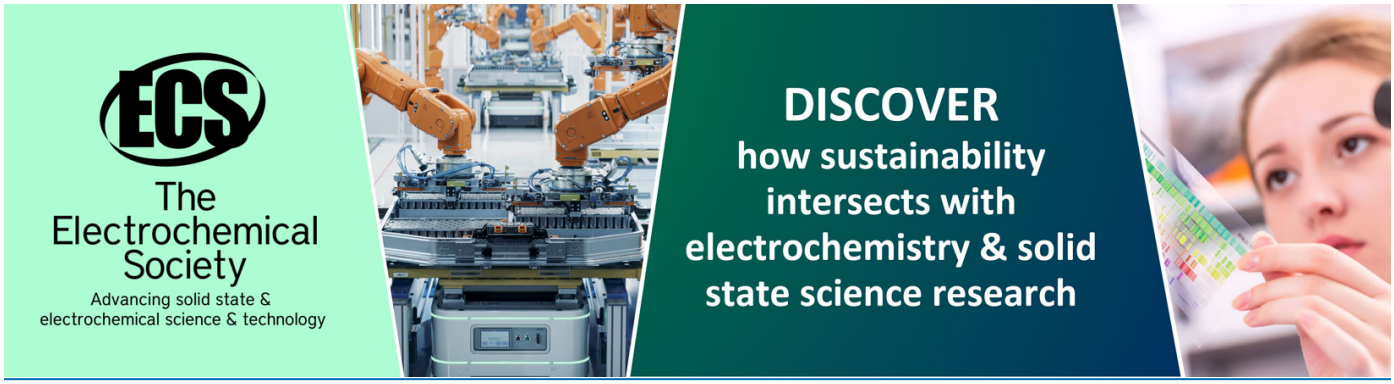
Multi-views Action Recognition on Deep Learning and K-SVD



IOP Conf. Series: Journal of Physics: Conf. Series **1176** (2019) 062015 doi:10.1088/1742-6596/1176/6/062015

Multi-views Action Recognition on Deep Learning and K-SVD

Chuanxu Wang[1], Guofeng Hu[2,*], Yun Liu[3]

[1,2,3]Qingdao University of Science & Technology, Qingdao, China

*Corresponding author e-mail: huguofengcool@qq.com

Abstract. In order to solve the problem of low action accuracy due to changes of view angles, this paper investigates this issue based on deep learning and K-SVD sparse algorithm. Firstly, the paper extracts the feature maps from the different views by convolutional neural networks (CNN) and long short term memory (LSTM), and the extracted feature maps are the multi-views high-level features with semantic information. Secondly, the paper uses the K-SVD sparse algorithm to get the dictionaries corresponding to each views, the dictionaries have very good sparse representations for the features of the action. Then the softmax classifier is used for classification and recognition. The results show that the accuracy are 89.22% and 91.4% on IXMAS datasets and WVU datasets respectively.

1. Introduction

Multi-views action recognition is a quite challenging research in computer vision, mainly because the features of the same action are very different in different views. By using the single-view methods[1-4] to deal with multi-view problems will result in poor results because of the relatively large difference in features[5, 6]. Therefore, it is necessary to find an algorithm that still maintains strong robustness when the view changes.

Researchers have developed a multi-views recognition algorithm on the basis of the single-view to overcome their own problems[5-7]. However, all these algorithms need to detect the interest points, followed by the establishment of feature descriptors by extracting interest points around such as HOG, SIFT and SURF feature vectors. These traditional methods have achieved good results before. In recent years, the large datasets have been set up, such as Hollywood[9], UCF Sports[10], UCF YouTube[11], etc. And using the traditional methods to do the action recognition, firstly, it may spend more times detecting the interest points, secondly, the parameters grows with describing a certain behavior better, then using the CPU only to deal with them, the computational cost is very large.

In order to solve the problem that the final accuracy is not high due to the changes of views, a multi-view recognition algorithm based on deep learning and K-SVD sparse algorithm is presented in this paper. Similar to the opinions in [6] and [7], the multi-views videos are first input into the

pre-trained convolutional neural network and recurrent neural network to extract the high-level spatio-temporal features. Then, the K-SVD algorithm to sparse the representation of the high-level features under multi-views and its corresponding dictionaries are obtained. At last, the softmax classifier is used to classify the actions.

