RECURRENT CONVOLUTIONAL NEURAL NETWORK FOR VIDEO CLASSIFICATION

Zhenqi Xu, Jiani Hu, Weihong Deng

Beijing University of Posts and Telecommunications
No. 10, Xitu Cheng Road, Haidian District, Beijing, China, 100876
xuzhenqi@bupt.edu.cn, jnhu@bupt.edu.cn, whdeng@bupt.edu.cn

ABSTRACT

Video classification is more difficult than image classification since additional motion feature between image frames and amount of redundancy in videos should be taken into account. In this work, we proposed a new deep learning architecture called recurrent convolutional neural network (RCNN) which combines convolution operation and recurrent links for video classification tasks. Our architecture can extract the local and dense features from image frames as well as learning the temporal features between consecutive frames. We also explore the effectiveness of sequential sampling and random sampling when training our models, and find out that random sampling is necessary for video classification. The feature maps from our learned model preserve motion from image frames, which is analogous to the persistence of vision in human visual system. We achieved 81.0% classification accuracy without optical flow and 86.3% with optical flow on the UCF-101 dataset, both are competitive to the state-of-the-art methods.

Index Terms— video classification, deep learning, recurrent convolutional neural network, action recognition

1. INTRODUCTION

Video classification is a challenge task which has drawn amount of attention in research area. Traditional video classification methods usually contains two stages: feature extraction and classification. Most methods consider exploring features both spatially and temporally. The hand-crafted features contains Histogram of Oriented Gradients (HOG) [1], Histogram of Optical Flow (HOF), gradient-based Motion Boundary Histogram (MBH) [2] and improved dense trajectories (IDT) [3]. The extracted features are then encoded using bag of words (BOW) or fisher vector to get a global video descriptor [4]. A discriminant classifier such as support vector machine (SVM) will give predictions on the encoded features.

Deep learning has been used broadly in many machine learning area due to its great performance recently. A good strategy to apply deep learning architecture in a task is to incorporate the prior knowledge of that task in the architecture. For example, convolutional neural network (CNN) for image tasks can utilize the image prior: pixels that are spatially nearby are highly correlated [5]. This local structure is extended to time dimension in videos, that is, the pixels in the same spatial position between consequent frames are also highly correlated. Pixels in different frames form motion (or temporal) features. This correlation induces a lot of redundancy in videos. Failure to deal with this redundancy may cause much computation and poor performance, which makes the video classification using deep learning methods difficult.

A simple way to drive image classification into video classification is to fuse the image features or scores to get a video prediction [6, 7]. This method ignores the motion features, since frames are treated to be independent. Stacking image frames as inputs also fails to model motion features, since the order of image frames is ignored. A good property is that this kind of methods can benefit from pre-training of large ImageNet dataset [8].

Recurrent neural network (RNN) is good at learning relations from sequence inputs [9]. Ng *et al.* [6] and Donahue *et al.* [10] use RNN with Long Short Term Memory (LSTM) units [11] as a fusion method to extract temporal features upon spatial CNN representations. We argue that it's not a good way to model motion, too. Because the goal of CNN architecture is to learn end-to-end features which is invariant to rotation and shifting [5], especially in the high level feature. And thus it will be invariant to motion. The better CNN model performs, the less motion remains in the CNN feature. Experiment results reported in [10] shows that lower feature (fc6) performs better than the last layer feature (fc7), which is consistent with our analysis. Another drawback is that it's hard to visualize the RNN feature, which is important for vision tasks.

Existing deep learning methods always recur to optical flow computed from consequent image frames to incorporate motion features [12]. The two-stream features are complementary to each other, thus fusion of optical flow and raw image frames on the score level can boost performance. This shows two things, firstly, the motion features are actually important for action recognition. Secondly, the existing architecture can not model the motion feature from raw images well.

