectures for action classification. As designs and training regimes vary widely, direct comparisons are hard to make. We list properties of individual architectures to better assess relative merit. Instead of using average, we simply use product fusion on predictions from our VideoLSTM models with RGB input and optical flow input. VideoLSTM obtains the best result on both UCF101 and HMDB51. We note that the second best performing LSTM architecture by Yue-Hei Ng et al. (2015), the same architecture as the second column in Table 1, performs pre-training on 1 million sport videos, where our approach pre-trains on ImageNet only. Our implementation of the ALSTM of Sharma *et al.* obtains 75.8 mAP on UCF101, where our VideoLSTM obtains 88.9, a notable difference due to the convolutions and motion-based attention. On HMDB51 the difference is even more pronounced.

State-of-the-art comparison: fusion. As a final experiment, we explore the benefit of a late fusion of VideoLSTM action predictions with approaches relying on hand-crafted iDT features (Peng et al., 2014; Wang and Schmid, 2013) and object responses (Jain et al., 2015). The prediction scores are fused by product with exponential weights. We report the results on UCF101 and HMDB51 in Table 4. First of all, adding our VideoLSTM on top of iDT features improves the results on UCF101 to 91.5% and on HMDB51 to 63.0% (data not shown). This

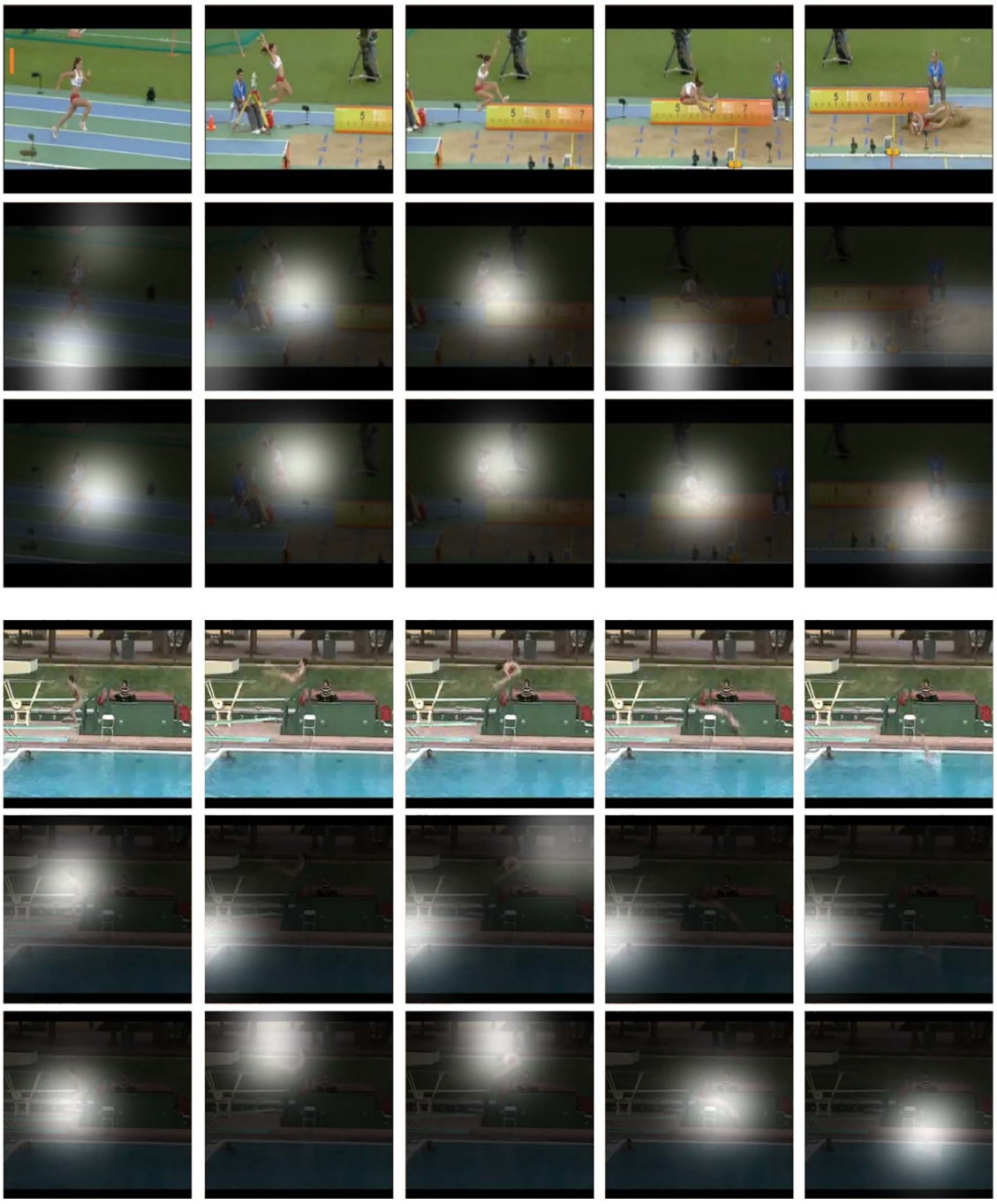