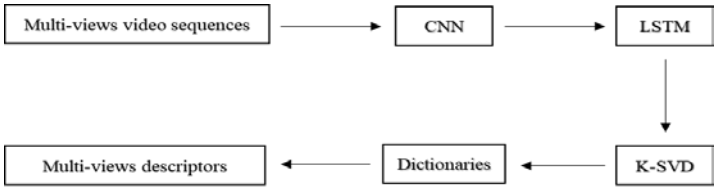


Figure 1. Flow chart of the model.

In the field of multi-views action recognition, researchers have conducted a great deal of researches. Imran et al. [11] proposed a self-similar matrix method which proposed that although the observation view is different, the movement distance of the key points of a certain action is similar in different views. Weinland et al. [12] evolved from two-dimensional motion history trajectories to three-dimensional motion history trajectories. The established three-dimensional motion history columns were got by Fourier transform to obtain the same view descriptor. Farhadi et al. [13] solved this multi-view problem by training a hidden model by proper initialization of the parameters. Yan et al. [14] combined with multi-view data to learn a 4D action model for action identification. Xiaofei Ji et al. [15] proposed to use the Hidden Markov model, and then apply multi-views to the multiple states of the traditional Hidden Markov Model respectively, and then use the hidden Markov's formula to model these states. However, all the above algorithms extract the low-level features of the human body and sometimes need to select some interest points manually. In this paper, the CNN and LSTM are used to extract the features, and finally the actions are classified and identified.

2. High-level feature extraction and representation

In this paper, the convolutional neural network (CNN) and the long short term memory (LSTM) are used to model and extract the spatio-temporal information. Every LSTM contains a memory structure (C_t in Equations 3 and 4) used to memorize the state. The purpose of C_t is to retain the important information during the forward and backward propagation of deep network.

This paper uses CNN to get the high-level features of the human body in each video frame. Firstly, we define the high-level features as x_t. Secondly, input the x_t to LSTM. The formulas for memory structure in LSTM can be defined as:

$$i_t = \sigma(W_i \bullet [h_{t-1}, x_t] + b_i) \tag{1}$$

$$f_t = \sigma(W_f \bullet [h_{t-1}, x_t] + b_f) \tag{2}$$

$$C_t^{'} = \tanh(W_c \bullet [h_{t-1}, x_t] + b_c) \tag{3}$$

$$C_t = f_t \circ C_{t-1} + i_t C_t^{'} \tag{4}$$

$$o_t = \sigma(W_o \bullet [h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t \circ \tanh(C_t) \tag{6}$$

Here, h_{t-1} is the output of LSTM at time t, C_t and x_t are the state information and input of the memory structure respectively at current time. i_t、f_t、o_t are the input gate, the forget gate and the output gate structure respectively. $\circ$ represent multiplies one by one, σ is the sigmoid function and

its equation is f(x)=1/(1+1/e-x). h_t is the output of LSTM, which is the feature that needs to be extracted eventually. The structure of $[h_{t-1}, x_t]$ establish contact with the current input x_t and h_{t-1} the previous time of LSTM. The input gate i_t also contains the output of the last time. h_t is $y_t(1 \le t \le N)$ in the action descriptor $Y_i(y_1, y_2, y_3, y_4, \cdots, y_N)_i$. The feature extraction model and its process are shown in Figure 2.

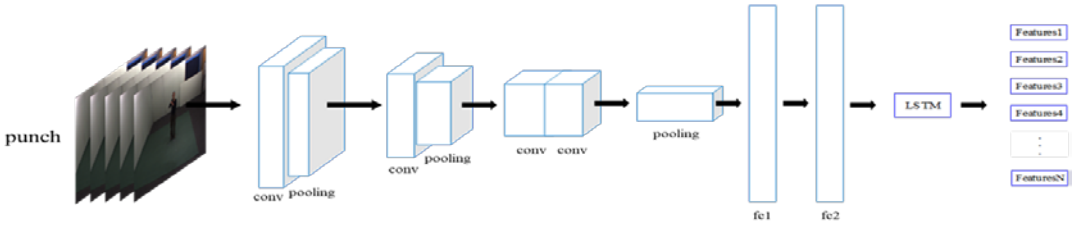


Figure.2 The extracted of high level features.

