Action Recognition Using Visual Attention with Reinforcement Learning

Hongyang Li[1,3(✉)], Jun Chen[1,2], Ruimin Hu[1,2], Mei Yu[3], Huafeng Chen[4], and Zengmin Xu[1]

[1] National Engineering Research Center for Multimedia Software, School of Computer Science, Wuhan University, Wuhan, China
{lihy,chenj,hrm,xzm1981}@whu.edu.cn
[2] Hubei Key Laboratory of Multimedia and Network Communication Engineering, Wuhan University, Wuhan, China
[3] College of Computer and Information Technology, China Three Gorges University, Yichang, China
yumei_sim@whu.edu.cn
[4] Jingchu University of Technology, Jingmen, China
chenhuafeng@whu.edu.cn

Abstract. Human action recognition in videos is a challenging and significant task with a broad range of applications. The advantage of the visual attention mechanism is that it can effectively reduce noise interference by focusing on the relevant parts of the image and ignoring the irrelevant part. We propose a deep visual attention model with reinforcement learning for this task. We use Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) units as a learning agent. The agent interact with video and decides both where to look next frame and where to locate the most relevant region of the selected video frame. REINFORCE method is used to learn the agent's decision policy and back-propagation method is used to train the action classifier. The experimental results demonstrate that this glimpse window can focus on important clues. Our model achieves significant performance improvement on the action recognition datasets: UCF101 and HMDB51.

Keywords: Human action recognition · Reinforcement learning Visual attention

1 Introduction

Action recognition is a prominent research area in video understanding, which can be applied to many applications such as video surveillance, human-computer interaction, human behavior understanding, etc. Though significant progresses have been made [2,17,20,23], action recognition still remains a challenging task due to intra-class variations, background complexity, high-dimensional feature description, and other difficulties.

© Springer Nature Switzerland AG 2019
I. Kompatsiaris et al. (Eds.): MMM 2019, LNCS 11296, pp. 365–376, 2019.
https://doi.org/10.1007/978-3-030-05716-9_30

Previous research in action recognition focus on Bag of Words (BoW) model based on shallow high-dimensional encodings of local features. Local feature vector of the video is commonly expressed by Motion Boundary Histogram (MBH), Histogram of Oriented Gradient (HOG), and Histogram of Optical Flow (HOF), etc. Improved Dense trajectories (IDT) feature [20] is a new local visual feature by combining trajectory shape descriptor, HOG, HOF and MBH, which is superior to other local feature in the most challenging video datasets.

Recently, Convolutional Neural Network (CNN) have show a great ability to produce a rich representation of the image and have highly successful in image understanding, such as image classification, object detection, image segmentation. Classifying videos instead of images adds a temporal information to the model of image classification. Wang et al. [21] extend convolutional kernel to multiple video frames to extract temporal information. Although they achieve great performance, much temporal information is still missing. LSTM have also been used to learn an effective representation of videos [15,19], which has been proven to be effective for action recognition task from video sequences. There are many approaches also tend to have CNN underlying the LSTM and classify sequences directly or do temporal pooling of features prior to classification [3]. For the target object is not in a fixed position, the policy of sampling in a fixed area is difficult to adapt to video frame sequences with large time span. Xu et al. [26] try to solve the problem by visual salient region boundary based dense sampling strategy. These salient regions model are not obtained for action recognition task and do not take full advantage of supervise information.

Attention mechanisms have become an integral part of compelling sequence modeling in mangy tasks. The most important function of selective visual attention is to quickly turn our attention to objects of interest in the visual environment. This ability to focus on regions in cluttered visual scenes is of evolutionary significance. The key instinct is that humans do not immediately focus on the whole scene, but rather focus on the sequential parts of the scene to extract relevant information. The process of action recognition is continuous and iterative observation and refinement. This paper draws inspiration from works that have used REINFORCE to learn spatial glimpse policy for image classification [1,14], and to learn temporal glimpse policy for action detection [27].

Main contributions of this work are: (1) We directly estimate the next video frame location and next retina region based on current and historical information by reinforcement learning method. (2) The next glimpse location is relative to the current position instead of the whole image.

