LEARNING DEEP TRAJECTORY DESCRIPTOR FOR ACTION RECOGNITION IN VIDEOS USING DEEP NEURAL NETWORKS

Yemin Shi, Wei Zeng, Tiejun Huang

School of Electronics Engineering
and Computer Science,
Peking University, Beijing 100871, China
{shiyemin; weizeng; tjhuang}@pku.edu.cn

*Yaowei Wang**

School of Information and Electronics,
Beijing Institute of Technology,
Beijing 100081, China
yaoweiwang@gmail.com

ABSTRACT

Human action recognition is widely recognized as a challenging task due to the difficulty of effectively characterizing human action in a complex scene. Recent studies have shown that the dense-trajectory-based methods can achieve state-of-the-art recognition results on some challenging datasets. However, in these methods, each dense trajectory is often represented as a vector of coordinates, consequently losing the structural relationship between different trajectories. To address the problem, this paper proposes a novel Deep Trajectory Descriptor (DTD) for action recognition. First, we extract dense trajectories from multiple consecutive frames and then project them onto a canvas. This will result in a "trajectory texture" image which can effectively characterize the relative motion in these frames. Based on these trajectory texture images, a deep neural network (DNN) is utilized to learn a more compact and powerful representation of dense trajectories. In the action recognition system, the DTD descriptor, together with other non-trajectory features such as HOG, HOF and MBH, can provide an effective way to characterize human action from various aspects. Experimental results show that our system can statistically outperform several state-of-the-art approaches, with an average accuracy of 95.6% on KTH and an accuracy of 92.14% on UCF50.

Index Terms— Deep Neural Network, Deep Trajectory Descriptor, action recognition

1. INTRODUCTION

Action recognition is one of active topics in computer vision and pattern recognition. Technologically, the task of action recognition is to identify the actions or behaviors of one or more persons from a series of observations in a video sequence. It can be applied to various domains such as public security, customer behavior analysis and in-home elder monitoring. However, accurately recognizing actions is still a challenging research task due to view changes, occlusion, variation in execution rate, and background clutter.

Motion representation plays an important role in the action recognition system. Recent studies show that, the Fisher vector representation [1] with the improved Dense Trajectory (IDT) features [2] is very effective for capturing motion information. Technologically, the original dense trajectory feature is to sample and track dense points from each frame in multiple scales, while its improved version also considers the camera motion estimation and replaces the bag-of-features with Fisher vector. As such, it can obtain the state-of-the-art performance on several action recognition datasets [2]. However, in both two versions, the dense trajectories are only used to get the locations of feature points, while their characteristics on motion representation have not been fully taken advantage of. Moreover, each dense trajectory is often represented as a vector of coordinates, consequently losing the structural relationship between different trajectories.

To partially address this problem, Ali *et al.* [3] represented the normalized trajectories of human joints as a sequence of locations which were then used to extract invariant features of the reconstructed phase space. However, their approach is heavily dependent on the quality of trajectories, thus easy to be affected by background clutter. In this study, we focus on developing a more effective way to encode the trajectories and then make full use of them. Our study is motivated by the recent great success of deep neural networks (DNN). Many works have shown that DNNs can achieve the state-of-the-art performance in a lot of pattern recognition tasks. More recently, they have also been introduced in action recognition so as to effectively capture the motion information. Ji *et al.* [4] proposed a novel DNN, i.e., 3D convolution neural network (CNN) model. This 3D-CNN gets input from multiple channels and performs 3D convolutions in each channel so as to extract features from both the spatial and the temporal dimensions, thereby effectively capturing the motion information encoded in multiple adjacent frames. By regularizing the output with high-level features and combining the predictions of a variety of different models, it can achieve a superior perfor-

Yaowei Wang is the corresponding auther.

