Computers, Materials & Continua CMC, vol.63, no.3, pp.1081-1091, 2020

3-Dimensional Bag of Visual Words Framework on Action Recognition

Shiqi Wang[1], Yimin Yang[1,*], Ruizhong Wei[1] and Qingming Jonathan Wu[2]

Abstract: Human motion recognition plays a crucial role in the video analysis framework. However, a given video may contain a variety of noises, such as an unstable background and redundant actions, that are completely different from the key actions. These noises pose a great challenge to human motion recognition. To solve this problem, we propose a new method based on the 3-Dimensional (3D) Bag of Visual Words (BoVW) framework. Our method includes two parts: The first part is the video action feature extractor, which can identify key actions by analyzing action features. In the video action encoder, by analyzing the action characteristics of a given video, we use the deep 3D CNN pre-trained model to obtain expressive coding information. A classifier with subnetwork nodes is used for the final classification. The extensive experiments demonstrate that our method leads to an impressive effect on complex video analysis. Our approach achieves state-of-the-art performance on the datasets of UCF101 (85.3%) and HMDB51 (54.5%).

Keywords: Action recognition, 3D CNNs, recurrent neural networks, residual networks, subnetwork nodes.

1 Introduction

Video action recognition is the basic building block in various applications such as video retrieval, natural human-machine interaction, video surveillance, and digital entertainment [Liu, Su, Nie et al. (2017); Herath, Harandi and Porikli (2017); Song, Yu, Zhao et al. (2019)]. In action recognition, there are two important and complementary aspects: appearance and dynamics. Video often has some complex factors, such as camera motion, scale change and viewpoint change. Therefore, whether the action recognition system can extract and utilize the relevant feature information is the key to its performance. However, it is not easy to extract features effectively. Therefore, the question of how to design a network structure to deal with these problems and retain classified information becomes crucial.

The recent rise of recurrent neural networks (RNNs) have been successfully applied to action recognition [Donahue, Anne, Guadarrama et al. (2015); Li, Qiu, Yao et al. (2016)]. Existing 3D motion recognition methods based on RNN are mainly used for time-domain modeling of long-term context information, representing the dynamic based on motion.

[1] Department of Computer Science, Lakehead University, Thunder Bay, Canada.

[2] Department of Electrical and Computer Engineering, University of Windsor, Windsor, Canada.

[*] Corresponding Author: Yimin Yang. Email: yyang48@lakeheadu.ca.

Received: 13 January 2020; Accepted: 26 February 2020.

However, in the spatial domain, there are also strong dependencies between nodes. For 3D action recognition tasks, the spatial configuration of nodes in video frames may be very recognizable. Liu et al. [Liu, Shahroudy and Xu (2016)] proposed a spatial-temporal long short-term memory (ST-LSTM) network and achieved a good performance.

The BoVW framework has recently been used in motion recognition with good results. This framework includes two parts: feature extractor and classifier. Most of the BoVW models adopt Fisher vectors of improved dense trajectories [Wang and Schmid (2013); Wang, Qiao and Tang (2016); Fernando, Gavves, Oramas et al. (2017)] or CNN features [Wang, Xiong, Wang et al. (2016); Wang, Qiao and Tang (2015)] with a classifier such as support vector machine (SVM), and achieve reliable results on pre-segmented video datasets, such as UCF-101 [Soomro, Zamir and Shah (2012)] and HMDB51 [Kuehne, Jhuang, Carrole et al. (2011)].

In action recognition field, 3D CNNs have recently been more effective than the CNNs with two-dimensional (2D) kernels [Carreira and Zisserman (2017)]. Recently, 3D CNNs have been used in accurate action recognition. However, the 2D model still has strong associations with video data. Even well-organized 2D models [Tran, Bourdev, Fergus et al. (2015); Varol, Lapttev and Schmid (2016)] cannot overcome the advantages of 2D CNNs combining stacked flow with RGB images [Simonyan and Zisserman (2014)], mainly because video datasets usually have small data-scales, preventing optimization of a large number of parameters in 3D CNNs cannot be optimized. In addition, 3D CNNs can only be trained on a video data set from scratch, while 2D CNNs can be pre-trained on ImageNet. Recently, Carreira et al. [Carreira and Zisserman (2017)] trained 3D CNNs on Kinetics dataset and boomed the performance, which also made it possible for us to use a 3D pre-trained model. Thus, we can now use a Kinetics datasets pre-trained model to perform our action recognition.

