HHS Public Access

Author manuscript

Neurocomputing. Author manuscript; available in PMC 2017 December 29.

Published in final edited form as:

Neurocomputing. 2016 February 20; 178: 87–102. doi:10.1016/j.neucom.2015.09.112.

ARCH: Adaptive recurrent-convolutional hybrid networks for long-term action recognition

Miao Xin[a], **Hong Zhang**[a], **Helong Wang**[b], **Mingui Sun**[c], and **Ding Yuan**[a,*]

[a]Image Processing Center, Beihang University, Beijing, China

[b]Luoyang Electro-Optical Equipment Research Institute, Luoyang, China

[c]Laboratory for Computational Neuroscience, University of Pittsburgh, PA, USA

Abstract

Recognition of human actions from digital video is a challenging task due to complex interfering factors in uncontrolled realistic environments. In this paper, we propose a learning framework using static, dynamic and sequential mixed features to solve three fundamental problems: spatial domain variation, temporal domain polytrope, and intra- and inter-class diversities. Utilizing a cognitive-based data reduction method and a hybrid "network upon networks" architecture, we extract human action representations which are robust against spatial and temporal interferences and adaptive to variations in both action speed and duration. We evaluated our method on the UCF101 and other three challenging datasets. Our results demonstrated a superior performance of the proposed algorithm in human action recognition.

Keywords

Action recognition; Deep learning; Hybrid feature learning

1. Introduction

Action recognition is an important field in computer vision. In recent years, this field attracted significant attention for its great value in both research and application [1,2]. However, owing to the huge amount of computation required, fickle interference factors and high subjectivity of human actions, recognizing human actions is a challenging task, especially in real-world environments.

To achieve a high recognition accuracy, the recognition system needs to take full advantage of various useful clues to understand video contents, including not only the static elements (e.g., senses categories, observed objects and environments), but also dynamic information such as the motion's trajectory, transition and saltation. However, understanding video contents has been extremely difficult in real-world scenarios. The following problems have been long-standing:

*Corresponding author. dyuan@buaa.edu.cn (D. Yuan).

1. *Spatial domain variations*: Spatial domain variations mainly refer to the fact that images' appearances are diverse. Videos may suffer from noise. The action entities may have different shapes and scales, and the differences in viewing location may lead to occlusion.
2. *Temporal domain polytropes*: Temporal domain polytropes refer to the phenomenon that actions may be distorted on the timeline. Usually, the same actions in two cases may also have different velocities and durations. We call it the actions' time domain multiscale property.
3. *Inter-class and intra-class correlations*: Naturally, different classes of actions may share some similar local motions, while two instances that belong to the same category may yet have quite discrepant motions in some local time. To achieve a satisfactory action recognition, the interclass differences must be enlarged while the intraclass differences must be reduced.

Generally, most of existing approaches [3–8] address a single aspect of the action recognition problem. For example, [7,8] target on camera motion problem, which is one factor of spatial domain variations. Refs. [3,4] redefine the notion of "pose", and propose a two-layer classification/regression model for detecting people and localizing body components. Refs. [5,6] employ time independent representations to deal with the problem of time-scale variations. Ref. [9] uses dynamic time warping (DTW) to regulate intra- and inter-person variabilities. Many of these approaches produce good results in their target problems, however, in real-world videos, these issues are usually entangled with each other, making these approaches ineffective. Moreover, most of existing methods used some simple assumptions for model construction, which may not be satisfied in real-life environments.

Although these problems represent enormous challenges to the existing feature designing methods, action recognition has numerous practical applications and these problems need to be solved. An important question is asked how we jointly address various typical problems, rather than just one at a time.

In this paper, we consider all three problems simultaneously, and jointly search and optimize our solution by hybrid model designing. Inspired by the data-driven feature learning methodology, we approach the solution along two paralleled lines. (1) *Data*: Inspired by methodologies in cognitive neuroscience, we propose a data preparation method specially designed for long-term action recognition. (2) *Feature extraction*: We propose a hybrid deep neural network to hierarchically extract features in space, local time and global sequence and then fuses them to obtain integrated and highly abstract representation of human actions.