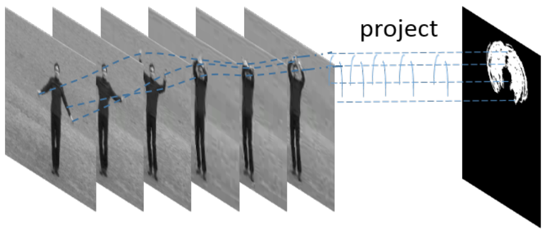

Fig. 1. Trajectories from multiple consecutive frames are projected onto a canvas so as to effectively characterize the relative motion in these frames using a 2D image.

mance in the real-world environment of airport surveillance videos. Similar to CNN, V. Le *et al.* [5] also proposed an independent subspace analysis (ISA) network. They first learn features with small input patches; the learned features are then convolved with a larger region of the input data. By stacking this convolution layer, the ISA network is able to learn a hierarchical representation of the data suitable for recognition [6].

Following these ideas, this paper proposes a novel Deep Trajectory Descriptor (DTD) to effectively characterize motion in video, consequently facilitating action recognition in complex scenes. Basically, our DTD can be viewed as a more compact and powerful representation of dense trajectories. First, we extract dense trajectories from multiple consecutive frames and then project them onto a canvas. This will result in a "trajectory texture" image which can effectively characterize the relative motion in these frames (as illustrated by Fig. 1). In this way, we transfer the raw 3D (two for spatial and one for temporal) space into a 2D space, hence significantly reducing complexity. Based on these trajectory texture images, a deep neural network (DNN) is utilized to learn a more macroscopical representation of dense trajectories. In the action recognition system, the DTD descriptor, together with other non-trajectory features such as HOG (histogram of gradient), HOF (histogram of optical flow) and MBH (motion boundary histogram), can provide an effective way to characterize human action from various aspects. Experimental results show that our system can statistically outperform several state-of-the-art approaches, with an average accuracy of 95.6% on KTH and an accuracy of 92.14% on UCF50.

The rest of the paper is organized as follows: In section 2, we briefly introduce the concept of dense trajectories. The proposed DTD is presented in section 3. Our action recognition system with DTD is described in section 4. Experimental results are discussed in section 5. Finally, section 6 concludes this paper.

2. DENSE TRAJECTORIES

In this section, we will describe the dense trajectory feature (DTF) [7] and its improved version (i.e., IDT) [2], which are used as the baseline in our system. Basically, dense trajectory is used to describe the motion between frames. Feature points are sampled on a grid with the space of W pixels and then tracked in multiple spatial scales. Point p_t at frame t is represented as (x_t, y_t). Then we track p_t by median filtering in a dense optical flow field $\omega = (u_t, v_t)$, as follows:

$$P_{t+1} = (x_{t+1}, y_{t+1}) = (x_t, y_t) + (M * \omega)|_{(\overline{x}_t, \overline{y}_t)}, \qquad (1)$$

where M is the median filtering kernel, and $(\overline{x}_t, \overline{y}_t)$ is the rounded position of (x_t, y_t). The length of trajectories is limit to L frames to avoid the drifting problem. Note that the too small or too large trajectories will be removed. Then a trajectory is represented as $D = (\Delta P_t, \Delta P_{t+1}, \Delta P_{t+2}, ..., \Delta P_{t+L-1})$, where ΔP_t is the displacement between P_t and P_{t+1}. The resulting vector is normalized by the sum of the magnitudes of the displacement vectors.

Beside the trajectory descriptors, some other features are also computed so as to capture texture and local motion information. In [7], Wang *et al.* compute HOG, HOF and MBH features along the dense trajectories. Different features are encoded with a standard bag-of-features approach and then combined in a multi-channel framework with a non-linear SVM and a χ^2-kernel [8]. A one-against-rest approach is used to handle the multi-class classification problem. In the improved Dense Trajectory (IDT) [2], the descriptor dimensionality is reduced by Principal Component Analysis (PCA), and the bag-of-features approach is replaced by Fisher vector. After applying power and L2 normalization to the Fisher vector, different features are concatenated and then a linear SVM is used for classification.