In this paper we propose a BoVW framework. Our network contains two parts, a feature extractor and a classifier (Fig. 1). We use a 3D residual networks (ResNet) He et al. [He, Zhang, Ren et al. (2016)] pre-trained model as our feature extractor and for the classifier we proposed a single layer feedforward network with subnetwork nodes (SLFN). We tested the ResNet model of different structures from a shallower to a deeper network model using the UCF-101 and HMDB-51 datasets in order to ascertain which structure has the best performance. Additionally, we also tested the feasibility of the pre-trained model. We optimized our SLFN parameters to achieve a better performance. In addition, we evaluated the approach we proposed in terms of accuracy and time consumption. Furthermore, other classifiers, such as SVM, can be used in the method as well. Our proposed method could use any type of videos. The proposed framework is summarized in the following section.

2 Method

2.1 Network structure

The BoVW structure plays a powerful role in image recognition. The general idea of BOVW is to reduce the dimensions of an image or video and encode it into a set of features. Features comprise key points and descriptors. The key points are the "salient points" in the image, so the key points are the same whether the image is rotated, shrunk,

or expanded. A descriptor is a description of the key points. So we can represent each image in terms of the frequency of its features, and by virtue of its feature frequency, we can predict the category of another image. In order to explore whether this structure is used in the field of action recognition, we built our network, see Fig. 1 below, and tested the performance.

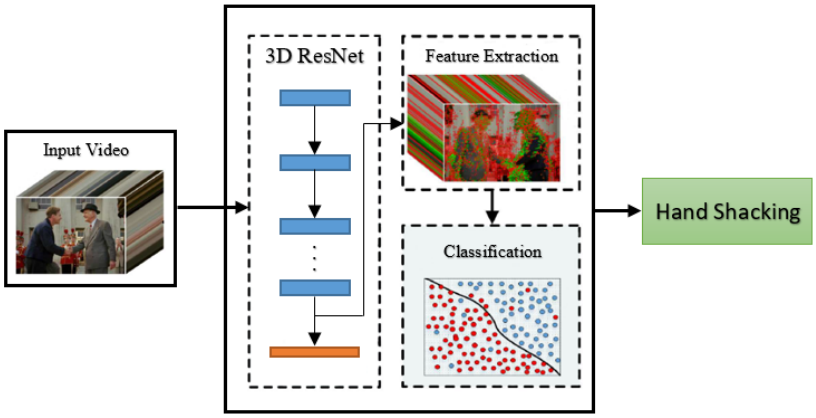

Figure 1: Proposed framework including feature extractor and classifier

2.1 3D ResNet

For the extractor, we focused on 3D CNNs that have begun to perform better than 2D CNNs on large-scale video datasets. Recently, Hara et al. [Hara, Kataoka and Satoh (2018)] conducted a series of experiments using different depth 3D ResNet. They also compared the performance between a base model and pertrained model. Their experiments showed that the ResNet-101 pre-trained model demonstrated the best performance.

In our study, based on those experiments results provided by Hara et al. [Hara, Kataoka and Satoh (2018)], we choose the 3D ResNet pre-trained model as our feature extractor. To ensure the accuracy of the results, we reevaluated all experiments. We also tested the performance of 3D ResNet with different depths. The results of our experiment are shown in the following section.

2.2 Feature extraction from 3D ResNet models

As it was mentioned in the previous subsection, the 3D ResNet models have been considered. This model was previously trained on a Kinetics dataset with obtained impressive results. ResNet is known to be a deep CNN model that consists entirely of several convolution layers but only one fully connected layer. Average-pooling layers are employed after convolution layers. As shown in Fig. 2 we extracted deep features from the average pool layer.