We attempt to provide an architectural framework for the action recognition problem. Intuitively, this framework could be described as a "network upon networks" architecture, which is composed of a data module and a learning module. The learning module again includes local motion learning and global sequence learning submodules. Our proposed learning module is called Adaptive Recurrent-Convolutional Hybrid (ARCH) Networks.

Contributions of this work are two folds: (1) a problem-oriented long-term key data selection method based on cognitive theory, and (2) a time–space–sequence combined feature learning method implemented by a hybrid deep neural network.

Our feature extraction and action recognition method is evaluated using a common dataset UCF101 [10]. We also examine the generalization of feature extraction using other datasets by transfer learning. Experiments demonstrated that our method outperforms state-of-the-art deep-learning-based methods [11–14].

The rest of this paper is organized as follows. In Section 2, we provide a short review on the research status. Our model design is described in Section 3. The details about the implementation and training methods are given in Section 4. We demonstrate our experimental results in Section 5, and conclude our work in Section 6.

2. Related works

Action recognition has been studied by computer vision community for decades. Early studies [15,1] aimed at simple actions such as posture changes, body tracking, and simple motions. With rapid developments in *feature engineering* techniques, studies on human actions are gradually advanced towards practical applications. In various methods reported, features (and the accesses to good features) are always the core of this research field.

Hand-crafted feature engineering: Hand-crafted designing has long been a feasible way to acquire effective features. A variety of approaches utilizes feature engineering for both low-level and high-level action recognition tasks. Low-level action recognition methods mainly focus on recognizing atomic actions of a person. One class of method is based on 3d space–time volume, e.g. “space–time shapes” [16] and motion history volumes (MHV) [17]. These approaches are good at recognizing transient and periodically repeated actions. As an improvement, trajectories-based methods concentrate on some relatively spare key-points or curves [7,8]. They improved the robustness to viewpoint variation. Another representative direction is lots of feature-based approaches, e.g., local interest points features, appearance-based local features, poselets methods [3,4]. Comparatively, these methods are more reliable in the presence of multiple interferences.

High-level actions are usually composed of a set of simple actions. In modeling sequence dynamics, there are classical methods such as state-based methods using Markov models [18,19] and graph-based methods [20]. In recognizing multiple people or group activities, there are description-based [21] and Bayesian-based [22,19] approaches.

Deep-learning-based methods: In recent years, the deep-learning technique was developed which attracted great attention in the computer vision field due to its high performance [23–28]. Comparing with the traditional methods, deep learning automatically selects features, which represents a powerful advantage over other methods. However, for video based action recognition, the current deep-learning-based methods [14,29,12] achieved only comparable or slightly better performance than the best hand-crafted methods. One reason is that the deep model requires enormous data for training. However, high-quality and large sets of data

are hard to collect. Besides, video data is far larger than image data, which makes the training time unrealistically long even powerful computers are used. Nevertheless, explorations on deep learning have exhibited promising results and this technique holds promise for a significant advancement in the action recognition field.

Convolutional Neural Networks (CNNs), as a specialized kind of neural network, demonstrated powerful abilities with the aid of high-performance computing platforms. Extending the 2D CNN, [30] uses 3D convolutional filters to address video processing problem. Ref. [11] uses a temporal pooling technique to address temporal fusion problem. Considering both the still and dynamic information, [14] presents a special “two-stream” network with an impressive performance.

Considering the problem of lacking high-quality labeled data, some studies attempt to extracts features in an unsupervised fashion. Generally, most of these works follow the *encoder–decoder* style. Ref. [31] uses the Restricted Boltzmann Machines (RBM) to obtain informative but short feature vectors, while [29] uses a Long Short-term Memory (LSTM) encoder–decoder framework to learn video representations.

Since it is hard to use a single method to accomplish complex tasks, some hybrid methods arise. Classical methods such as Dynamic Time Warping (DTW) [9] and Hidden Markov Model (HMM) [32] have been used to model sequence dynamics. Generally, both DTW and HMM factor certain hypothesis into local (instantaneous) metrics and transition metrics [32]. Ref. [33] proposed a hybrid architecture for human pose estimation, which consists of a deep convolutional network and a Markov Random Field (MRF). However, although superior in the interpretive perspective, these methods have a drawback in lacking representability for complex non-linear problems.