Fig. 4. VideoLSTM generates attention maps for an input sequence of images. For each video, the first row indicates the original video frames. The second row represents the attention maps generated using RGB raw frames as input, while the third row is generated using optical flow images as input. Attention maps are shown superimposed over frames, and the brightness indicates the strength of attention. Note that the attention generated using optical flow images is more likely to focus consistently on the person performing the action.

Table 3
State of the art comparison for LSTM-like architectures. Results on UCF101 and HMDB51 are averaged over all 3 splits, the UCF101 results for Sharma et al. (2015, 2016) are based on our implementation of their ALSTM. Srivastava et al. (2015) report results on HMDB51 using RGB input only. Even without the need to train on more than 1 million sports videos, VideoLSTM is competitive amidst alternative LSTM architectures.

	Input		ConvNet		LSTM		Pre-Training		UCF101	HMDB51
	RGB	Flow	Deep	Very deep	Plain	Attention	ImageNet	Sports1M		
Donahue et al. (2015)	✓	✓	✓	–	✓	–	✓	–	82.3	n/a
Yue-Hei Ng et al. (2015)	✓	✓	–	✓	✓	–	✓	✓	88.6	n/a
Srivastava et al. (2015)	✓	✓	–	✓	✓	–	✓	✓	84.3	44.0
Sharma et al. (2015); 2016)	✓	–	–	✓	–	✓	✓	–	75.8	41.3
This paper	✓	✓	–	✓	–	✓	✓	–	88.9	56.4

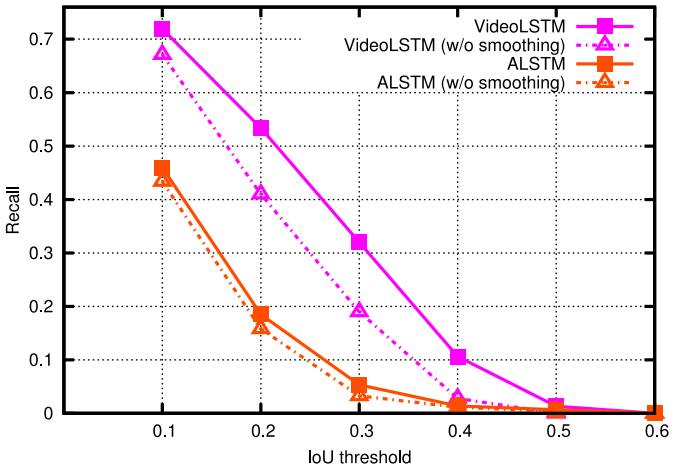

Fig. 5. VideoLSTM performs better than ALSTM of Sharma et al. (2015, 2016). The benefit of temporal smoothing suggests most of our attentions are on the action foreground, which does not seem to be the case for ALSTM.

Table 4
Comparison to state-of-the-art: fusion. The fusion experiment shows that VideoLSTM is highly complementary to hand-crafted iDT features (Peng et al., 2014; Wang and Schmid, 2013) and object responses (Jain et al., 2015), resulting in good accuracy (averaged over all 3 splits) on UCF101 and state-of-the-art on HMDB51.