2 Related Work

Two-stream convolutional network model combines the predictions of two convolutional neural networks: one trained on single video frames and the other trained on short sequences of dense optical flow images [17]. This deep architecture is competitive with the classical shallow representations in spite of being trained on relatively small datasets. On this basis, researchers have proposed more improved models [5,7,22,24]. Karpathy et al. [9] used a multi-resolution

CNN architecture to perform action recognition in videos. ARTNet architecture to are constructed by stacking multiple generic building blocks, whose goal is to simultaneously model appearance and relation from RGB input in a separate and explicit manner [23]. Using reinforcement learning, Ji et al. [8] separate human action to several patterns and learn temporal transition expected values in activity sequences.

Attention networks were originally proposed on the basis of the REINFORCE algorithm, that is called hard attention. Soft attention mechanisms were proposed by using weighted averages instead of hard selections. Long et al. [13] propose a local feature integration framework based on multimodal soft-attention clusters, and introduce a shifting operation to capture more diverse signals. VideoLSTM based on soft attention LSTM is an end-to-end sequence learning model [12]. Girdhar et al. [6] introduce soft-attention pooling model to action recognition and human object interaction tasks. Soft-attention model is also used in action recognition. Sharma et al. [16] propose a soft attention based model for the task of action recognition in videos, which learns which parts in the frames are relevant for the task at hand and attaches higher importance to them and classifies videos after taking a few glimpses. Zhang et al. [28] propose a novel attention mechanism that leverages the gate system of LSTM to compute the attention weights, which is embedded in a recurrent attention network that can explore the spatial-temporal relations between different local regions to concentrate important ones.

RAM model based on a recurrent neural network is capable of extracting information from an image or video by adaptively selecting a sequence of regions or locations and only processing the selected regions at high resolution; While the model is non-differentiable, it can be trained using reinforcement learning methods to learn task-specific policies [14]. DRAM extends attention based model for recognizing multiple objects in images [1]. Xu et al. [25] use both soft attention and hard attention mechanisms to describe the content of images. Yeung et al. [27] formulate the hard attention model as a recurrent neural network based agent that interacts with a video over time and decides both where to look next and when to emit a prediction for action detection task.

3 The Method

Our task is to take a long sequence of video as input and predict any class labels of a given human action. Figure 1 shows the model structure. At each time step, the RNN as an agent processes the glimpse from one video frame, integrates information over time, and chooses how to act and how to locate next frame and get glimpse at each time step.

3.1 Architecture

The architecture is built around a RNN, which consists of four main components: an observation network, a convolutional neural network, a recurrent network and a prediction network. The observation network is used to select video frame and choose focusing image patch. The image patch is then sent to the CNN, which is responsible for transforming this two-dimensional images patch into one-dimensional feature vector. RNN is the core component and used to process feature vector sequences from CNN. The prediction network takes the current state of RNN as input and makes a prediction on when and where to extract the next video frame patch for the observation network. We explain how we use a combination of back-propagation and REINFORCE method to train the model in end-to-end fashion.

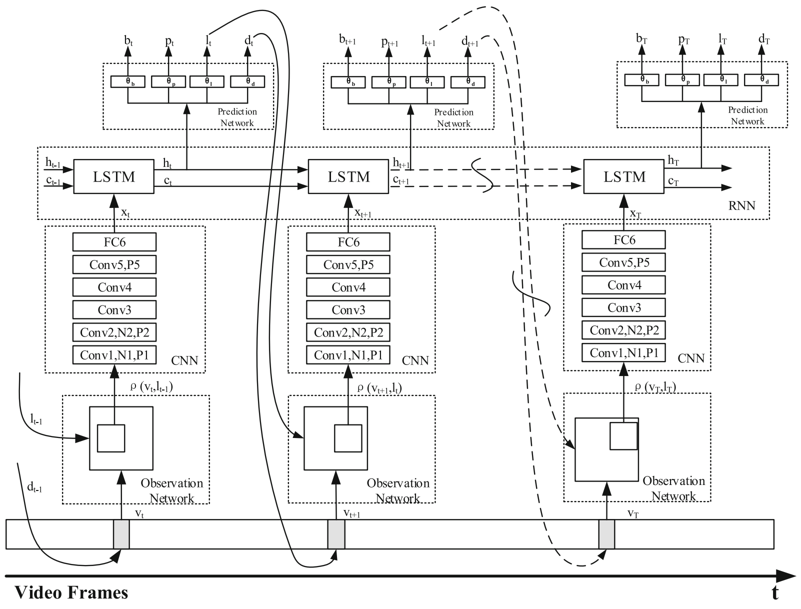

Fig. 1. Model architecture.