Comparably, the Recurrent Neural Network (RNN) [34,35] is a powerful tool for time–sequence modeling. As an abstract dynamical system, it can well-represent high-dimensional hidden states and non-linear dynamics [36,34] which are highly desirable in action recognition. Therefore, a natural idea is to jointly use CNN and RNN for spatial and temporal feature extraction. Ref. [37] reports an initial work on combining 3D CNNs and LSTM. Ref. [29] develops a recurrent convolutional architecture for large-scale visual learning.

Recent trends on action recognition can be summarized in two major aspects: (1) large-scale action classification in real-world applications; (2) recognition of complex and high-level sematic activity, e.g., long-term tasks, interactions, event-oriented, and group activities. We believe that more advanced methods will be constantly improved. However, a noteworthy problem is how to well integrate these methods in a high-efficiency framework.

3. ARCH networks for action recognition

In this section, we propose a static-dynamic-sequential synthesis feature-learning method, aiming at extracting a combined action representation in the temporal, spatial and sequential domains.

Usually, a complex human action consists of a set of sub-actions which are short and discriminative motions. Each sub-action is characterized with features in time and space. These features distribute in different stages of action sequences, providing important cues for recognizing the action. In the rest of this section, we first give an overview of our ARCH networks. Then, we will describe the details of this model individually.

3.1. Architecture overview

Overall, an ARCH network is composed of two modules, a data module and a feature learning module.

Data module: The data module performs data-preparation for the high-level feature extraction system. The main functions of this module are compressing data and solving the time warping problem. It selects key-frames, generates short-memory sequences (SMSs), and performs hybrid data packaging. Its output is fed to the feature-learning module. Details of the data module are described in Section 3.2.

Feature learning module: The feature learning module is composed of two parts (Fig. 1) where the lower part is a group of specially designed CNNs for extracting key motion patterns. Different from the standard CNN, we design a Temporal–Spatial-Fusion Convolutional Neural Network (TSF-CNN), which adopts a double-channel architecture specifically used for static and dynamic feature exaction. The details of TSF-CNNs are provided in Section 3.3. The upper part of the ARCH network is a recurrent network for modeling the distribution patterns of typical motions during a human action. The output of the recurrent network is integrated ARCH features with respect to the input video. This part will be described in Section 3.4.

Our goals of the ARCH network design are described below.

1. The multi-layers fusion feature learned from TSF-CNN should be robust against some typical interferences. This robustness is achieved by a unique nosing-denosing training (see Section 5.2.1). This design addresses the first problem stated in the beginning of the Introduction section.
2. Recognizing sparse key-motions (in SMSs) is an effective method to overcome the problem of speed variation in an action. Thus, it can be used to solve the time domain multi-scale problem (the second problem stated previously). In addition, it can remarkably reduce the amount of data to be processed, greatly reducing the computational load, which is important in long-term action recognition.
3. Although different classes of action may share common motions at a certain time, their orders are unlikely to be the same. The recurrent neural network can well model the nonlinear correlation, i.e., the inter-class and intra-class variability. This addresses the third problem.

3.2. Data preparation: unsupervised key-motions segmentation

The purpose of data preparation is two-fold. First, it reduces the amount of data to be processed. Since videos are rapidly generated and much larger than static images, learning

directly from raw data frame-by-frame are computational expensive and redundant. Essentially, not all frames are significantly discriminative, and some outliers may even play negative roles in action recognition. Hence, utilizing all data (every frame) is not productive, and bypassing unimportant frames is a better approach.

Second, data preparation provides a solution to the time-domain multiscale problem. Different categories of actions always have different and varying durations. We use the key-frame concept to partition the action into different sub-actions and normalize them. Our concept is implemented by key-frames selection and short-term memory sequence generation.

3.2.1. Sparse key-frames selection—*Key-frames* usually refer to frames at start and end points of a smooth motion [1]. Usually, smooth and radical motions are continuously distributed in an action sequence. In neuroscience [38,39], it has been demonstrated that the human visual perception system is quite sensitive to saltation which refers to a sudden and significant change in direction, speed or frequency of the motion. This discovery implies that motion transitions play an important part in cognition and recognition. Based on this knowledge, we employ some low-level visual processing techniques to segment a motion into relatively smooth sub-sequences.