3. DEEP TRAJECTORY DESCRIPTOR

Fig. 2 illustrates the framework of our Deep Trajectory Descriptor (DTD). After extracting dense trajectories from a video sequence, a denoising step is performed so as to reduce the influence of clutter background. Instead of the camera motion estimation method in [2], a background subtraction method is used in our approach for trajectories denoising. The trajectories are then projected onto a canvas to obtain trajectory texture images. On these trajectory texture images, the DNN model is trained to generate the DTD descriptors.

3.1. Background-subtraction-based IDT (bIDT)

In [2], a camera motion estimation method is used to remove the trajectories generated by camera motion. Namely, they use SURF features to estimate camera motion and remove the trajectories which have a smaller displacement compared with the camera motion; then a state-of-the-art human detector is used to solve the problem of human dominating the frame. However, this method may be complicated and low efficient sometimes.

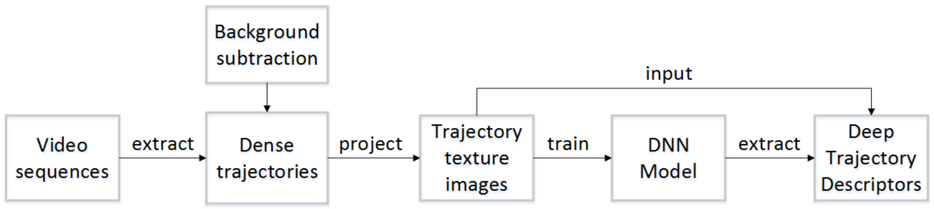

Fig. 2. Illustrating the pipeline of our Deep Trajectory Descriptor (DTD). We compute dense trajectories first. In order to capture the structure information, the trajectories are projected onto a canvas to obtain trajectory texture images. The DNN model is then trained to extract the DTD features from trajectory texture images.

In our approach, we improve the dense trajectories by using ViBe [9]. ViBe is a background subtraction technique that incorporates several innovative mechanisms. First, we apply background subtraction to the raw video clips to get foreground clips. When extracting trajectories from raw video clips, every feature point is tested in a $r * r$ square area in the corresponding foreground frame f_t:

$$S^{fore} = \sum_{i=-\frac{r}{2}}^{\frac{r}{2}} \sum_{j=-\frac{r}{2}}^{\frac{r}{2}} f_t(\overline{x}_t + i, \overline{y}_t + j), \tag{2}$$

where S^{fore} is the sum of the foreground square area, (i, j) index around the square area. If S^{fore} is less than F where F is the threshold, this feature point is regarded as coming from camera motion and will be removed.

3.2. Trajectory Texture Image

Traditionally, most approaches treat videos in the three-dimensional space and try to extract features from both temporal and spatial dimension [4]. This leads to very high computational overhead when a large volume of video data are processed. To address this problem, we propose a novel way to convert a video into a two-dimensional space so that it can be easily handled. Imagine there is a fixed invisible canvas behind the video frame, and every motion in the video will be projected onto this canvas. Then after several frames, we can get a new image of motion trajectories, which records the information of all the motions in the past frames. Note that Fig. 1 has demonstrated this idea in an intuitive way.

To implement this idea, we first extract dense trajectories to obtain coordinates and optical flow along every trajectory (denoted as $(\overline{x}_t, \overline{y}_t, u_{\overline{x}_t,\overline{y}_t,t}, v_{\overline{x}_t,\overline{y}_t,t})$, where t is the frame number, $\overline{x}_t$ is the rounded value of x at frame t and $(u_{\overline{x}_t,\overline{y}_t,t}, v_{\overline{x}_t,\overline{y}_t,t})$ is the value of the dense optical flow on vertical and horizontal directions separately). We then simply draw trajectories from multiple frames onto a canvas C so as to assemble them together.

$$C_{\overline{x}_t,\overline{y}_t,s} = \sqrt{u^2_{\overline{x}_t,\overline{y}_t,t} + v^2_{\overline{x}_t,\overline{y}_t,t}}, \tag{3}$$

where s is the canvas index. In case of too many trajectories, we use S canvases and divide the trajectories from a video clip into S segments. After the projecting, every clip will correspond to several canvases.