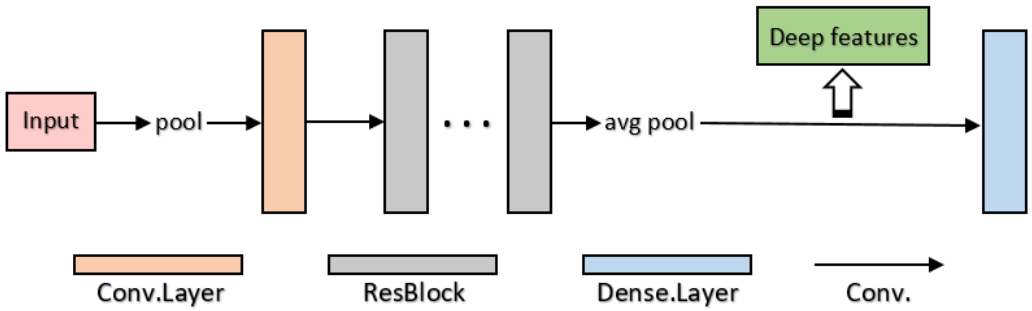

Figure 2: Architecture of the 3D ResNet

2.3 Single-layer classifier with sub-network nodes

In video data processing, time cost and computation cost are often relatively high, so our proposed new classifier greatly reduces time and computational cost compared to traditional classifiers such as SVM and ELM, and iterative training tends to achieve better results. Fig. 3 shows the structure of our classifier.

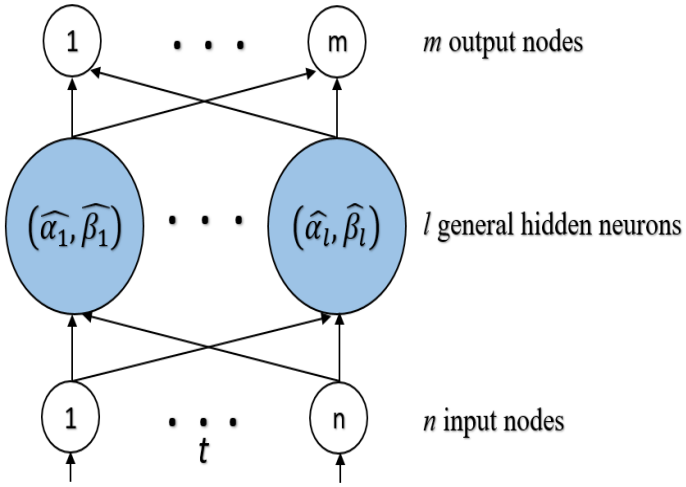

Figure 3: Our proposed classifier

As we have encoded features data, we will use the proposed SLFN on the encoded dimension data to classify objects with L numbers of subnetworks. Thereafter, we will split the data along with target label data with n size and send it to the network to train it. First, we will use (x_0, t_0) chunk of data for the initial training of the network. Thereafter, e_i will represent the residual network error and $(\widehat{\boldsymbol{\alpha}_l}, \widehat{\boldsymbol{\beta}_l})$ would define the input weight and output weight which will be updated in every iteration.

$$\boldsymbol{\alpha}(0) = \mathbf{x}_0^{-1} \cdot h^{-1}(\boldsymbol{e}_0) = (\mathbf{x}_0^T \mathbf{x}_0 + (I_{d\times d}/c))^{-1} \mathbf{x}_0^T h^{-1}(\boldsymbol{e}_0) \quad (1)$$

We will use $\mathbf{m}_0$ to represent $\mathbf{x}_0^{\mathrm{T}} \mathbf{x}_0 + (I_{d\times d}/c)$. Now we can write Eq. (1) in following way:

$$\boldsymbol{\alpha}(0) = \mathbf{m}_0 \mathbf{x}_0^T h^{-1}(\boldsymbol{e}_0) \tag{2}$$