Naturally, this problem could be converted to the sequence partition problem, which can be solved by unsupervised clustering methods. However, we cannot assign the number of motion cluster centers easily because we do not have the appropriate number of key-frames. We approach this problem using a time-constrained clustering method based on the *FastSearch* [40] algorithm to group frames into several fragments. The border time-point of each fragment is the key-frame that we want.

Given a sequence S, let the frame at the moment t be f^t. We treat the frame order as an extra feature dimension in addition to frame data. Formally, the input to the timing-constrained cluster is a vector set V, which is composed of v^j, $V = v^j | j \in (1, t)$ where v^j is also a compound vector, given by

$$v_i^j = \begin{cases} \vec{f}_i^{\,j} & 1 \le i < w \cdot h \\ \omega \cdot t_{norm}^j & i = w \cdot h + 1 \end{cases} \quad (1)$$

where $\vec{f}$ is the vector form of f^t, $\vec{f} = f^{(x,y)}$, $1 \le x \le h$, $1 \le y \le w$, h and w are the image height and width, ω is the time dimension weight, t_{norm} is the normalized time vector, $t_{norm} = \frac{t - \bar{t}}{\sum t}$.

However, in fact, the clustering cannot work well if it is directly conducted upon original high-dimensional data. In Eq. (1), $\vec{f}_i$ is a raw video frame. In order to achieve good performance, the dimension of the input vector should be reduced substantially. Despite this reduction, the action information in the raw data should be preserved as much as possible. In most cases, the input data is highly redundant. Hence, we perform dimensionality reduction employing the signal's sparse prior.

Compressed sensing [41,42] is an advanced signal processing technique. In this paper, we use a sparse measurement matrix [43] R which possesses the Restricted Isometry Property (RIP) that reserves the metric distance of data samples in a low-dimensional subspace to effectively convert the original data from the image space to a low-dimensional, compressed space.

We first construct a random sensing matrix [43] by

$$r_{(i,j)} = \sqrt{3} \times \begin{cases} -1 & with\ probability\ \frac{1}{6} \\ 0 & with\ probability\ \frac{2}{3} \\ 1 & with\ probability\ \frac{1}{6} \end{cases} \quad (2)$$

where $r_{(i,j)}$ denotes an element in R. It has been proven that matrix R satisfies the Johnson–Lindenstrauss lemma [44], which theoretically guaranteed its Restricted Isometry Property (RIP).

Then, we conduct dimensionality reduction from $\vec{f}^j$ to x^j, by multiplying it with the sparse matrix R:

$$x^j = R\,\vec{f}^{\,j} \quad (3)$$

x^j is the dimension-reduced vector.

We replace $\vec{f}^j$ with x^j for clustering. An example is demonstrated in Fig. 2.

Next, action sequence S is segmented into small sub-sequences:

$$S = \{s^1, s^2, \ldots, s^n\} \quad (4)$$

$$s^i = \left\{f^1, f^2, \ldots, f^m\right\} \quad (5)$$

where s^i is a sub-sequence of video sample S, and f^j denotes the frame of s^i.

3.2.2. Short-memory sequence generation—Short-memory sequence (SMS) is designed for two purposes. First, different instances of the same action may have various relative velocities. Short-memory sequence normalize speed variations (question 2 mentioned in Section 1), and rapidly achieve week alignment of sequences.

Another important function of SMSs is dealing with long-term issues. In fact, SMSs simulate the Short-Term Memory [45] of humans' cognition system. In 1950s, George Armitage Miller, one of the founders of cognitive psychology, conducted a series of quantify

research with respect to the short-term memory capacity. One most famous conclusion is that humans' working memory should not be measured by the amount of information, but the numbers of "*Chunks*" [46,47]. Further speaking, human could hold approximately seven chunks (so it is called "the magical number seven") in short-term memory.

The mechanism behind this conclusion is quite attractive. It means that when dealing with high-dimensional and rapidly generating data, the working pattern of the human brain is not running through all information, but processing several small chunks of information each time. This conclusion may partially contradict some viewpoints many studies followed, e.g., making the sequence recognition as long as possible. However, this is an effective pattern, particularly when facing very long-term tasks.

Inspired, we artificially generate SMS chunks. The method is discrete sampling. In this paper, we propose a *Gaussian Sampling* method.