Observation Network. At each step t the agent receives a image patch from one video frame. The agent does not have full access to the video frame but rather can extract information by focusing on particular region. The location of this region is determined by two parameters: l_t and d_t. At training time, the location $d_t^{'}$ of observation next frame is sampled from Gaussian distribution with a mean of d_t and a fixed variance; at test time, the maximum a posteriori estimate is used. When the value of $d_t^{'}$ goes beyond the range $[0, 1]$, we clip it by 0 or 1. We define D as maximum span which is the number of frames that observation network can be skipped. Given next frame location of $d_t^{'}$, we get $d = d_t^{'} \cdot D$, this means we can skip $d + 1$ frames to get the next frame. Similarly, the location $l_t^{'}$ of observation region center is sampled from Gaussian distribution with a mean of l_t and a fixed variance. The difference between the two parameters is that $d_t^{'}$ is a scalar and $l_t^{'}$ is a two-dimensional vector. The $l_t^{'}$ represent horizontal and vertical position relative to the previous retina region. The range of value of $l_t^{'}$ is $\{[-1,1],[-1,1]\}$. We set the L as maximum pixel distance from reference position. Given next glimpse location of $l_t^{'}$ and the coordinates c_{t-1} in the image of previous image patch, we get $l = l_t^{'} \cdot L$. This means that the center position of the next glimpse region location is $l + c_{t-1}^{'}$.

Convolutional Network. In our implementation, we choose the configurations similar to two-stream CNN [17] due to their good performance on the challenging datasets. The CNN structure from the first to the fifth layer is similar to spatial stream ConvNet [17], except the first layer's stride equal 1. Using shorthand notation, the CNN configuration is $C1(96,7,1) - N1 - P1(2,2) - C2(256,5,2) - N2 - P2(2,2) - C3(512,3,1) - C4(512,3,1) - C5(512,3,1) - P5(2,2) - FC6(1024)$. where $C\#(c,k,s)$ indicates a convolutional layer with c filters of spatial size $k \times k$, applied to the input with stride s. $P\#(k,s)$ is max pooling layer with spatial size $k \times k$ and stride s. $FC\#(n)$ is a fully connected layer with n nodes. $N\#$ is local response normalization layer described in Krizhevsky et al. [10] and use the same parameters: $k = 2, n = 5, \alpha = 10^{-4}, \beta = 0.5$. The Rectified Linear Units (ReLU) activation function is applied to the output of every convolutional and fully-connected layer. The final layer is connected to LSTM cell as frame presentation feature vector. Except the $FC6$ layer, the parameter values of the other layers using the existing values from the model of [17].

Recurrent Network. The video can be taken as a sequence frames, We use the LSTM implementation in [25], which is given as follows;

$$\begin{aligned}
i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\
f_t &= \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \\
o_t &= \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \\
g_t &= \tanh\left(W_g \cdot [h_{t-1}, x_t] + b_g\right) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh\left(c_t\right)
\end{aligned}$$

where i_t is the input gate, f_t is the forget gate, o_t is the output gate, band c_t is the cell state, h_t is the hidden state, and the vector x_t is the input to the LSTM at time-step t. The x_t is the CNN feature representation vector, capturing the visual information associated with a particular input glimpse. σ and $\odot$ be the logistic sigmoid activation and element-wise multiplication respectively. The trainable parameters are weight matrices W and biases vectors b.

Prediction Network. This network acts as a controller that directs attention based on the current internal states from the RNN. As the agent reasons on a video, four outputs are produced at each timestep: temporal location d_t indicating the frame to observe next. spatial location l_t indicating the glimpse region. p_t represents the classification results of the prediction; Whether in the training or testing stage, only the last unit output p_t is used. b_t is just an auxiliary output.