The CNN model uses a relatively simple AlexNet with fewer layers and superior effects [9]. The input of the LSTM layer is the output of the full connection layer. Then the outputs of LSTM layer are extracted, and the extracted features maps are one-dimensional vector $h_i = (f_1, f_2, f_3, \cdots, f_d)$, here, $i(1 \le i \le N)$, $f_j(1 \le j \le d)$ is float, d is the dimension of the data, N is the numbers of the video frames. Feature 1 to Feature N here correspond to one-dimensional vectors f_1 to f_d respectively. Here h_i is $h_i = Y_i(y_1, y_2, y_3, y_4, \cdots, y_N)_i$. The value of T can set according to the actual situation of their own datasets. Due to the limitation of the hardware of the laboratory, after several experiments we set T=5.

3. Sparse features and the establishment of the dictionary

This paper uses the K-SVD algorithm [17] to perform sparse representation of features. The function of K-SVD is $Y = DX$. $Y(Y \in (n * N))$ is the signal that needs to be sparsely represented, $D(D \in (n * K))$ is the overcompleted dictionary that needed to be obtained, $n < K$, $X(X \in (K * N))$ is the sparse coefficients corresponding to overcompleted dictionary. Therefore, our purpose is to find the sparse coefficient X and its corresponding dictionary D based on the given signal Y.

The features $Y_i(y_1, y_2, y_3, y_4, \cdots, y_N)_i$ extracted during the feature extraction stage as a signal to be sparsely represented by the K-SVD algorithm. The algorithm is divided into two steps: sparse representation and dictionary update. In a sparse representation, a dictionary D is first initialized and then a given Y is sparsely represented by the dictionary D. That is to say, the signal Y is represented as closely as possible with as few coefficients as possible. DX will be transformed as follows:

$$Y = DX = \sum_{j=1}^{K} d_j x_j \tag{7}$$

Here, d_j represents the jth column in dictionary D, x_j represents jth rows in coefficient matrix X,. The signal Y in this algorithm is a column vector. In the dictionary update phase, the objective function of K-SVD can be transformed into:

$$\begin{aligned} \left\| Y_i^T - DX \right\|_F^2 &= \left\| Y_i^T - \sum_{j=1}^{K} d_j x_T^j \right\|_F^2 \\ &= \left\| \left(Y_i^T - \sum_{j \ne k} d_j x_T^j \right) - d_k x_T^k \right\|_F^2 \\ &= \left\| E_i^K - d_k x_T^k \right\|_F^2 \end{aligned} \tag{8}$$

Here, $E_i^K = Y_i^T - \sum_{j \neq k} d_j x_T^j$, the F norm chosen here is the 2 norm in the paper, then do the SVD decomposition of E_i^K, $E_i^K = U \Delta V^T$, finally get the first column of U as d_k, $V \Delta(1,1)$ is x_T^k. After some iterations, we will get the the overcompleted dictionary D_i of the action. After overcompleted dictionary is obtained by K-SVD sparse algorithm, every action can be obtained by D_i, which is defined as D_i^α, α is less than or equal to the numbers of actions, Taking the IXMAS dataset as an example, $D_i^\alpha \in (D_1^\alpha, D_2^\alpha, ..., D_4^\alpha)$. So the action can be represented by this common dictionary. Then combine the D_i^α of multi-views to form an overcompleted dictionary, which is used as a descriptor of the action to train. The supercompleted dictionary of behaviors is shown in Table 1. Then the descriptor is used as an input vector, and sequentially input D_i^1, D_i^2, …, D_i^α into the neural network for training to obtain a view invariant action model, and finally use the softmax for classification and recognition.

Table 1. Supercompleted dictionary of human action.