Gaussian sampling: The "Gaussian Sampling" can also be called the *Gaussian contour sampling*. It assumes that the importance of frames before and after a key-frame satisfies two normal distribution with σ_1 and σ_2. Sampling points are just at the Gaussian curve's contours (Fig. 3). For different lengths of subsequences, *sigma* parameters are used to adjust sampling intervals, so as to bridge the gap of different cases. The details are shown in Algorithm 1.

In Algorithm 1, y_{tc} is a predefined truncation value (the Motion-bias Correction Parameter, or MCP). Intuitively, it controls the Gaussian curve's shape. The larger this value is, the distance between each sampling point is more uniform, which means the memory' forgetting degrees is set to be weaker and key-frames' effect is therefore less salient, vice versa.

Algorithm 1

Gaussian sampling.

INPUT: $S = \{s^1, s^2, \ldots s^n\}$, $s = \{f^1, f^2, \ldots f^m\}$, $k = \{k^1, k^2, \ldots k^n\}$, y_{tc};

for each k^i **do**

set $y^l_{edge} = npd(x^l_{edge}, 0, \sigma_l), y^l_{edge} = y_{tc}$, calculate the σ_l;

$\Delta y_l = npd(0, 0, \sigma_l)/3$

set $y^r_{edge} = npd(x^r_{edge}, 0, \sigma_r), y^r_{edge} = y_{tc}$, calculate the σ_r;

$\Delta y_r = npd(0, 0, \sigma_r)/3$

$x_0 = 0$;

for $-3 \le k < 0$ **do**

$x_k = norminv(k\Delta y_l, 0, \sigma_l)$;

end for

for $0 < k \le 3$ **do**

$x_k = norminv(k\Delta y_r, 0, \sigma_r)$;

end for

end for

OUTPUT: $X = x_k \mid -3 \leq k \leq 3$.

The MCP's setting is based on statistics. We extracted key-frames from more than 20,300 video clips from which the average interval between neighboring key-frames is 12.25 frames (in 25fps). When the MCP is set to be large (greater than 0.03), the variance of Gaussian distribution increases considerably. Then, the Gaussian sampling in this situation is actually close to an average sampling (temporal average pooling) within memory spans, which means it has lost the ability to normalize motion bias. However, if this value is too small (lower than 0.01), sampled frames will gather around the moment that is far from the key-frame. This will cause a situation that motion-smooth frames impact is too large, while the effects of key-frames are suppressed. This will significantly reduce the recognition performance. And it may cause instability in the training process. After careful adjustments, we found that the recognition accuracy was not significantly changed when the MCP was between 0.01 and 0.025. Considering the long training time, we adopted a relatively moderate value (0.02) empirically as the recommended truncation value.

It should be noted that one SMS's border may touch adjacent sub-sequence (s^{i+1}) in order to achieve *associated memory* (or context). This elastic structure enlarges the range of associated memory.

3.2.3. Data packaging—Next, we conduct hybrid data packaging. The final data structure can be represented as "RGB+XY+T", where "RGB" denotes the color channel, "XY" is the optical flow field, and "T" is the time dimension. It is constructed by both raw and artificial data.

We first extract the foreground region for each frame f^i in SMS, then calculate the optical flow displacement field. The complete structure of the output data is as follows:

$$O^i = \langle C_r, C_g, C_b, X, Y \rangle, \quad (6)$$

where C_r, C_g and C_b are color channels. X and Y are corresponding optical-flow field components in x and y directions.

Up to here, we have completed the data-preparation. The following is layer-wise feature extraction.

3.3. Local (in time) feature extraction

The learning module of ARCH framework is constructed by two main parts. As demonstrated in Fig. 1, the lower part is a group of *local space–time feature extractors*, and the upper part is a *sequence dynamics extractor*. We introduce them separately in two sections (Sections 3.3 and 3.4) for ease of description, but actually they tightly work together.

In this section, we propose a Temporal–Spatial-Fusion ConvNet (TSF-CNN) framework, which is designed for learning combined short time action representations in the spatial and temporal domains. We redesign a convolutional neural network, and apply it to the short-term action feature extraction.