Loaction of Next Frame: The temporal location $d_t \in [0, 1]$ indicate the video frame location that the agent chooses to observe. The next video frame location d_t is relative to current frame location l_{t-1}. The difference from [27] is that our agent only skip forwards a video, which is more consistent with human's cognitive habits and more easier to handle online. The d_t is computed as $d_t = f_d(h_t, \theta_d)$, where the f_d is a fully connected layer, such that the agent's decision is a function of its past observations and current observation.

Location of Next Observation Region: The spatial location $l_t \in [-1, 1]$ is a two-tuples, which is computed as $l_t = f_l(h_t, \theta_l)$, where the f_l is a fully connected layer with parameters θ_l. The elements of the two tuples correspond to the coordinates of the focusing region center relative to the previous location. The parameters of θ_l are a two-dimensional matrix and a biases vector.

Classifier of Video: Our task is to recognize the human action category in videos, So we need a classifier for this task. $p_t = f_p(h_t, \theta_p)$, where f_p is a fully connected hidden layer with parameters θ_p followed by a softmax output layer. In course of training, we use softmax loss as its loss function and back-propagation method to update parameters θ_p. Whether in the course training or testing, his component outputs the result only at the final timestep.

Baseline of State Value: In reinforcement learning, it is generalized to include a comparison of the action value to an arbitrary baseline $b(s)$. The choice of the baseline does not affect the expected update of the algorithm, but it does affect the variance of the update and thus the rate of convergence. $b_t = f_b(h_t, \theta_b)$, where f_b is a fully connected layer with parameters θ_b. Taken the cumulative reward R as reference value, we use back-propagation method to update parameters θ_b, but gradient is not transferred to the RNN layer. In course of testing, this component is not used.

3.2 Training

The observation location outputs $\{l_t,\ d_t\}$ are non-differentiable components of our model that cannot be trained with standard back-propagation. The parameters of our agent are given by the $\{\theta_l, \theta_d\}$ parameters of the prediction network. After executing an action the agent receives a new visual observation of the video frame v_t and obtains a reward r_t. The goal of the agent is to maximize the cumulative reward which is usually very sparse and delayed: $R = \sum_{t=1}^{T} r_t$. In this paper, our goal is to make a correct classification of the videos. So we set $r_T = 1$ if the action is classified correctly after T steps and 0 otherwise.

The above setup is a special instance of what is known in the RL community as a Partially Observable Markov Decision Process (POMDP). The true state of the environment is unobserved. In this view, the agent needs to learn a policy $\pi((l_t, d_t)|s_{1:t}; \theta)$ with parameters θ that, at each step t, maps the history of past interactions with the environment $s_{1:t} = x_1, l_1, d_1, ..., x_{t-1}, l_{t-1}, d_{t-1}, x_t$ to a distribution over actions for the current time step, subject to the constraint of the observation network. In this paper, the policy π_θ is defined by the prediction

Algorithm 1. Prediction Network Parameters Algorithm

```
Input: differentiable policy $\pi(l_t|h_t, \theta_l), \pi(d_t|h_t, \theta_d)$,step size $\lambda$
    Initialize policy parameters $\theta$
 1: $\delta_l \leftarrow 0, \delta_d \leftarrow 0$
 2: for $m = 1 \rightarrow M$ do
 3:     if $(argmax(p_T) == y)$ then
 4:         $R^m \leftarrow 1$
 5:     else
 6:         $R^m \leftarrow 0$
 7:     end if
 8:     for $t = 1 \rightarrow T$ do
 9:         $\delta_R^m \leftarrow R^m - b_t^m$
10:         $\delta_l \leftarrow \delta_R^m \nabla_{\theta_l} \ln\pi(l_t^m|h_t^m, \theta_l)$
11:         $\delta_d \leftarrow \delta_R^m \nabla_{\theta_d} \ln\pi(d_t^m|h_t^m, \theta_d)$
12:     end for
13: end for
14: $\theta_l \leftarrow \theta_l + \lambda\delta_l,\ \theta_d \leftarrow \theta_d + \lambda\delta_d$
```

network outlined above, and the history s_t is summarized in the state of the hidden units h_t. Give an agent interacting with a video, π_θ is the agent's policy. The objective function of learning distribution over actions conditioned on the interaction sequences is defined as:

$$J(\theta) = \mathrm{E}\left(\left(\sum_{t=1}^{T} r_t\right); \pi_\theta\right) = \sum_s \pi(s, \theta)R$$

where s is interaction sequence obtained by running the current agent π_θ, R is the sum of rewards, $\pi(s, \theta)$ is the agent's policy, $\sum\limits_s \pi(s, \theta)$ represents averaging multiple sequences. The aim of reinforcement learning is to find the best parameter θ to maximize the objective function $J(\theta)$. The process of finding the optimal parameter θ is to find the optimal policy or the optimal path. The above problem is essentially an optimization problem. The simplest and most commonly used method is the gradient descent method. That is:

$$\theta_{new} = \theta_{old} + \lambda \nabla J(\theta)$$

The gradient of the object function is:

$$\begin{aligned}
&\nabla_\theta J(\theta) = \nabla_\theta \sum_s \pi(s, \theta)R \\
&= \sum_s \nabla_\theta \pi(s, \theta)R \\
&= \sum_s \pi(s, \theta) \tfrac{\nabla_\theta \pi(s,\theta)R}{\pi(s,\theta)} \\
&= \sum_s \pi(s, \theta)R\nabla_\theta \log \pi(s, \theta)
\end{aligned}$$

The upper formula is equivalent to the $R\nabla_\theta \log \pi\left(s, \theta\right)$ expectation. This expression can be estimated by Monte Carlo sampling, that is, we get multiple episodes according to the current policy. So, the approximation is:

$$\nabla_\theta J\left(\theta\right) \approx \frac{1}{M}\sum_{m=1}^{M} R^m \nabla_\theta \log \pi\left(s^m, \theta\right)$$

The $\pi\left(s, \theta\right)$ is probability of T timesteps episode, which is represents as:

$$\pi\left(s, \theta\right) = \prod_{t=1}^{T} \pi\left(s_t, \theta\right)$$

Then, the gradient of object function is:

$$\nabla_\theta J\left(\theta\right) \approx \frac{1}{M}\sum_{m=1}^{M}\sum_{t=1}^{T} R^m \nabla_\theta \log \pi\left(s_t^m, \theta\right)$$

To reduce the variance of the gradient estimate, a baseline reward b_t^m is often estimated, e.g. via $b = f_b\left(h_t, \theta_b\right)$ in prediction network, and subtracted so that the gradient equation becomes:

$$\nabla_\theta J\left(\theta\right) \approx \frac{1}{M}\sum_{m=1}^{M}\sum_{t=1}^{T} \left(R_t^m - b_t^m\right) \nabla_\theta \log \pi\left(s_t^m, \theta\right)$$

where $R_t^m = \sum_{n=t}^{T} r_n^m$ is the cumulative reward obtained from t step to T step. In this paper $R_t^m = R^m$. And b_t^m is the estimated value at the h_t state of the mth episode. Due to $\pi\left(\theta\right) = \pi\left(\theta_d\right)\pi\left(\theta_l\right)$, we get:

$$\nabla_\theta \log \pi\left(s, \theta\right) = \nabla_{\theta_d} \log \pi\left(s, \theta_d\right) + \nabla_{\theta_l} \log \pi\left(s, \theta_l\right)$$

Since $\pi\left(s, \theta_d\right)$ is Gaussian distribution, we get:

$$\nabla_{\theta_d} \log \pi\left(s, \theta_d\right) = \frac{\left(l_d^{'} - l_d\right)}{\sigma^2}$$

Here l_d is prediction output, that is taken as mean of this Gaussian distribution. $l_d^{'}$ is the sampled value. As a hyper-parameter, σ is the fixed variance of this Gaussian distribution. In the same way, we can calculate the $\nabla_{\theta_l} \log \pi\left(s, \theta_l\right)$.

The resulting algorithm increases the log-probability of an action that was followed by a larger than expected cumulative reward, and decreases the log-probability if the cumulative reward was smaller. After update $\{\theta_l, \theta_d\}$ parameters, other parameters under the prediction network, e.g. (W, b) of RNN and $FC6$ of CNN, are updated using back-propagation, because they are differentiable. Besides the policy optimization, we also need to optimize the classifier, whose loss function of classifier is defined as softmax loss. To update baseline parameter θ_b, we define its loss function as Euclidean loss. The baseline gradient does not propagate backward to the next layer. Algorithm 1 shows the basic process of gradient descent method. In experiments, we actually use stochastic gradient descent algorithm.