To model spatial and motion feature implicitly in the same

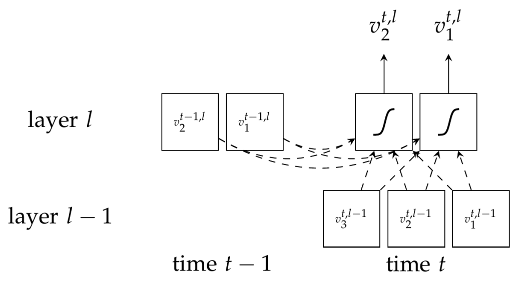


Fig. 1. Illustration of RCNN layer. It firstly sum the spatial inputs and temporal inputs, and then apply non-linear transformation.

time, Shuiwang Ji *et al.* [13] extended 2D convolution to 3D convolution. This method is computation costly, and induces lots of weights. The inefficiencies both in space and time make this method hard to scale to large models as well as large datasets. And it can be seen as a failure to reduce the redundancy in videos.

Our goal here is to fully use the prior knowledge of videos in designing our deep learning architecture. A good architecture for video classification should 1) utilize the local structure in image frames, 2) incorporate the motion features between image frames 3) reduce the redundancy of videos. We deal with video classification problem by extending traditional RNN to recurrent convolutional neural network (RCNN, see Fig. 1), which can learn local and dense feature through convolution operation as well as modelling motion between image frames by the linking of previous time step. The key point is that by doing this, we can learn motion features from low level where the motion feature exists. To reduce the redundancy, we use random sampling strategy and make a comparison with sequential sampling methods.

By applying convolution operation for recurrent neural networks, our architecture has several advantages. Firstly, it is suitable for vision tasks, since the image is locally relevant. Secondly, our architecture can deal with variable length inputs without cutting a video into several clips. Thirdly, our architecture can explore the motion feature both in the low level and in the high level. Lastly, we can visualize the motion features just like the spatial features, which is helpful for understanding the learned models and diagnosing the model when training. We achieve 81.0% accuracy on UCF-101 dataset [14] without using optical flow and 86.3% with optical flow.

2. METHODS

2.1. From RNN to RCNN

In traditional RNNs, each layer has a connection from outputs of the previous time step. The layer firstly do a linear transformation for both inputs from the previous time step and inputs from the present layer. Then sum of these two parts are through a non-linear function, usually sigmoid function, tanh function or recently rectified linear unit (ReLU). Formally, we denote $a^{t,l}$ as the linear combination of inputs, $v^{t,l}$ as activation of layer l at time step t, W^l as the weight matrix between layers, $\widehat{W}^l$ as the weight matrix between time steps, b^l as the bias term of layer l and $f(\cdot)$ as the element-wise activation function:

$$a^{t,l} = \begin{cases} W^l \times v^{t,l-1} + \widehat{W}^l \times v^{t-1,l} + b^l & t > 1 \\ W^l \times v^{t,l-1} + b^l & t = 1 \end{cases} \tag{1}$$

$$v^{t,l} = f(a^{t,l}) \tag{2}$$

The RNNs can model the relations from sequence inputs [9]. But the traditional RNN may not work well for videos, since it doesn't incorporate the prior knowledge that images are locally relevant. CNN can model this prior knowledge, but ignores the motion between image frames. The best way to deal with video tasks is to combine RNN and CNN, which forms our RCNN layer:

$$a_c^{t,l} = \begin{cases} \sum\limits_{i=1}^{m} W_{c,i}^l * v_i^{t,l-1} + \sum\limits_{j=1}^{n} \widehat{W}_{c,j}^l * v_j^{t-1,l} + b_c^l & t > 1 \\ \sum\limits_{i=1}^{m} W_{c,i}^l * v_i^{t,l-1} + b_c^l & t = 1 \end{cases} \tag{3}$$

$$v_c^{t,l} = f(a_c^{t,l}) \tag{4}$$

where subscripts c, i, j denotes index of feature maps in activation, spatial inputs and temporal inputs. $*$ denotes convolution operation. Matrix W means filters here. Figure 1 illustrate the operations. It should be noted that the output of spatial and temporal convolutions should have the same dimension, this can be achieved by appropriate padding and striding.

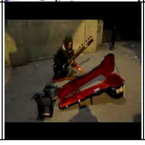
To make discussion simple and clean, we denote $\Phi_{W,b}(x)$ to be the non-linear mapping from inputs to the highest hidden layer outputs, with spatial parameters $W = \{W^l, for\ all\ hidden\ l\}$ and $b = \{b^l, for\ all\ hidden\ l\}$ to be parameters of the CNN model.