Convolutional Neural Networks (CNNs) are famous for their powerful features learning abilities. And CNN-based methods achieve superior results on many static visual recognition tasks [23,25,27]. However, in video tasks, dynamic features are more important. A local (in time) dynamic feature captures short time motion patterns, such as simple movements, single motions or poses.

We consider extending CNN to the domain of dynamic feature learning. However, original CNNs are designed for static tasks such as image classifications and sense recognition, it is not appropriate to apply it directly to non-static/context-sensitive tasks. A challenge is how to embed motions information into existing static-feature learner architecture.

The Temporal–Spatial-Fusion ConvNet (TSF-CNN) adopts a double-channel structure (Fig. 4): one Spatial-Net (S-Net) and one Temporal-Net (T-Net), which are abreast laid on the lower part of the model. The S-Net performs stacked 2d convolution on feature maps from image data, while the T-Net performs 3d convolution on the movement vector fields. Then, the Fusion-Net (F-Net) fuses static and dynamic informations from these two nets.

3.3.1. S-Net: stacked volume convolution in spatial domain—The S-Net is a stacked multi-layers convolutional network. It focuses on images' spatial information.

The structure of the S-Net (Fig. 5) is similar to the classical LeNet [48] model. Stacked 2D filters capture spatial features of images, i.e., edges, corners, and more complex patterns. However, the convolution operation in S-Net is organized in *volume*, i.e., a group of convolution kernels separately corresponds to a volume of feature maps.

The *volume convolution* operation is formalized as follows:

$$v^{(x,y)} = \max\left(0, b + \sum_{m=1}^{M}\sum_{p=0}^{P-1}\sum_{q=0}^{Q-1} w_m^{pq} v_m^{(x+p)(y+q)}\right) \quad (7)$$

where max(0, x) is the rectified linear unit (ReLU) activation, b is the bias, m denotes the number of feature-maps, P and Q are the height and width of one feature map, ω is the weight matrix.

We perform volume convolution operations on feature maps, which are originally derived from full images, i.e., including not only the foreground (movement region), but also the background. This is different from some methods which discard background and purely focus on movement regions. We argue that background regions provide important sense context information, which is helpful for recognition. For example, the motion patterns of "*Diving*" and "*TrampolineJumping*" are quite similar (Fig. 6), but they can be easily

distinguished by human because the background environments are quite discriminative. Hence, we insist on reserving background information.

In our framework, the S-Net consists of five main layers, each of which has different sub-layers such as convolution (c), pooling (p), activation (a), and normalization (n). The detailed configurations of the S-Net are listed in Table 1.

3.3.2. T-Net: stacked cube convolution in temporal domain—In the S-Net, we use 2d convolution to capture spatial features. The T-Net extracts motions' movement patterns on multiple contiguous frames (Fig. 7). This is achieved by *cube convolution* operations upon 3d spatio-temporal data.

Formally, the *temporal-spatio cube convolution* is represented as follows:

$$v^{(x,y,z)} = \max\left(0, b + \sum_{m=1}^{M}\sum_{t=0}^{T}\sum_{p=0}^{P}\sum_{q=0}^{Q} w_m^{pqt} v_m^{(x+p)(y+q)(z+t)}\right) \quad (8)$$

where max(0, x) is the ReLU activation function, b is the bias, m denotes the number of feature-maps, P and Q are the height and width of a feature map, t is the temporal size of the SMS.

The cube convolution is similar to [30], however, we do not have overlapping stride on the time dimension, and use the ReLU rather than hyperbolic activation.

The optical flow is focused on the foreground. It does not take the background into consideration. This is based on the prior knowledge that key-motions are mainly on foreground regions. This approach can reduce interference, because the background may also have movement but it is less useful for recognition.

The T-Net in our framework also contains five main layers. Configurations of all sub-layers are listed in Table 1.

3.3.3. F-Net: spatial and temporal information fusion—The F-Net is a two-layer full connected network (Fig. 8), which is designed for feature fusion and selection.

When used as a part of the ARCH network, the F-Net does not have a classification layer. However, for the purpose of step-wise training (Section 4.1), it needs to add a loss layer during training temporarily. In our implementation, we use *softmax* with L1 regularization for feature selection. Hence, the complete loss function is as below

$$l(f(x;\theta), y) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1_{(y=j)} \log \frac{e^{a(x)}}{\sum_{l=1}^{k} e^{a(x)}}\right] + \lambda \sum_{i=1}^{k} |\theta| \quad (9)$$

where $a(x) = g(Wx+b)$ and $\theta = \{W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}, \ldots, W^{(l+1)}, b^{(l+1)}\}$.