After the initial training, we will update the input weight in a sequential manner with the next batch of training samples $(\mathbf{x}_i, t_i)$, we combine $\mathbf{x}_0$ and $\mathbf{x}_1$ together as well as their corresponding residual error $\mathbf{e}_0$ and $\mathbf{e}_1$; theoretically, we can get the following:

$$\widehat{\boldsymbol{\alpha}_1} = \mathbf{m}_1^{-1} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \end{bmatrix}^T \begin{bmatrix} h^{-1}(\mathbf{e}_0) \\ h^{-1}(\mathbf{e}_1) \end{bmatrix} \tag{3}$$

where

$$\begin{cases} \mathbf{m}_1 = I_{d\times d}/c + \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \end{bmatrix} \\ \mathbf{m}_1 = I_{d\times d}/c + \mathbf{x}_0^T \mathbf{x}_0 + \mathbf{x}_1^T \mathbf{x}_1 \\ \mathbf{m}_1 = \mathbf{m}_0 + \mathbf{x}_1^T \mathbf{x}_1 \end{cases} \tag{4}$$

and

$$\begin{cases} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \end{bmatrix}^T \begin{bmatrix} h^{-1}(\mathbf{e}_0) \\ h^{-1}(\mathbf{e}_1) \end{bmatrix} = \mathbf{x}_0^T h^{-1}(\mathbf{e}_0) + \mathbf{x}_1^T h^{-1}(\mathbf{e}_1) \\ \qquad = \mathbf{m}_0 \mathbf{m}_0^{-1} \mathbf{x}_0^T h^{-1}(\mathbf{e}_0) + \mathbf{x}_1^T h^{-1}(\mathbf{e}_1) \\ \qquad = \mathbf{m}_0 \boldsymbol{\alpha}_{(0)} + \mathbf{x}_1^T h^{-1}(\mathbf{e}_1) \\ \qquad = (\mathbf{m}_1 - \mathbf{x}_1^T \mathbf{x}_1) \boldsymbol{\alpha}_{(0)} + \mathbf{x}_1^T h^{-1}(\mathbf{e}_1) \\ \qquad = \mathbf{m}_1 \boldsymbol{\alpha}_{(0)} - \mathbf{x}_1^T \mathbf{x}_1 \boldsymbol{\alpha}_{(0)} + \mathbf{x}_1^T h^{-1}(\mathbf{e}_1) \end{cases} \tag{5}$$

According to Eqs. (3)-(5), we derive

$$\begin{cases} \widehat{\boldsymbol{\alpha}_1} = \mathbf{m}_1^{-1} [\mathbf{m}_1 \boldsymbol{\alpha}_{(0)} - \mathbf{x}_1^T \mathbf{x}_1 \boldsymbol{\alpha}_{(0)} + \mathbf{x}_1^T h^{-1}(\mathbf{e}_1)] \\ \quad = \boldsymbol{\alpha}_{(0)} - \mathbf{m}_1^{-1} \mathbf{x}_1^T \mathbf{x}_1 \boldsymbol{\alpha}_{(0)} + \mathbf{m}_1^{-1} \mathbf{x}_1^T h^{-1}(\mathbf{e}_1) \\ \quad = \boldsymbol{\alpha}_{(0)} + \mathbf{m}_1^{-1} \mathbf{x}_1^T [h^{-1}(\mathbf{e}_1) - \mathbf{e}_1 \boldsymbol{\alpha}_{(0)}] \end{cases} \tag{6}$$

We can generalize Eq. (6) to

$$\widehat{\boldsymbol{\alpha}_{i+1}} = \boldsymbol{\alpha}_{(i)} + \mathbf{m}_{i+1}^{-1} \mathbf{x}_{i+1}^T [h^{-1}(\mathbf{e}_{i+1}) - \mathbf{x}_{i+1} \boldsymbol{\alpha}_{(i)}] \tag{7}$$

Instead of calculating $\widehat{\boldsymbol{\alpha}_n}$ for each epoch, we can use the previous knowledge and update it by passing new chunk of encoded. Suppose the chunk of data we send for initial training is $\mathbf{x}_{en\_init}$ and remaining data is considered as $\mathbf{x}_{en\_seq}$. If total numbers of training epochs are *TOTAL_EPOCHS* and batch size for sequential data is *BATCH_SIZE*. We will train our network in the following manner:

Algorithm OS-Subnetwork training algorithm

Result: Update α_L sequentially and calculate corresponding β_L

Split the dataset for initial and sequential training

epoch ← 0;