The existing image classification models can be extended to learn motion features in videos simply by replacing the CNN layer with proposed RCNN layer and replacing the fully connected layer with RNN layer. We denote $\widehat{\Phi}_{W,\widehat{W},b}^t(x^t, v^{t-1})$ to be the RCNN mapping at time step t, where $\widehat{W} = \{\widehat{W}^l, for\ all\ hidden\ l\}$ is the temporal parameters between time steps and $v^{t-1} = \{v^{t-1,l}, for\ all\ hidden\ l\}$ to be the activation from last time step. Specifically, $\widehat{\Phi}_{W,\widehat{W},b}^1(x^1, v^0) = \Phi_{W,b}(x^1)$.

2.2. Training – truncated BPTT

To optimize our RCNN architecture, we can use truncated back-propagation through time (truncated BPTT) [15] to up-

Table 1. Illustration of sequential sampling and random sampling.

Iteration	1	2	3	...	k	k+1	k+2
Sequential sampling				...			
Random sampling				...			Old Yang Style Tai Chi Long Form www.taiji.net Paul Brecher 2002

date the parameters of the network. The back-propagation of RCNN layer can be done as follows:

$$\frac{\partial L^t}{\partial v_i^{t,l-1}} = \sum_{j=1}^{n} f'(a_j^{t,l}) \times \frac{\partial L^t}{\partial v_j^{t,l}} * W_{j,i}^{l} \quad (5)$$

$$\frac{\partial L^t}{\partial v_i^{t-1,l}} \overset{truncate}{=\!=\!=\!=} 0 \quad (6)$$

$$\frac{\partial L^t}{\partial W_{c,i}^{l}} = f'(a_c^{t,l}) \times \frac{\partial L^t}{\partial v_c^{t,l}} * v_i^{t,l-1} \quad (7)$$

$$\frac{\partial L^t}{\partial \widehat{W}_{c,i}^{l}} = f'(a_c^{t,l}) \times \frac{\partial L^t}{\partial v_c^{t,l}} * v_i^{t-1,l} \quad (t > 1) \quad (8)$$

$$\frac{\partial L^t}{\partial b_c} = f'(a_c^{t,l}) \times \frac{\partial L^t}{\partial v_c^{t,l}} \quad (9)$$

where L^t is the loss of time step t. The multiply operation here is element-wise multiply, and the convolution operation should be padded and stridden appropriately based on the forward convolution.

The truncated BPTT algorithm does not compute exact gradient, but works well in practice. Here, we ignore the gradient in time directions (see formula 6), by doing this, we can easily integrate our architecture with caffe toolbox [16] as well as using a single GPU to boost training speed. The good property of truncated BPTT is that it's totally online. The image frames can be fed into the model continuously, and the model can deal with arbitrary lengths of inputs.

2.3. Training – random sampling

When using the truncated BPTT algorithm to optimize our RCNN architecture, the inputs are sequential. If one training video is over, another video continues, with a special mark to indicate the first frame. Here, we call this input scheme as **sequential sampling**. Even this method works well on many sequential learning tasks [9], it may not be effective on video classification. Different from other sequential inputs, the consecutive image frames in a video are very alike. In the experiments (see Sec. 3), we find that the sequential sampling makes the training slow. This can also be seen as not considering the redundancy in videos.

To overcome this problem, we need **random sampling**. That is, make the consecutive inputs to the network not in the same video, see Tab. 1. One way to achieve this, is to cache the output of the last frame v^{t-1}. When the video is selected in training, we can compute the mapping by $\widehat{\Phi}_{W,\widehat{W},b}^{t}(x^t, v^{t-1})$, and update the parameters as well as v^{t-1}. The space complexity of this method is $O(MN)$, with M meaning the number of videos in the training dataset, and N meaning the space to cache v^{t-1}. For large dataset like UCF-101 [14], and deep models like GoogleNet [17], this method is impractical to deploy in a computer with a GPU card due to its limited memory.