Considering the magnitude of parameters, we have to carefully handle overfitting. In F-Net, we leverage a feature noising technique (the dropout training), to overcome overfitting.

The TSF-CNN can be used in standalone when state and sequence problems are not considered, e.g., short animations' classification or dynamic pictures' recognition. However, since we apply it to the long-term video tasks, time-sequences cannot be neglected. We need to model sequence dynamics.

In the ARCH network, a group of TSF-CNNs are ruled by a Recurrent Neural Network (RNN).

3.4. Sequence dynamics extraction

In action recognition, errors may happen when different categories of actions possess similar sub-actions. This is a typical class of errors, which is caused by intra-class and inter-class similarity. Some standard methods such as video pooling or sampling are challenged when applied to encode long-term actions, partly because these methods do not consider the relationship between sub-actions. However, the sequentiality provides important support for solving this problem.

The Recurrent Neural Network (RNN) [34,35] is a powerful tool for time–sequence modeling. Comparing with other approaches such as HMM, the RNN provides further data-driven character and generalization.

As an abstract dynamical system, RNN could well represent high-dimensional hidden states and non-linear dynamics [36,34]. The special *rollback* structure (Fig. 9) between hidden and input layers allows hidden neurons to remember the history of previously processed status. Theoretically, an RNN with a sufficient number of hidden units can approximate any measurable sequence-to-sequence mapping to arbitrary accuracy [49]. A standard RNN's [50] output is calculated by the following equations:

$$s(t)=f(Ux(t)+Ws(t-1)+b_s) \quad (10)$$

$$y(t)=g(Vs(t)+b_y) \quad (11)$$

However, although it is superior in theory, some drawbacks limited its usage in some real-world tasks. For example, RNN is difficult to train. Some studies attribute the training problem to the reason of "pathological curvature problem" [51]. Hence, some 2nd-order optimization methods are proposed to substitute 1st-order optimization methods. However, recent studies show that previous attempts at training RNN failed partly due to improper random initialization methods [34]. This opinion directly contradicts widespread beliefs about the inability of 1st-order methods.

We consider this problem in another way. Most approaches directly train RNN upon raw data, however, we prefer the RNN modeling high-level (more abstract) information.

Although RNN has a wide range, the depth of model's abstraction is not sufficient. A deeper feature pre-extraction can get rid of the unnecessary information, and help the RNN to work.

3.4.1. Memory-weights-independent RNN—The TSF-CNN is an initialization for high-level feature extraction. In our work, RNN receives input data from TSF-CNNs' output. This design reduces sequences' length, so as to help RNN overcome training problems.

Furthermore, we make an improvement to the standard RNN model. In the original RNNs, weight parameters are shared over different time steps, so as to avoid the parameters grow exponentially with the length of sequence. The price to be paid for that advantage is that optimizing the parameters is more difficult. However, in ARCH network, thanks to the SMSs and TSF-CNNs, the sequences imputed into the RNNs are much shorter than original sequences. We hence can use non-shared parameters in ARCH networks.

As shown in Fig. 10, in the ARCH network, an RNN is connected with multiple TSF-CNNs. For each time step t, the output of TSF-CNN is a vector $x(t)$. Correspondingly, it is multiplied with a weight matrix U_t, which is specially for the current time-step and non-shared with others.

The outputs of the hidden layer and the output layer are computed as follows:

$$s(t)=f(U_t x(t)+W s(t-1)) \quad (12)$$

$$y(t)=g(V s(t)) \quad (13)$$

For the reason that SMSs are much shorter (usually less than 10 steps) than original video frames, it suffers less from weight decay. This advantage is more significance for long actions. Meanwhile, owing to independent weights, it is more dedicated to sequence states' changes.

The RNN's output is a combined representation with respect to the input video. And this representation is generalized for typical video tasks such as video classification, action recognition, and video parsing. In classification tasks, RNN's output vector is connected to a classification layer so as to convert the output into a probability distribution with respect to categories. We simply use a softmax layer here. The RNN is trained by standard BPTT algorithm [52,53]. The final output is a predicted probability distribution (a 101 dimensions vector) w.r.t. the action's category.