For ($\mathbf{x}_{en\_init}, \mathbf{t}_{en\_init}$), we obtain $\boldsymbol{\alpha}_L$ and $\boldsymbol{\beta}_L$ for each subnetwork

while *epoch* < *TOTAL_EPOCHS* **do**

  l ← *0;*

  while *l* < *length*(x_{en_seq}) **do**

    if *l* + *BATCH_SIZE* ≤ *length*(x_{en_seq}) **then**

      $\mathbf{x}_{batch} \leftarrow \mathbf{x}_{en\_seq}[l: l + BATCH\_SIZE]$;

      $\mathbf{t}_{batch} \leftarrow \mathbf{t}_{en\_seq}[l: l + BATCH\_SIZE]$;

    else

      $\mathbf{x}_{batch} \leftarrow \mathbf{x}_{en\_seq}[l:]$;

      $\mathbf{t}_{batch} \leftarrow \mathbf{t}_{en\_seq}[l:]$;

    end

    l ← *l* + *BATCH_SIZE;*

    Update sequentially $\boldsymbol{\alpha}_L$ and calculate $\boldsymbol{\beta}_L$;

  end

  epoch ← *epoch* + 1*;*

end

The performance of the proposed algorithm depends on both the number of hidden neurons m (encoding dimension) and the pre-defined constant ***c***. A proper way of selecting the optimal value of ***c*** is chosen by the trial-and-error method [Huang, Zhou, Ding et al. (2012)]. Now, we will use the Online Sequential-Subnetwork on encoded dimension data to classify objects with ***L*** numbers of subnetworks.

3 Experiment results

3.1 Datasets

The HMDB-51 Kuehne et al. [Kuehne, Jhuang and Garrote (2011)] and UCF-101 [Soomro, Zamir and Shah (2012)] datasets are currently the most successful in the field of action recognition. UCF101 has a total of 13,320 videos, including 101 action categories. Moreover, video in this dataset has diverse actions, with very different camera movements and often messy background. It is one of the most challenging data sets available. The videos in 101 action categories are divided into 25 groups, where each group can consist of 4-7 videos of an action. The videos from the same group may share some common features, such as similar background, similar viewpoint and so on. The

HMDB51 dataset contains 6849 clips divided into 51 action categories, each containing a minimum of 101 clips. Fig. 4 shows samples from the video frames of datasets utilized to evaluate our method performance.

Figure 4: The samples of video frames from UCF-101 and HMDB-51 datasets

In the training and testing process, it is very important to separate the video belonging to the same group. Since videos within a group are all from a single long video, sharing videos from the same group in the training set and testing set can achieve higher performance. So, each of these datasets provide train and test splits files. We evaluated our network performance based on these splits files and calculated average performance.

3.2 Environment setting

To test our proposed method, we compare our work with different depth 3D ResNet and classifiers; we also compare our proposed method with other state-of-the-art methods. To find the best performance of our extractor, we compared it with pre-trained 3D ResNet with different depth and learning from scratch. We compared the classifier with support vector machine (SVM), extreme learning machine (ELM), K-nearest neighbors (KNN) and random forest (RF) [Breiman (2011)]. For SVM and ELM, parameters were set up from $[2^{-10}, 2^{-9}, \ldots, 2^{10}]$ in each experiment. For KNN we set K to 3 and used auto algorithm. We selected 100 as our RF estimators. For our proposed classifier, the regularization parameter C was selected from $C \in \{2^{-4}, \ldots, 2^{8}\}$. To test the efficiency of our algorithm, we run our experiments on two datasets, which are conducted on a machine with an NVidia GTX-1080Ti GPU.

3.3 Extractor evaluation

According to a previous study Hara et al. [Hara, Kataoka and Satoh (2018)], 3D ResNet trained on UCF-101 and HMDB-51 does not achieve high accuracy whereas a Kinetics pre-trained model works well. In this section, aiming to find the optimal feature extractor, we tried to reproduce the performance in the experiment. In this process, we trained 3D ResNets with different depths by UCF-101 and HMDB-51 dataset from scratch and then we trained kinetics pre-trained 3D ResNet models as well. To make the result fairly, we use train and test split file 1; choose batch size of 32, and train 50 epochs. The performances are shown in Tab. 1.