The way we use here is to avoid caching v^{t-1}. We sample two consecutive image frames $\{x^{t-1}, x^t\}$ each iteration, then compute $\Phi_W(x^{t-1})$, make an approximation that v^{t-1} equals to the activation of all hidden layers from this forward, then compute $\widehat{\Phi}_{W,\widehat{W},b}^{t}(x^t, v^{t-1})$. Our method will cost about double space and double time each iteration, but can gain great speed in training compared to sequential sampling method used in the truncated BPTT algorithm.

By make this approximation, we can benefit from the random sampling. The drawback is that this approximation architecture ignores the relations from early frames (except the previous frames) and thus can not model long time relations in videos. We believe it's a worthy trade-off, since the temporal parameters can still represent the motion features. In the rest of the paper we denote the sequential sampling model as RCNN-s, and the random model as RCNN-r.

2.4. Testing

Giving a sequence of image frames, our model will give a prediction for each frame. In order to get video level prediction when testing, we gather all the predictions of image frames, return their average score, maximum score and log-average score (by first applying a log transformation on the scores and then averaging them). The reason we use log-average score is that the cross entropy loss is optimized in the log level of

Table 2. Basic results of split 1.

No.	Setting				Accuracy		
	Fps	Signal	Patches	Sampling	Mean fusion	Max fusion	Log-mean fusion
1	25	High layer	5	Random	79.5%	79.6%	79.7%
2	5	High layer	1	Random	79.4%	79.1%	79.6%
3	5	High layer	5	Random	79.8%	79.0%	**80.0%**
4	5	High layer	10	Random	79.7%	78.7%	79.6%
5	5	Medium layer	5	Random	78.1%	78.1%	78.0%
6	5	Low layer	5	Random	74.0%	73.6%	73.4%
7	5	Fusion of three	5	Random	78.8%	78.2%	78.6%
8	5	High layer	5	Sequential	71.6%	72.9%	73.4%

softmax predictions. The log-average score shows better performance than the other two, see Sec. 3.

3. EXPERIMENTS

We have done several experiments on the UCF-101 dataset [14] which contains 101 action classes with total 13320 videos. We follow the standard evaluation protocol, use the first split to compare models and report the average classification accuracy over three splits.

3.1. Architecture details

We revise the GoogleLeNet [17] architecture into GoogleLeNet-RCNN architecture for video classification problem by replacing the convolutional layers with RCNN layers and the fully connected layers with RNN layers. On top of GoogleLeNet, we put a softmax layer to get score for input frame each time step. The details of GoogleNet architecture is given in the supplement material.

We implement our architecture using the caffe toolbox [16]. Our models are optimized using Stochastic Gradient Descent (SGD) starting with a learning rate of 0.01. We set the momentum to be 0.9, weight decay 0.0002 and the batch size is 64 for most of time. The learning rate decreases exponentially through iterations.

The temporal parameters ($\widehat{W}$) are initialized using the xavier method [18] and the spatial parameters (W and b) are initialized to half of the parameters learned from a ImageNet model. We reduce the parameters of pre-training model to its half to keep the statics of inputs to each layer stable since each layer will sum the spatial part and temporal part from its inputs. Besides, we also use two stages training strategy. In the first stage, we freeze the spatial parameters W, and only update the temporal parameters. In the second stage, we finetune the whole model.

3.2. Data preparation

Most of the videos are of size 320×240, and we simply resize all the frames into 256×256. When training, We randomly pick a 224×224 patch from the resized image frames and then randomly flip them. This data augmentation method can help reduce over-fitting especially when the training data is not sufficient. When testing, we compare three sampling methods, 1) only crop the central area from image frames, 2) cropping the central and the four corner areas, 3) cropping above five patches and flip them. The score is obtained by averaging scores from multi patches. Tab. 2 shows the basic results of split 1. From row 2 to row 4, we can see that multi patches testing can gain better performance. But ten patches does not seem to work better than five patches.

We extract image frames with different frames per second (fps). Large fps will lead to small motion between image frames. We test two fps setting, 25 fps and 5 fps, see Tab. 2 row 1 and row 3. Our model perform better in 5 fps, suggesting that our model prefer bigger motion between image frames.

When training the GoogleNet-RCNN models, two additional signals are added to the network to gain fast learning, we denote the three signals as low signal, medium signal, high signal, and test their performance (Tab. 2 row 3, 5 and 6). The higher signal is better than medium signal and outperform the low signal by a large margin. This means that the good property of deep structure also benefit our RCNN layer. The best result is generated from the No.3 setting, which is 80.0% accuracy on split 1.