3.3. Deep Trajectory Descriptor (DTD)

In order to learn a more compact and powerful representation of trajectories, a deep neural network (DNN) is used to process the trajectory texture images. All trajectory texture images for a video clip are stacked together and used as one multi-channel sample for DNN. Our DNN model consists of four convolution layers, three max-pooling layers, two local response normalization layers and one full connection layer.

Basically, the convolutional layers perform the convolution operation to the previous layer and extract features from local neighborhood. Then an additive bias is applied and the result is passed through an activation function. Formally, the value of an unit at position (x, y) of the j^{th} feature map in the i^{th} layer, denoted as v^{xy}_{ij}, is given by

$$v^{xy}_{ij} = a\Big(b_{ij} + \sum_{m=0}^{M-1} \sum_{w=-\Delta}^{\Delta} \sum_{h=-\Theta}^{\Theta} \omega^{wh}_{ijm} v^{(x+w)(y+h)}_{(i-1)m}\Big), \tag{4}$$

where $\Delta = \frac{W_i}{2}$, $\Theta = \frac{H_i}{2}$, $a(\cdot)$ is the activation function, b_{ij} is the bias for this feature map, M is the number of feature maps in the $(i-1)^{th}$ layer, ω^{wh}_{ijm} is the weight of the kernel at position (w, h), and W_i and H_i are the width and height of the kernel, respectively. We use the ReLU [10] activation function in all convolution layers. In terms of training time with gradient descent, DNNs with ReLUs train several times faster than their equivalents with $tanh$.

The pooling layers pool over local neighborhood on the feature maps in the previous layer. Thus, the resolution is reduced, thereby enhancing the invariance to distortions on the input.

The LRN layer implements a form of lateral inhibition inspired by the type found in real neurons. The response-normalized activity at position (x, y) is given by the expression

$$r^{xy}_{ij} = v^{xy}_{ij} \Big/ \Big(k + \alpha \sum_{j=max(0,j-n/2)}^{min(M-1,j+n/2)} (v^{xy}_{ij})^2\Big)^{\beta}, \tag{5}$$

where k, n, α, and β are hyper-parameters.

The DNNs are trained on the training set first. After the training phase, we re-run the DNN models on the whole dataset (both training and test sets) and ignore the back propagation computation. We save all the values at the fourth convolutional layer at every iteration as the final representation of the video clips.

4. ACTION RECOGNITION WITH DTD

In this section, we first describe our system framework for action recognition, and then discuss some details about the system implementation.

4.1. The System Framework

The framework is shown in Fig. 3. Roughly speaking, this system consists of three modules, i.e., dense trajectory extraction, DTD generation, and classification. It should be noted that, to further boost the recognition performance, we also introduce some other non-trajectory features so as to capture texture and local motion information, just as [7] did. For each trajectory, we compute HOG, HOF and MBH descriptors and these descriptors are then encoded with Fisher vector. The encoded features are simply concatenated with DTD and a linear SVM is used for classification.

4.2. Implementation Issues

We use the ViBe's implementation provided by Barnich[1] and our dense trajectory is modified based on Wang's implementation [2].

To achieve the speedup and enable our model to be trained and applied to large datasets, we adopt caffe [11], a popular deep learning framework developed with cleanliness, readability, and speed in mind.