	UCF101	HMDB51
Jain et al. (2015)	88.5	71.3
Wang et al. (2015b)	91.5	65.9
Zhu et al. (2016)	93.1	63.3
Feichtenhofer et al. (2016b)	93.5	69.2
Wang et al. (2016)	94.2	69.4
de Souza et al. (2016)	92.5	70.4
Lev et al. (2016)	94.1	67.7
Feichtenhofer et al. (2016a)	94.6	70.3
Feichtenhofer et al. (2017)	94.9	72.2
VideoLSTM	88.9	56.4
VideoLSTM + iDT + Objects	92.2	73.7

implies that our end-to-end prediction is highly complementary to hand-crafted approaches, as previously also observed by others (Feichtenhofer et al., 2016b; Lev et al., 2016; de Souza et al., 2016). When combined with the object and iDT motion encoding from Jain et al. (2015), we further improve performance to 92.2% on UCF101, where Feichtenhofer et al. (2016a) score 94.6%, and we set a new state-of-the-art of 73.7% on HMDB51.

5.4. Action localization

In the final experiment we evaluate VideoLSTM for action localization. We set the threshold for saliency maps in Eq. (16) to a constant ($\theta_t = 100$) to ensure pixels with reasonable attention are included. We did not optimize or cross-validate over training data as it would require bounding-box ground-truth. Along with VideoLSTM, we also process the saliency maps from the ALSTM of Sharma et al. (2016), and compare the two in Fig. 5.

There are two things to note here. First, VideoLSTM leads to strikingly higher recalls than ALSTM. Due to its motion awareness and spatial structure preserving properties, VideoLSTM is capable of localizing actions, whereas ALSTM does not seem to do so. Second, the impact of temporal smoothing is considerable in case of VideoLSTM, which means most of the attentions are on the action/actor foreground. In contrast, smoothing does not help ALSTM which suggests that the attention is either stationary or is evenly distributed between foreground and background.

In Table 5, we compare with state-of-the-art action localization methods on the THUMOS13 localization dataset for an IoU thresholds of 0.2, following (Mettes et al., 2016). When we consider approaches that use action class labels only VideoLSTM is state-of-the-art, improving considerably over alternatives. Naturally, the action localization approaches with the most amount of supervision thrive and VideoLSTM cannot compete with them. Interestingly, however, VideoLSTM is able to slightly outperform recent methods that rely on unsupervised action proposals in combination with box (van Gemert et al., 2015) or point supervision (Mettes et al., 2016), in addition to the action class. Note that these methods need thousands of spatio-temporal action proposals (van Gemert et al., 2015), where we predict a single proposal only. Several visual examples of our action localization results are shown in Fig. 6. We conclude that VideoLSTM returns very good action localization from an action class label only, especially considering that only a single spatio-temporal proposal is returned.

6. Conclusion

In this work we postulate that in order to model videos accurately with LSTMs, we must adapt the model architecture to fit the video medium and not vice versa. The video model must, therefore, address common video properties, *i.e.*, what is the right appearance and motion representation, how to transform spatio-temporal video content into a memory vector, how to model the spatio-temporal locality of an action, *simultaneously* and not in isolation. We propose *VideoLSTM*, a recurrent neural network architecture intended for action recognition. VideoLSTM makes three novel contributions to address these video properties in a joint fashion: *(i)* it introduces convolutions to exploit the spatial correlations in images, *(ii)* it introduces a convolutional neural network to allow for motion information to generate motion-based attention maps and finally *(iii)* by only relying on video-level action class labels it exploits the attention maps to localize the action spatio-

Table 5
State-of-the-art localization results ordered by supervision on THUMOS13 localization for an overlap threshold of 0.2. The numbers of Sharma et al. (2015); 2016) are based on our ALSTM, as they don't report action localization results. When considering action class supervision only, VideoLSTM is state-of-the-art. Compared to Saha et al. (2016) and Weinzaepfel et al. (2015) VideoLSTM is worse, but note that we rely on an action class label only to make one prediction, whereas they need box supervision and up-to 16k trials to find the best action location. Interestingly, VideoLSTM is able to outperform methods that also leverage box (van Gemert et al., 2015) or point supervision (Mettes et al., 2016) in addition to the action class.