Action 1	D_0^1	D_1^1	.....	D_4^1
Action 2	D_0^2	D_1^2	.....	D_4^2
.....	.....	.....	.....	.....
Action a	D_0^a	D_1^a	.....	D_4^a

4. Experiments and data analysis

4.1 Datasets

INRIA Xmas Motion Acquisition Sequences referred to IXMAS dataset, mainly used to study the multi-views behavior recognition. The IXMAS dataset has 13 daily actions, each of which is performed by 11 actors to perform three times. The dataset contains 5 views, extracting 23 frames per second. These actions include: 0: nothing, 1: check watch, 2: cross arms, 3: scratch head, 4: sit down, 5: get up, 6: turn around, 7: walk, 8: wave, 9: punch, 10: kick, 11: point , 12: pick up, 13 and 14: throw. We used 1th to 14th actions in the experiment, 13 and 14 as the same action, and therefore, we performed 13 actions in the final experiment.

The WVU dataset contains eight views of human action, which are performed by 10 actors three times. The behavior of these eight perspectives is shown in Fig. 5. Experiments were conducted with six actions selected from the IXMAS dataset for training and testing including: 0: waving, 1: punch, 2: jogging, 3: kicking, 4: picking-up, 5: throw.

4.2 Experimental design

This paper uses the leave-one-out training and validation methods. According to the advices of the two datasets, The training and validation datasets are classified as follows: the IXMAS dataset was trained using five views of 11 individuals and was validated using the remaining one's action, and the same strategy is used for the WVU dataset. In the use of deep learning, this experiment in the Ubuntu operating system, using the caffe. In the training process, the initial learning rate for this experiment was 0.0001, 0.1 times reduction for 20,000 iterations. The initialization methods for the convolutional layers and the fully connectioned layers are MSRA and Xavier respectively.

4.3 Experimental results and data comparison

For the IXMAS dataset, the experimental accuracy shown in Table 2. It can be seen from the table that the final recognition rate is the highest in the camera3 view, the worst recognition rate is camera4 view. The reason for this problem is because camera3 view can basically be regarded as a positive perspective, therefore, the highest recognition rate in this view. The camera4 view for looking down from the view of the actor's head to capture the action, so some actions block the camera above, and also IXMAS dataset simply uses RGB images to extract features, resulting in poor recognition rate. In the experiment, we conducted comparisons of some models, as shown in Table 3. From the table, we can see that the model presented in this paper has been greatly improved compared to some other models [12, 15, 17], but the algorithm in the [6] is still more obvious difference, we will do some improvement in the future.

For the WVU dataset, the experiment uses the same experimental method, and the average recognition rate under each perspective is shown in Table 4. Due to some actions in different views to block the sights of cameras, resulting in a different view of the recognition rate quite different. Just as the data from the right of Table 5, the presented method reaches a good recognition rate of 91.4% in the WVU dataset. It can be seen from the Table 5 that the K-SVD sparse algorithm plays a great role in our multi-view dataset, and the recognition rate has been greatly improved.

Table 2. The accuracy of every view in the IXMAS datasets.

Action	Cam0	Cam1	Cam2	Cam3	Cam4
Check watch	95.13	95.76	96.8	95.24	71.76
Cross arms	82.94	92.07	86.33	91.27	87.02
Scratch head	85.41	96.35	96.4	100	81.83
Sit down	85.56	86.55	88.45	97.12	83.67
Get up	74.99	82.93	78.43	93.19	75.71
Turn around	95.17	100	96.77	100	96
Walk	94.08	93.25	99.61	95.25	92.08
Wave	100	97.15	96	96.87	85.73
Punch	65.21	71.75	72.1	82.6	75.21
Kick	100	78	86	92	81
Point	96.87	98.44	98.44	93.75	81.27
Pick up	85	93.33	100	100	100
Throw	69.21	84.1	82.32	90.36	75.77
average	86.89	89.78	90.59	94.43	84.4

Table 3. Comparison in the different algrithms.

Models	Cam0	Cam1	Cam2	Cam3	Cam4	Average
Weinland et al	86.7	89.9	86.4	87.6	66.4	83.4
Zheng et al	99.4	99.8	99.4	99.7	93.6	98.2
Liu et al	76.7	73.3	72.0	73.0	N/A	73.8
Junejo et al	74.8	74.5	74.8	70.6	61.2	71.2
ours	86.7	89.8	90.6	94.4	84.4	89.2

Table 4. Accuracy of different views of the WVU datasets.

Model	Cam1	Cam2	Cam3	Cam4	Cam5	Cam6	Cam7	Cam8
Ours	87.6	93.6	90.0	93.2	90.1	90.3	91.0	92.2

Table 5. Comparison of test accuracy with/without K-SVD algorithm between two datasets.