We use Principal Component Analysis (PCA) to reduce the dimensionality of the descriptors (DNN feature excluded) by a factor of two. We randomly sample 20 percent of the features to estimate $K = 256$ GMM for each kind of features. We then apply L2 normalization to every Fisher vector and simply concatenate their normalized Fisher vectors to combine different descriptors. The encoded feature is 107,520 dimensions: 6,144 for DTD features, 24,576 for HOG, 27,648 for HOF, and 49,152 for MBH. For classification, we fix $C = 100$ for the SVM in our implementation and use the one-against-rest approach for multi-class classification. Here LIBSVM[12] is employed to do the classification job.

[1] `http://www.motiondetection.org/`

[2] `http://lear.inrialpes.fr/~wang/improved_trajectories`

5. EXPERIMENTS

In this section, we first briefly introduce our experimental settings and the datasets used in our evaluation in section 5.1. Section 5.2 reports the gain due to our background-subtraction-based IDT, and the performance of our system by comparing with several state-of-the-art methods.

5.1. Experimental Settings

In our experiments, we use a fixed configuration for DNN. The size of trajectory texture images is kept at $165 * 165$. We set the number of convolution kernels in the four convolution layers to 96, 256, 384 and 384 and set the kernel sizes to $6*6$, $6*6$, $5*5$ and $4*4$ individually. The three pooling layers have the kernel sizes of $2*2$, $3*3$ and $3*3$. We set $k = 2$, $n = 3$, $\alpha = 5 \times 10^{-5}$, and $\beta = 0.75$ for LRN layers. All our experiments were performed with a single NVIDIA K40 GPU.

Two standard action datasets, KTH and UCF50, were used in our experiments:

The **KTH** action dataset [13] contains six types of human actions (see Fig.4) and 25 subjects. The sequences are captured in four different scenarios with a homogeneous background. The dataset consists of 2,391 sequences. Following the original experimental setup, we divide the samples into the test set (9 subjects: 2,3,5,6,7,8,9,10, and 22) and the training set (the remaining 16 subjects). We will train and evaluate a multi-class classifier and report the average accuracy over all classes as the performance measure.

The **UCF50** [14] dataset is a more challenging dataset with 50 action categories. Each category contains 25 groups and each group includes at least 4 action clips. There are 6,618 video clips in total. The clips in the same group may share some common features. Thereby the dataset should be split in a group level. In order to do quick test, we random select one group as the test set and train on the others. This can be extended to leave-one-group-out cross-validation easily.

5.2. Evaluations

5.2.1. Gain from Background-subtraction-based IDT

In our experiments, we use default parameters for ViBe. We compare our background-subtraction-based method with the camera motion estimation method in the original IDT [2]. Our background-subtraction-based method always performs a foreground test during computing feature points, and removes trajectories in the background. As shown in Table.1, our background-subtraction-based method significantly reduces the space cost and keeps the performance of dense trajectories in the same time. On average, our improvement saves 59.14% and 21.92% disk usage on KTH and UCF50 datasets respectively.

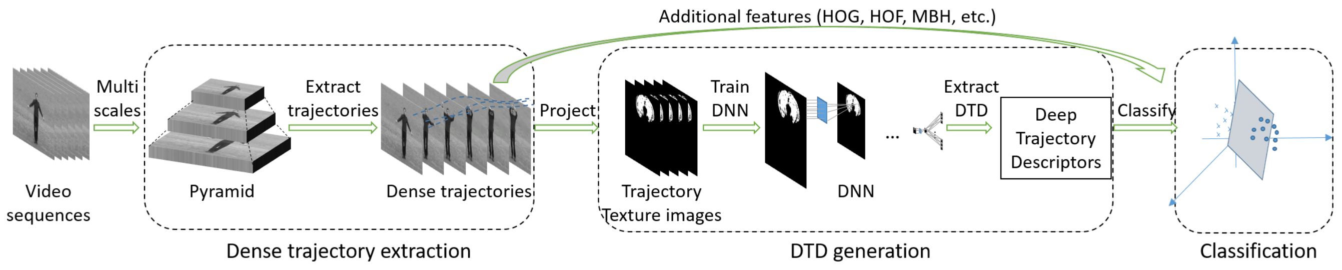

Fig. 3. The framework of our action recognition system with DTD.