	Training per video			Testing on THUMOS13	
	#Class	#Annotations	#Proposals	#Proposals	mAP
Saha et al. (2016)	action label	~ 400 boxes	16k	16k	0.668
Weinzaepfel et al. (2015)	action label	~ 400 boxes	256	256	0.468
van Gemert et al. (2015) (from Mettes et al., 2016)	action label	~ 400 boxes	2299	2299	0.345
Mettes et al. (2016)	action label	~ 400 points	2299	2299	0.348
Sharma et al. (2015, 2016)	action label	n/a	0	1	0.055
Cinbis et al. (2014) (from Mettes et al., 2016)	action label	n/a	0	1	0.136
This paper	action label	n/a	0	1	0.369

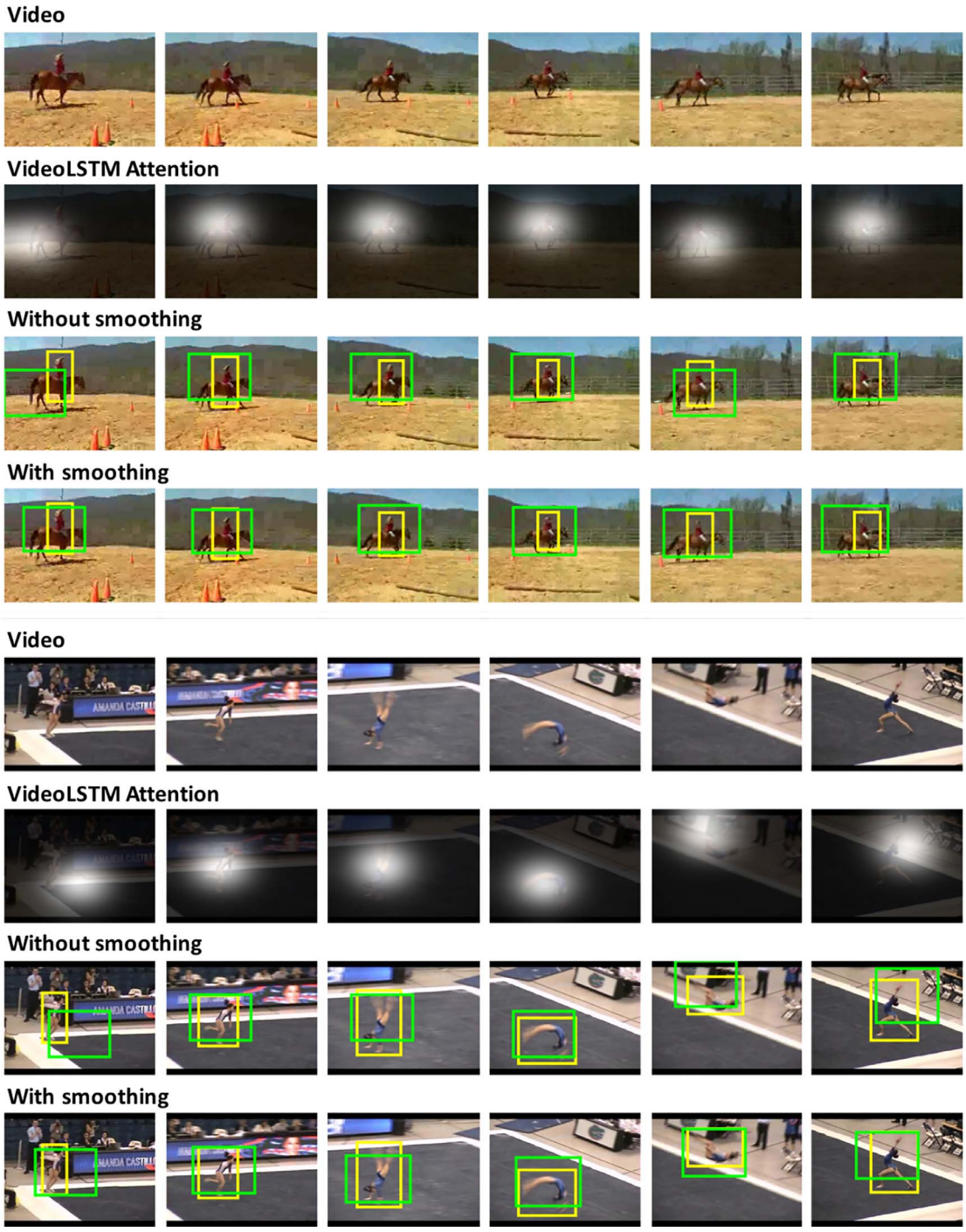

Fig. 6. Two examples of our action localization results using VideoLSTM. In each video, we show the attention generated from VideoLSTM (second row), action localization without temporal smoothing (third row) and action localization with temporal smoothing (fourth row). The green boxes are our predictions, while the yellow boxes indicate ground-truth. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