3.3. Sequential sampling vs. random sampling

Fig. 2 shows the training loss against training iteration curve, from which we can see that the random sampling is necessary for action classification, it will lead to faster training and better loss compared to sequential sampling. For sequential learning, the image frames in a video comes into the model sequentially. This will cause two problems when optimization, firstly, the input images are highly relevant, thus training will be slow. Secondly, the sequential inputs of image frames makes the model over-fitting time to time, when a new video comes, the loss will go up. The training curve of sequential learning is unstable even the loss are averaged across forty iterations (see Fig. 2).

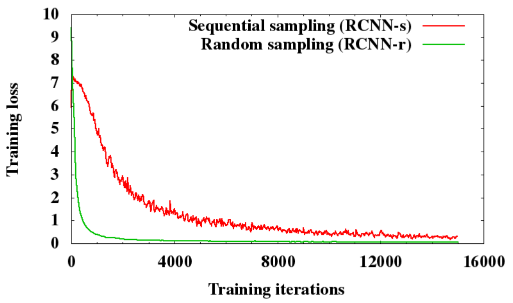


Fig. 2. Training loss-iteration curve of different sampling methods. (The loss is averaged over 40 iterations.)

Results of sequential and random sampling are shown on Tab. 2 (row 3 and row 8). The sequential sampling methods is not efficient, which is consistent with our analysis.

3.4. Overall performance

Following [12], we done experiments on dense optical flow. We computed the optical flow with off-the-shelf GPU implementation of [19] from the OpenCV toolbox. The architecture we trained optical flow stream is the same as the raw image stream. The only difference is that the input for optical flow is two channels. Thus we trained optical flow from scratch.

Tab. 3 shows the comparison of our methods and state-of-the-art methods. Our best result of RCNN-r model achieve 81.0%, which outperforms the single frame CNN model of [12] by a large margin, and only perform a little worse than [6] which pre-trains on a large dataset Sports-1M. We use the log-average score to fuse the two-stream outputs, and obtained 86.3% accuracy, which is competitive with the state of the art. Our model can process one frame in 12.5ms averagely in a machine with i7-4770 CPU, 32 GB memory and a Nvidia-k40c GPU card.

To fully understand our model, we visualize the feature maps from our learned RCNN model. In Fig. 3, we can see two flutes with different pixel intensities, this phenomenon

Table 3. Comparison with state-of-the-art methods.

Method	3-fold Accuracy		
	image	optical flow	two-stream
Improved dense trajectories (IDT) [3]	85.9%		
IDT with encodings [4]	87.9%		
Slow fusion CNN [7][1]	65.4%	–	–
Conv pooling CNN [6][1]	**82.6%**	–	88.2%
CNN + LSTM [6][1]	–	–	**88.6%**
CNN + SVM [12][2]	73.0%	83.7%	88.0%
CNN + LSTM [10]	71.1%	77.0%	82.9%
RCNN-s (ours)	74.1%	–	–
RCNN-r (ours)	**81.0%**	76.4%	86.3%

[1] Pre-training on sports-1M dataset.
[2] The training dataset was enlarged with HMDB dataset.