Fig. 2. Sample frames from different actions datasets. (a) UCF101 (b) HMDB51

4 Experiments

In this section, we evaluate performance of the proposed method for action recognition on two action data sets, and compare it with previous methods in literature. The experiments were doing in two datasets that are the most challenging datasets in recently. The performance of the action recognition is evaluated by the average precise. Some example frames are illustrated in Fig. 2.

4.1 Datasets

The **UCF101** dataset [18] has 101 action categories and contains 13,320 videos, consisting of realistic videos taken from YouTube ranging from general sports to daily life exercises, with each category containing at least 100 clips. The dataset is particularly interesting because it gives the largest diversity in terms of actions and with the presence of large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, illumination conditions, etc. There are three splits for training and testing (70% training and 30% testing).

The **HMDB51** dataset [11] collects video clips in abundant source, both from movies and Internet, there are 6,766 videos and 51 action categories in total. We follow the original protocol using three train-test splits. For every class and split, there are 70 videos for training and 30 videos for testing. We report average accuracy over the three splits as performance measure on the original videos.

4.2 Implementation Detail

All the videos in the datasets, the observation image patch is 76×76. For the RNN, the LSTM cell has 512 hidden units. The agent is given a fixed number of observations for each episode, typically $T = 10$ in our experiments. We train model by using stochastic gradient descent optimization algorithm with mini-batches of size 128 episodes and momentum of 0.9, weight decay of 0.0005, initial learning rate of 0.001. Other hyper-parameters were selected using random search. In this paper, we set the $D = 10$, the $L = 70$, the Gaussian distribution variance fixed 0.1.

4.3 Comparison to Baseline

We train our model only using RGB video frames, not using optical flow data. The CNN model [17] and CNN+LSTM model [3] are taken as baseline. As can be seen from Table 1, our model obtains robust improvements over the baseline on UCF101 and HMDB51. The result of the experiment demonstrates that the hard attention approach can decrease the irrelative visual information influence and increase the correctness of classifier.

Table 1. Comparison to baseline on the UCF101 and HMDB51 data sets.

Model	UCF101		HMDB51	
	RGB	RGB+Flow	RGB	RGB+Flow
CNN (Two Stream Network) [17]	73.0	88.0	40.5	59.4
CNN+LSTM [3]	68.2	82.7	-	-
Our Hard-attention Model	93.2	-	66.8	-

4.4 Comparison to State-of-the-Art

Finally, we compare our method against the state-of-the art models. As shown in Table 2, our model can achieve competitive results in comparison with existing published methods. Our results are closed to the [2,13,23] which used much larger pre-train data sets. It is also interesting to observe that in some cases, the model is able to attend to important objects in the video frames and attempts to track them to some extent in order to correctly identify the performed activity.

Table 2. Action Recognition mAP (%) on the UCF101 and HMDB51 data sets.

Model	UFC101	HMDB51	Year	Pre-train dataset
IDT [20]	72.4	40.2	2013	None
Two Stream Network [17]	88.0	59.4	2014	ImageNet
CNN+LSTM [3]	82.7	-	2015	ImageNet
Spatial TDD [22]	82.8	50.0	2015	ImageNet
CNN+LSTM fusion [5]	92.5	65.4	2016	ImageNet
Spatial Stream ResNet [4]	82.3	43.4	2016	ImageNet
TSN Spatial Network [24]	86.4	53.7	2016	ImageNet
RGB-I3D [2]	95.6	74.8	2017	ImageNet+Kinetics
ARTNet with TSN [23]	94.3	70.9	2018	Kinetics
Attention Cluster RGB+Flow [13]	94.6	69.2	2018	Kinetics
Our Hard-attention Model	93.2	66.8	2018	ImageNet

5 Conclusion

In this paper, we have proposed reinforcement learning method for action recognition in videos, which aims to select the most informative frames and the retina region of the input sequences. because of the stochasticity in the glimpse policy during training, The hard-attention based model is less prone to over-fitting than common deep model. Our architecture should be applicable to related tasks such as action localization and detection in video. In terms of future work, we hope to add optical information for prediction of next frame location.