Table 1: Performances of 3D ResNets

Model	UCF-101	HMDB-51
Learning from scratch		
3D ResNet-18	35.9	10.1
3D ResNet-34	33.3	12.5
3D ResNet-50	34.5	12.8
3D ResNet-101	38.1	14.1
Transfer learning		
3D ResNet-18	71.4	41.6
3D ResNet-34	76.4	44.3
3D ResNet-50	75.6	46.4
3D ResNet-101	**78.9**	**47.8**

As Tab. 1 shows kinetics pre-trained models perform significantly better then learning from scratch. 3D ResNet-101 has the best performance both for learning from scratch and transfer learning. It indicates that the 3D ResNet-101 pre-trained model can learn optimal features more accurately in less time compared to other methods. Therefore, we choose it as our extractor and use it in later experiments.

3.4 Classifier evaluation

After the experiment above, we choose a 3D ResNet-101 pre-trained model as our extractor. In this section we evaluated our proposed classifier's performance. A performance comparison has been evaluated among SVM, ELM, KNN, RF and our proposed algorithm. For the accuracy and fairness of the experiment, we first trained an extractor and extracted deep features; later, the classifier recognized the actions. Further, we carried out this experiment in Tab. 2 to compare our single-layer network with those learning algorithms. Here, it can be seen that the accuracy of our method is clearly higher compared that of other classifiers.

Table 2: Performances of different classifiers

Model	UCF-101	HMDB-51
3D ResNet-101+SVM	83.3	50.8
3D ResNet-101+ELM	84.7	53.7
3D ResNet-101+KNN	82.3	48.6
3D ResNet-101+RF	43	39.1
3D ResNet-101+***Ours***	**85.3**	**54.5**

We also evaluated the time consumption, and the comparison results of time consumption are shown in Tab. 3, which indicates that our classifier has seen a significant improvement in training speed compared to SVM. Further, the ELM training speed is similar. However, our classifier supports iterative training, and the corresponding batch

size can be set according to the situation, which is especially important in video processing and in especially large video datasets.

Table 3: Time consumptions of different classifier (s)

Model	UCF-101	HMDB-51
SVM	486	116
ELM	10	3.7
KNN	567	85
RF	18.6	4.8
Ours	**2.2**	**0.96**

3.5 Framework evaluation

The above experiments show that a kinetics dataset can be used to train our network. Comparisons with other state-of-the-art architectures are shown in Tab. 4. As can be seen from Tab. 4, our method achieved higher accuracies compared with 3D Resnet-101, 3D ResNext-101 [Hara, Kataoka and Satoh (2018)], C3D [Tran, Bourdev, Fergus et al. (2015)], P3D [Qiu, Yao and Mei (2017)], and two-stream I3D [Carreira and Zisserman (2017)]. We can also observe that two-stream I3D, which is pre-trained by the Kinetics dataset, achieves the best accuracy. In addition, we believe that combining the two-stream architecture with our framework can further improve the accuracy of two-stream I3D.

Table 4: Performances of different classifiers

Model	UCF-101	HMDB-51
3D ResNet-101	78.9	10.1
3D ResNeXt-101	81.4	50.3
C3D	78.1	-
I3D	80.2	-
Two-stream I3D	**92.5**	**63.7**
Ours	85.3	54.5

4 Conclusion

In this paper, we tested various CNNs architectures of spatio-temporal three-dimensional convolution kernels on the current video dataset. According to these experimental results, the following conclusions can be drawn: (1) The BoVW structure is efficient on video processing. (2) It is effective for 3D CNNs to pre-train on the Kinetics dataset, which has sufficient data to optimize the 3D CNN network. (3) Instead of using randomized input weights, we can approach a classifier where weights would be configured by calculation and reach to the steepest descent in iterative manner without configuring the learning rate.

Funding Statement: The author(s) received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Breiman, L. (2001): Random forests. *Machine Learning*, vol. 45, no. 1, pp. 5-32.