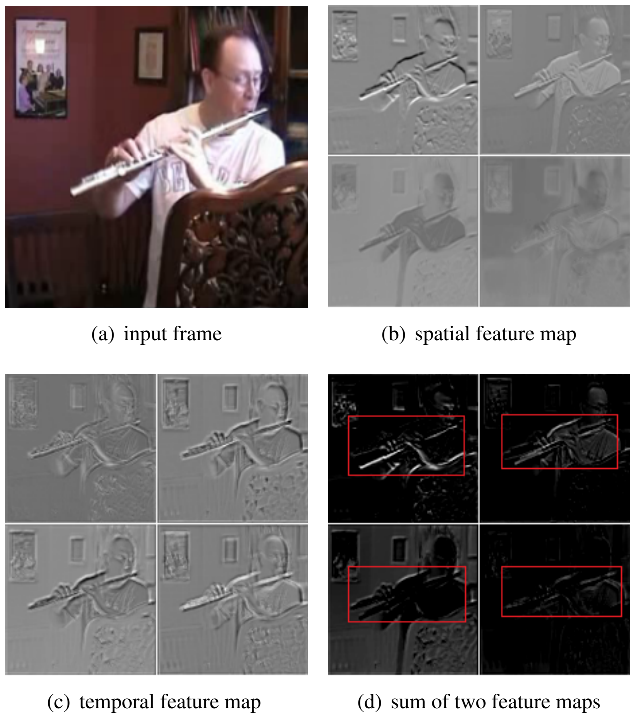


Fig. 3. Sample feature maps from the first convolution layer of our RCNN-r model. The moving parts (flute, in the red rectangle) are preserved in the sum of spatial and temporal features, which is analogous to the persistence of vision phenomenon. (Best seen in color.)

is analogous to the persistence of vision phenomenon [20] in the human visual system. This proves that our architecture can model motion well.

4. CONCLUSION AND FUTURE WORK

We proposed a new deep learning framework called RCNN by extending traditional RNN layer to RCNN layer. Our network can extract spatial feature by convolution operation and learning the temporal feature by recurrent linking of the neuron layers. Thus, it's appropriate for video classification tasks. Besides, we explore random sampling and sequential sampling when deal with videos, and prove that random sampling is necessary for video classification due to the large redundancy in videos. Our model can benefit from visualization of motion features, which is analogous to the persistence of vision phenomenon.

One drawback of our model is that it can not model long time relations from videos. We will refine our model by inducing LSTM units used commonly in the RNN architecture.

Acknowledgments

This work was partially sponsored by supported by the NSFC (National Natural Science Foundation of China) under Grant No. 61375031, No. 61573068, No. 61471048, and No.61273217, the Fundamental Research Funds for the Central Universities under Grant No. 2014ZD03-01, This work was also supported by the Beijing Higher Education Young Elite Teacher Program, Beijing Nova Program, CCF-Tencent Open Research Fund, and the Program for New Century Excellent Talents in University.

5. REFERENCES

[1] Navneet Dalal and Bill Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. IEEE, 2005, vol. 1, pp. 886–893.

[2] Navneet Dalal, Bill Triggs, and Cordelia Schmid, "Human detection using oriented histograms of flow and appearance," in *Computer Vision–ECCV 2006*, pp. 428–441. Springer, 2006.

[3] Heng Wang and Cordelia Schmid, "Lear-inria submission for the thumos workshop," in *ICCV Workshop on Action Recognition with a Large Number of Classes*, 2013.

[4] Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu Qiao, "Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice," *arXiv preprint arXiv:1405.4506*, 2014.

[5] Yann LeCun and Yoshua Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, 1995.

[6] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici, "Beyond short snippets: Deep networks for video classification," *arXiv preprint arXiv:1503.08909*, 2015.

[7] Andrej Karpathy, George Toderici, Sachin Shetty, Tommy Leung, Rahul Sukthankar, and Li Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. 2014, pp. 1725–1732, IEEE.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. 2009, pp. 248–255, IEEE.

[9] Alex Graves et al., *Supervised sequence labelling with recurrent neural networks*, vol. 385, Springer, 2012.

[10] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2625–2634.

[11] Sepp Hochreiter and Jrgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] Karen Simonyan and Andrew Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, 2014, pp. 568–576.

[13] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu, "3d convolutional neural networks for human action recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 1, pp. 221–231, 2013.

[14] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[15] Ronald J. Williams and Jing Peng, "An efficient gradient-based algorithm for on-line training of recurrent network trajectories," *Neural Computation*, vol. 2, no. 4, pp. 490–501, 1990.

[16] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B Girshick, Sergio Guadarrama, and Trevor Darrell, "Caffe: Convolutional architecture for fast feature embedding.," in *ACM Multimedia*, 2014, vol. 2, p. 4.

[17] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[18] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[19] Thomas Brox, Andrs Bruhn, Nils Papenberg, and Joachim Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Computer Vision-ECCV 2004*, pp. 25–36. Springer, 2004.

[20] M Coltheart, "The persistences of vision," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 290, no. 1038, pp. 57–69, 1980.