Models	Average（IXMAS）	Average（WVU）
CNN+LSTM	60.0	51.2
CNN+LSTM+KSVD	89.2	91.4

5. Conclusion

This paper uses the deep learning to deal with the behavior of multi-views. Firstly, the spatio-temporal information is modeled by CNN and LSTM. Secondly, at the time of multi-views processing, using a sparse coding approach, has resulted in a two-part step. Finally, we use the softmax for classification and recognition. We can try to use deep learning approach to deal with the multi-views integration, from the beginning of the feature extraction to the final recognition into a black box, this training will also save a lot of time. In this paper, the experiment still uses the RGB images. In the future, we can try to use the depth (D-RGB) images to add depth information while using pixel information to improve the final recognition rate.

Acknowledgments.
This work was financially supported by China Nature Science Fund Project 61472196, 61672305 and Shandong Nature Science Fund Project ZR2015FM012.

References

[1] Messing, R. Pal, C. Kautz, H, Activity recognition using the velocity histories of tracked key points. Proc of 12th IEEE Int Conf on Computer Vision Piscataway. NJ: IEEE. 104—111 (2009).
[2] Wang Heng. Klaser, A. Schmid, C. et al, Dense trajectories and motion boundary descriptors for action recognition. International Journal of Computer Vision. 103(1): 60—79 (2013).
[3] Karpathy, A. Toderici, G. Shetty, S. et al, Large-Scale Video Classification with Convolutional Neural Networks. Computer Vision and Pattern Recognition. IEEE.1725—1732 (2014).
[4] Feichtenhofer, C. Pinz A. Zisserman A, Convolutional Two-Stream Network Fusion for Video Action Recognition. Computer Vision and Pattern Recognition. IEEE. 1933—1941 (2016).
[5] Liu, J. Shah, M. Kuipers, B. et al, Cross-view action recognition via view knowledge transfer. Computer Vision and Pattern Recognition. IEEE. 3209—3216 (2011).
[6] Zheng, J. Jiang, Z. Phillips, P J. et al, Cross-View Action Recognition via a Transferable Dictionary Pair. Bmvc. 25(6) (2012).
[7] Tong Hao. Dan Wu. Qian Wang. et al, Multi-view representation learning for multi-view action recognition. Journal of Visual Communication & Image Representation. 48 (2017).
[8] Marszalek, M. Laptev, I. Schmid, C, Actions in context. Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE. 2929—2936 (2009).
[9] Rodriguez, M, D. Ahmed, J. Shah, M, Action MACH a spatio-temporal Maximum Average Correlation Height filter for action recognition. Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE. 1—8 (2008).

[10] Liu, J. Luo, J. Shah, M, Recognizing realistic actions from videos. Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE. 1996—2003 (2009).
[11] Imran, N, Junejo. Emilie Dexter. Ivan Laptev. and Patrick Pe´rez, View-Independent Action Recognition from Temporal Self-Similarities. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 33, NO. 1, JANUARY (2011).
[12] Weinland, D. Ronfard, R. Boyer, E, Free viewpoint action recognition using motion history volumes. Computer Vision and Image Understanding. 104(2):249—257 (2006).
[13] Farhadi, A. Tabrizi, M, K. Endres, I. et al, A latent model of discriminative aspect. IEEE, International Conference on Computer Vision. IEEE. 948—955 (2009).
[14] Yan, P. Khan, S, M. Shah, M, Learning 4D action feature models for arbitrary view action recognition. Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE. 1—7 (2008).
[15] Xiaofei Ji. Zhaojie Ju. Ce Wang. Changhui Wang, Multi-view transition HMMs based view-invariant human action recognition method. Multimed Tools Appl. 75:11847—11864 (2016).
[16] Jozefowicz. Rafal. W, Zaremba. and I, Sutskever, An empirical exploration of recurrent network architectures. International Conference on International Conference on Machine Learning JMLR.org. 2342—2350 (2015).
[17] Liu, J. Shah, M. Kuipers, B. et al, Cross-view action recognition via view knowledge transfer. Computer Vision and Pattern Recognition. IEEE. 3209—3216 (2011).