temporally. Experiments on three challenging datasets outline the theoretical as well as the practical merits of VideoLSTM. As previously observed by Fernando et al. (2015) on Hollywood2, a video representation capturing temporal dynamics is particularly superior for videos with longer clips. We will also investigate the capability of our VideoLSTM for modeling large-scale untrimmed videos, *e.g.*, ActivityNet (Heilbron et al., 2015), in our future work.

References

Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., Baskurt, A., 2010. Action classification in soccer videos with long short-term memory recurrent neural networks. In: ICANN.
Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., Baskurt, A., 2011. Sequential deep learning for human action recognition. In: HBU.
Bahdanau, D., Cho, K., Bengio, Y., 2015. Neural machine translation by jointly learning to align and translate. In: ICLR.
Ballas, N., Yao, L., Pal, C., Courville, A., 2016. Delving deeper into convolutional networks for learning video representations. In: ICLR.

Cinbis, R.G., Verbeek, J., Schmid, C., 2014. Multi-fold mil training for weakly supervised object localization. In: CVPR.
Donahue, J., Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T., 2015. Long-term recurrent convolutional networks for visual recognition and description. In: CVPR.
Feichtenhofer, C., Pinz, A., Wildes, R., 2017. Spatiotemporal multiplier networks for video action recognition. In: CVPR.
Feichtenhofer, C., Pinz, A., Wildes, R.P., 2016. Spatiotemporal residual networks for video action recognition. In: NIPS.
Feichtenhofer, C., Pinz, A., Zisserman, A., 2016. Convolutional two-stream network fusion for video action recognition. In: CVPR.
Fernando, B., Gavves, E., Oramas, J., Ghodrati, A., Tuytelaars, T., 2015. Modeling video evolution for action recognition. In: CVPR.
van Gemert, J.C., Jain, M., Gati, E., Snoek, C.G.M., 2015. APT: action localization proposals from dense trajectories. In: BMVC.
He, K., Zhang, X., Ren, S., Sun, J., 2015. Deep residual learning for image recognition. In: CoRR.
Heilbron, F.C., Escorcia, V., Ghanem, B., Niebles, J.C., 2015. ActivityNet: a large-scale video benchmark for human activity understanding. In: CVPR.
Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9, 1735–1780.
Idrees, H., Zamir, A.R., Jiang, Y.-G., Gorban, A., Laptev, I., Sukthankar, R., Shah, M., 2017. The THUMOS challenge on action recognition for videos “in the wild”. In: CVIU 155, 1–23.
Itti, L., Koch, C., Niebur, E., 1998. A model of saliency-based visual attention for rapid scene analysis. In: PAMI 20, 1254–1259.
Jain, M., van Gemert, J.C., Jégou, H., Bouthemy, P., Snoek, C.G.M., 2014. Action localization by tubelets from motion. In: CVPR.
Jain, M., van Gemert, J.C., Snoek, C.G.M., 2015. What do 15,000 object categories tell us about classifying and localizing actions? In: CVPR.
Jain, M., Jégou, H., Bouthemy, P., 2013. Better exploiting motion for better action recognition. In: CVPR.
Ji, S., Xu, W., Yang, M., Yu, K., 2013. 3d convolutional neural networks for human action recognition. In: PAMI 35, 221–231.
Jia, X., Gavves, E., Fernando, B., Tuytelaars, T., 2015. Guiding the long-short term memory model for image caption generation. In: ICCV.
Jiang, Y.-G., Liu, J., Roshan Zamir, A., Laptev, I., Piccardi, M., Shah, M., Sukthankar, R., 2013. THUMOS challenge: action recognition with a large number of classes.
Karpathy, A., Fei-Fei, L., 2015. Deep visual-semantic alignments for generating image descriptions. In: CVPR.
Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L., 2014. Large-scale video classification with convolutional neural networks. In: CVPR.
Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In: NIPS.
Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T., 2011. HMDB: a large video database for human motion recognition. In: ICCV.