Fig. 4. Sample frames from video sequences of KTH (first row) and UCF50 (second row) action datasets.

Table 1. Comparison of disk usage between our bIDT with the original IDT [2]. We remarkably reduce the space cost, meanwhile remaining the performance.

Dataset	IDT	bIDT	Saving	Best Acc
KTH	17.4G	7.11G	59.15%	95.60%
UCF50	146G	114G	21.92%	92.14%

However, our background-subtraction-based method is kind of rude and possibly removes trajectories of slight movements or tiny objects. Thus, it may lead IDT to a worse result due to the partly loss of statistical information. On the other hand, the loss is too tiny to change the overall structure. Owing to the fact that our DTD mainly takes advantage of structural information and DNN is also robust to distortion, the loss is tolerable for DTD. As shown in Table 4, although bIDT produces a worse result compared with DTF in [7], we get a better result after combining the bIDT with DNN.

5.2.2. *Evaluation of DTD*

We compare our DTD with DTF [7], IDT [2], and bIDT, in terms of the action recognition performance. We use the same parameters as discussed previously. To compute the trajectories, we set $N = 32, n_\delta = 2, n_\gamma = 3$ for dense trajectories and $r = 3, F = 256$ for background subtraction. We fix the trajectory length to $L = 15$ and the step size to $W = 5$ as in [7].

Table 2. Performance confusion matrix for our DTD on the KTH dataset. The overall accuracy rate is 95.60%.

	bx	cl	wv	jg	rn	wk
box	1	0	0	0	0	0
clap	0	1	0	0	0	0
wave	0	0.125	0.875	0	0	0
jog	0	0	0	0.980	0.020	0
run	0	0	0	0.118	0.882	0
walk	0	0	0	0	0	1

The KTH action dataset is the most common dataset used in evaluations of action recognition. The confusion matrix of KTH for our approach is shown in Table 2. Our approach successfully recognizes boxing, hand clapping, walking with the accuracy of 100%, and recognizes jogging with the accuracy of 98.0%. We also compare our results with the state-of-the-art results in Table 3. The average accuracy of our DTD is 95.60%, which is comparable to the state-of-the-art result, i.e., 95.70%.

The UCF50 dataset is an extension of the YouTube dataset [15]. Since we use a different training/testing split setup, we only compare our DTD with bIDT and IDT's results in [2]. On this dataset, our DTD method obtains better results when compared with bIDT. As Wang *et al.* [2] did, we also apply the power normalization and get the accuracy of 92.14%, which is much better than bIDT about 3.5% and IDT [2] about 1% .

Table 3. Comparison of our DTD with the state-of-the-art methods on the KTH dataset in the literature.

Approach	Accuracy
Laptev et al. [15]	91.80%
Kovashka et al. [16]	94.53%
Gilbert et al. [17]	**95.70**%
Le et al. [5]	93.90%
Wang et al. [7]	94.20%
Ours	**95.60**%

Table 4. Comparison between our DTD and IDT on the KTH and UCF50 datasets.