Acknowledgement. The research was supported by the National Nature Science Foundation of China (61671336, U1611461, U1736206), Technology Research Program of Ministry of Public Security (2016JSYJA12), Hubei Province Technological Innovation Major Project (2016AAA015, 2017AAA123), Hubei Provincial Education Department Project (16Q070), Nature Science Foundation of Jiangsu Province (BK20160386).

References

1. Ba, J., Mnih, V., Kavukcuoglu, K.: Multiple object recognition with visual attention. In: ICLR (2015)
2. Carreira, J., Zisserman, A.: Quo vadis, action recognition? A new model and the kinetics dataset. In: CVPR (2017)
3. Donahue, J., et al.: Long-term recurrent convolutional networks for visual recognition and description. In: CVPR (2015)
4. Feichtenhofer, C., Pinz, A., Wildes, R.: Spatiotemporal residual networks for video action recognition. In: NIPS (2016)
5. Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional two-stream network fusion for video action recognition. In: CVPR (2016)
6. Girdhar, R., Ramanan, D.: Attentional pooling for action recognition. In: NIPS (2017)
7. Girdhar, R., Ramanan, D., Gupta, A., Sivic, J., Russell, B.: ActionVLAD: learning spatio-temporal aggregation for action classification. In: CVPR (2017)
8. Ji, Y., Yang, Y., Xu, X., Shen, H.T.: One-shot learning based pattern transition map for action early recognition. Signal Process. **143**, 364–370 (2018)
9. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: CVPR (2014)
10. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
11. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: HMDB: a large video database for human motion recognition. In: ICCV (2011)
12. Li, Z., Gavrilyuk, K., Gavves, E., Jain, M., Snoek, C.G.: VideoLSTM convolves, attends and flows for action recognition. Comput. Vis. Image Underst. **166**, 41–50 (2018)
13. Long, X., Gan, C., de Melo, G., Wu, J., Liu, X., Wen, S.: Attention clusters: purely attention based local feature integration for video classification. In: CVPR (2018)
14. Mnih, V., Heess, N., Graves, A., Kavukcuoglu, K.: Recurrent models of visual attention. In: NIPS (2014)

15. Ng, J.Y.H., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: deep networks for video classification. In: CVPR (2015)
16. Sharma, S., Kiros, R., Salakhutdinov, R.: Action recognition using visual attention. In: ICLR (2016)
17. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: NIPS (2014)
18. Soomro, K., Zamir, A.R., Shah, M.: UCF101: a dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012)
19. Srivastava, N., Mansimov, E., Salakhudinov, R.: Unsupervised learning of video representations using LSTMs. In: ICML (2015)
20. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: ICCV (2013)
21. Wang, K., Wang, X., Lin, L., Wang, M., Zuo, W.: 3D human activity recognition with reconfigurable convolutional neural networks (2014)
22. Wang, L., Qiao, Y., Tang, X.: Action recognition with trajectory-pooled deep convolutional descriptors. In: CVPR (2015)
23. Wang, L., Li, W., Li, W., Van Gool, L.: Appearance-and-relation networks for video classification. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
24. Wang, L., et al.: Temporal segment networks: towards good practices for deep action recognition. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9912, pp. 20–36. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46484-8_2
25. Xu, K., et al.: Show, attend and tell: neural image caption generation with visual attention. In: ICML (2015)
26. Xu, Z., Hu, R., Chen, J., Chen, H., Li, H.: Global contrast based salient region boundary sampling for action recognition. In: Tian, Q., Sebe, N., Qi, G.-J., Huet, B., Hong, R., Liu, X. (eds.) MMM 2016. LNCS, vol. 9516, pp. 187–198. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-27671-7_16
27. Yeung, S., Russakovsky, O., Mori, G., Fei-Fei, L.: End-to-end learning of action detection from frame glimpses in videos. In: CVPR (2016)
28. Zhang, M., Yang, Y., Ji, Y., Xie, N., Shen, F.: Recurrent attention network using spatial-temporal relations for action recognition. Signal Process. **145**, 137–145 (2018)