Lan, Z., Lin, M., Li, X., Hauptmann, A.G., Raj, B., 2015. Beyond Gaussian pyramid: multi-skip feature stacking for action recognition. In: CVPR.
LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proc. IEEE 86, 2278–2324.
Lev, G., Sadeh, G., Klein, B., Wolf, L., 2016. RNN Fisher vectors for action recognition and image annotation. In: ECCV.
Mettes, P., van Gemert, J.C., Snoek, C.G.M., 2016. Spot on: action localization from pointly-supervised proposals. In: ECCV.
Peng, X., Wang, L., Wang, X., Qiao, Y., 2016. Bag of visual words and fusion methods for action recognition: comprehensive study and good practice. In: CVIU 150, 109–125.
Peng, X., Zou, C., Qiao, Y., Peng, Q., 2014. Action recognition with stacked fisher vectors. In: ECCV.
Sadanand, S., Corso, J.J., 2012. Action bank: a high-level representation of activity in video. In: CVPR.
Saha, S., Singh, G., Sapienza, M., Torr, P.H.S., Cuzzolin, F., 2016. Deep learning for detecting multiple space-time action tubes in videos. In: BMVC.
Sharma, S., Kiros, R., Salakhutdinov, R., 2015. Action recognition using visual attention. In: NIPS workshop.
Sharma, S., Kiros, R., Salakhutdinov, R., 2016. Action recognition using visual attention. In: ICLR workshop.
Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., Woo, W.-C., 2015. Convolutional LSTM network: a machine learning approach for precipitation nowcasting. In: NIPS.
Simonyan, K., Zisserman, A., 2014. Two-stream convolutional networks for action recognition in videos. In: NIPS.
Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In: ICLR.
Soomro, K., Zamir, A.R., Shah, M., 2012. UCF101: a dataset of 101 human actions classes from videos in the wild. In: CoRR.
de Souza, C.R., Gaidon, A., Vig, E., López, A.M., 2016. Sympathy for the details: Dense trajectories and hybrid classification architectures for action recognition. In: ECCV.
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. In: JMLR 15, 1929–1958.
Srivastava, N., Mansimov, E., Salakhutdinov, R., 2015. Unsupervised learning of video representations using LSTMs. In: ICML.
Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. In: CVPR.
Tieleman, T., Hinton, G., 2012. Rmsprop: divide the gradient by a running average of its recent magnitude. Coursera Course: Neural Networks for Machine Learning.
Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M., 2015. Learning spatiotemporal features with 3d convolutional networks. In: ICCV.
Wang, H., Schmid, C., 2013. Action recognition with improved trajectories. In: ICCV.
Wang, L., Qiao, Y., Tang, X., 2015. Action recognition with trajectory-pooled deep-convolutional descriptors. In: CVPR.
Wang, L., Qiao, Y., Tang, X., 2015. Action recognition with trajectory-pooled deep-convolutional descriptors. In: CVPR.
Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L., 2016. Temporal segment networks: towards good practices for deep action recognition. In: ECCV.
Weinzaepfel, P., Harchaoui, Z., Schmid, C., 2015. Learning to track for spatio-temporal action localization. In: ICCV.
Wu, Z., Wang, X., Jiang, Y.-G., Ye, H., Xue, X., 2015. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In: MM.
Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., Bengio, Y., 2015. Show, attend and tell: neural image caption generation with visual attention. In: ICML.
Yu, G., Yuan, J., 2015. Fast action proposals for human action detection and search. In: CVPR.
Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G., 2015. Beyond short snippets: deep networks for video classification. In: CVPR.
Zach, C., Pock, T., Bischof, H., 2007. A duality based approach for realtime tv-l1 optical flow. DAGM Symposium on Pattern Recognition.
Zhu, W., Hu, J., Sun, G., Cao, X., Qiao, Y., 2016. A key volume mining deep framework for action recognition. In: CVPR.