(a) Without power normalization

	KTH	UCF50
Wang's DTF [7]	94.2%	N/A
bIDT	93.63%	87.86%
DTD	**95.60%**	**90.00%**

(b) With power normalization

	KTH	UCF50
Wang's IDT [2]	N/A	91.2%
bIDT	**95.37%**	88.57%
DTD	95.24%	**92.14%**

Table 4 compares our results with bIDT on the KTH and UCF50 datasets. On both datasets, our DTD outperforms bIDT by 2% and outperforms DTF [7] and IDT [2] by 1%. Power normalization is designed for Fisher vector to increase the magnitude of each value. We test our results after applying power normalization to the encoded Fisher vector features. When only L2 normalization is applied, DTD performs better than bIDT by 2% in both KTH and UCF50 datasets. However, bIDT achieves a good result of 95.37% on the K-TH dataset with the power normalization approach. On the contrary, DTD is not able to improve the result by using the power normalization. On the UCF50 dataset, bIDT achieves the accuracy of 88.57% when using the power normalization, which is slightly better than the case when only using the L2 normalization. Similarly, the performance of our DTD is also improved from the accuracy of 90.00% to 92.14% by using the power normalization.

6. CONCLUSIONS

This paper proposes a novel Deep Trajectory Descriptor (DT-D) as a more compact and powerful representation of dense trajectories. We show that the disk usage of IDT can be significantly reduced by using the background-subtraction-based method. Experimental results show that our DTD can statistically outperform several state-of-the-art approaches, with an average accuracy of 95.6% on KTH and an accuracy of 92.14% on UCF50.

Acknowledgement. This work is partially supported by grants from the National Basic Research Program of China under grant 2015CB351806, and the National Natural Science Foundation of China under contract No. 61390515, and No. 61471042. The Tesla K40 used for this research was donated by the NVIDIA Corporation.

7. REFERENCES

[1] Florent Perronnin, Jorge Sánchez, and Thomas Mensink, "Improving the fisher kernel for large-scale image classification," in *ECCV*, pp. 143–156. Springer, 2010.

[2] Heng Wang and Cordelia Schmid, "Action recognition with improved trajectories," in *ICCV*. IEEE, 2013, pp. 3551–3558.

[3] Saad Ali, Arslan Basharat, and Mubarak Shah, "Chaotic invariants for human action recognition," in *ICCV*. IEEE, 2007, pp. 1–8.

[4] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu, "3d convolutional neural networks for human action recognition," *PAMI*, vol. 35, no. 1, pp. 221–231, 2013.

[5] Quoc V Le, Will Y Zou, Serena Y Yeung, and Andrew Y Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *CVPR*. IEEE, 2011, pp. 3361–3368.

[6] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *ICML*. ACM, 2009, pp. 609–616.

[7] Heng Wang, Alexander Klaser, Cordelia Schmid, and Cheng-Lin Liu, "Action recognition by dense trajectories," in *CVPR*. IEEE, 2011, pp. 3169–3176.

[8] Muhammad Muneeb Ullah, Sobhan Naderi Parizi, and Ivan Laptev, "Improving bag-of-features action recognition with non-local cues.," in *BMVC*, 2010.

[9] Olivier Barnich and Marc Van Droogenbroeck, "Vibe: A universal background subtraction algorithm for video sequences," *TIP*, vol. 20, no. 6, pp. 1709–1724, 2011.

[10] Vinod Nair and Geoffrey E Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010, pp. 807–814.

[11] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[12] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: A library for support vector machines," *ACM TIST*, vol. 2, pp. 27:1–27:27, 2011, Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[13] Christian Schuldt, Ivan Laptev, and Barbara Caputo, "Recognizing human actions: a local svm approach," in *ICPR*. IEEE, 2004, vol. 3, pp. 32–36.

[14] Kishore K Reddy and Mubarak Shah, "Recognizing 50 human action categories of web videos," *MVA*, vol. 24, no. 5, pp. 971–981, 2013.

[15] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld, "Learning realistic human actions from movies," in *CVPR*. IEEE, 2008, pp. 1–8.

[16] Adriana Kovashka and Kristen Grauman, "Learning a hierarchy of discriminative space-time neighborhood features for human action recognition," in *CVPR*. IEEE, 2010, pp. 2046–2053.

[17] Andrew Gilbert, John Illingworth, and Richard Bowden, "Action recognition using mined hierarchical compound features," *PAMI*, vol. 33, no. 5, pp. 883–897, 2011.