Carreira, J.; Zisserman, A. (2017): Quo vadis, action recognition? A new model and the kinetics dataset. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4724-4733.

Donahue, J.; Anne, H. L.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S. et al (2015): Long-term recurrent convolutional networks for visual recognition and description. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2625-2634.

Feichtenhofer, C.; Pinz, A.; Wildes, R. (2016): Spatiotemporal residual networks for video action recognition. *Advances in Neural Information Processing Systems*, pp. 3468-3476.

Feichtenhofer, C.; Pinz, A.; Wildes, R. (2017): Spatiotemporal multiplier networks for video action recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4768-4777.

Feichtenhofer, C.; Pinz, A.; Zisserman, A. (2016): Convolutional two-stream network fusion for video action recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1933-1941.

Fernando, B.; Gavves, E.; Oramas, J.; Ghodrati, A.; Tuytelaars, T. (2017): Rank pooling for action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 773-787.

Hara, K.; Kataoka, H.; Satoh, Y. (2018): Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet? *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6546-6555.

He, K.; Zhang, X.; Ren, S.; Sun, J. (2016): Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778.

Herath, S.; Harandi, M.; Porikli, F. (2017): Going deeper into action recognition: a survey. *Image and Vision Computing*, vol. 60, pp. 4-21.

Huang, G. B.; Zhou, H.; Ding, X.; Zhang, R. (2012): Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems*, *Man and Cybernetics, Part B*, vol. 42, no. 2, pp. 513-529.

Kuehne, H.; Jhuang, H.; Garrote, E.; Poggio, T.; Serre, T. (2011): HMDB: a large video database for human motion recognition. *International Conference on Computer Vision*, pp. 2556-2563.

Li, Q.; Qiu, Z.; Yao, T.; Mei, T.; Rui, Y. et al. (2016): Action recognition by learning deep multi-granular spatio-temporal video representation. *Proceedings of the ACM on International Conference on Multimedia Retrieval*, pp. 159-166.

Liu, A.; Su, Y.; Nie, W.; Kankanhalli, M. (2017): Hierarchical clustering multi-task learning for joint human action grouping and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 1, pp. 102-114.

Liu, J.; Shahroudy, A.; Xu, D.; Wang, G. (2016): Spatio-temporal LSTM with trust gates for 3D human action recognition. *European Conference on Computer Vision*, pp. 816-833.

Qiu, Z.; Yao, T.; Mei, T. (2017): Learning spatio-temporal representation with pseudo-3D residual networks. *Proceedings of the International Conference on Computer Vision*, pp. 5533-5541.

Simonyan, K.; Zisserman, A. (2014): Two-stream convolutional networks for action recognition in videos. *Advances in Neural Information Processing Systems*, pp. 568-576.

Song, W.; Yu, J.; Zhao, X.; Wang, A. (2019): Research on action recognition and content analysis in videos based on DNN and MLN. *Computers*, *Materials & Continua*, vol. 61, no. 3, pp. 1189-1204.

Soomro, K.; Zamir, A. R.; Shah, M. (2012): UCF101: a dataset of 101 human actions classes from videos in the wild. arXiv:1212.0402.

Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; Paluri, M. (2015): Learning spatiotemporal features with 3D convolutional networks. *Proceedings of the International Conference on Computer Vision*, pp. 4489-4497.

Varol, G.; Laptev, I.; Schmid, C. (2016): Long-term temporal convolutions for action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1510-1517.

Wang, H.; Schmid, C. (2013): Action recognition with improved trajectories. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3551-3558.

Wang, L.; Qiao, Y.; Tang, X. (2016): MoFAP: a multi-level representation for action recognition. *International Journal of Computer Vision*, vol. 119, no. 3, pp. 254-271.

Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D. et al. (2016): Temporal segment networks: towards good practices for deep action recognition. *European Conference on Computer Vision*, pp. 20-36.

Wang, L.; Qiao, Y.; Tang, X. (2015): Action recognition with trajectory-pooled deep-convolutional descriptors. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4305-4314.

Yang, Y.; Wang, Y.; Yuan, X. (2012): Bidirectional extreme learning machine for regression problem and its learning effectiveness. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 9, pp. 1498-1505.