4. Implementation details

The ARCH network is a big model, which is not only on aspect of model's depth (abstraction levels), but also on width (temporal context range). One advantage of large models lies in the information capacity, meaning that it can store knowledge in a larger capacity, and utilize more extensive context information. However, drawbacks are also

obvious. It is computationally time-consuming, and may suffer from *vanishing gradient* (or, unusual, *explosion*) problem, which is commonly believed to be one factor of training failure.

Although these problems are inherent, we take some necessary measures to deal with them. Approaches are both in aspects of models' design and training method.

4.1. Step-wise training and global fine-tuning

The ARCH network is stepwise trained, which means we do not optimize the whole model simultaneously, but train each part asynchronously. Generally, this is achieved by three stages.

Stage 1: Train TSF-CNNs group separately. We first add a classification layer to the top of the TSF-CNN, and use the SGD algorithm to update weights at each back-propagation progress. We use 64 samples each mini-batch (momentum=0.9, dropout rate=0.9). The learning rate is initially set to be 10^{-2}, and this value is decreased to 1/5 every 10,000 iterations.

Stage 2: Train RNN separately. When the test accuracy outperforms an acceptable level (70%, set by us), the RNN's training is activated. We discard the temporary classification layer of TSF-CNN, and connect the penultimate layer to the RNN. During this stage, gradients of errors are temporarily not continually downward propagate to the TSF-CNNs.

Stage 3: Global fine-tuning. This is achieved by rounds of global back-propagation. The learning rate is kept on 10^{-3}. The error is back propagated from RNN to TSF-CNN. After 12,000 iterations, one round of training completes.

This training method is more complex than directly end-to-end training from the beginning. However, our test demonstrates that, achieving similar accuracy, this method costs less time. Although a theoretical proof could not be given so far, intuitively, this approach can be seen as optimization by adding supervised information step-wisely, so as to boost each parts' performance and achieve a global optimization finally. Besides, another advantage is that it allows us solely pre-training some modules on other more diversified dataset, which plays an important role in boosting robustness.

Stages from 1 to 3 constitute one round of whole model training. It can obtain better performance if we perform several rounds of training, especially combined with some tricks such as artificial data augmentation and pre-training on S-Net and T-Net. However, that depends on the expectation between time consuming and model accuracy.

4.2. Distributed implementation

The TSF-CNN is implemented based on the open-source Caffe project [54], and the RNN is based on the RNNLM [55]. We conduct many customized modifications.

Due to huge computational complexity, we utilize distributed computing techniques to accelerate the processing. One latent property of ARCH network is that its parallelism in both model and data. In our implementation, computations are physically distributed among

multiple servers. Different from some popular architectures that use one server with multiple GPUs, we employ a distributed GPU cluster for offloading computation.

The system follows a *master–slave* style. One server runs RNN, and the other three run TSF-CNNs. They are connected by 10 Gbps ethernet. To eliminate the time delay caused by data handling, each server reserves a copy of the entire dataset. They pass messages and necessary intimidate results among nodes.

Totally, we trained 10 TSF-CNNs, 9 for each key-motion step, and 1 for any else. Notice that TSF-CNNs are not trained simultaneously. We first train one TSF-CNN. After 1 round of whole samples iteration, the next TSF-CNN's training is started. One-by-one, the former provides initialization for the next. This is for the purpose of overcoming the bias in samples amount, and decreasing the time-consumption.

The whole framework is developed by C+ + for efficiency. The training lasted 18 days.

5. Experiments

We focus on the UCF101 [10] dataset to evaluate our methods. The UCF101 dataset contains 13,320 clips (101 actions categories) which are collected from internet videos. Samples in UCF101 dataset are unconstrained real videos, and they are unprocessed with camera motion, various lighting conditions, partial occlusion, low resolution, etc. Besides, we also evaluate our work on other popular datasets, including HMDB51 [56], KTH [57], and Holly-wood2 [58] dataset.

5.1. Results

5.1.1. Tests on the TSF-CNN—We first evaluate the performance of TSF-CNN. Some early works demonstrated that training ConvNet on single dataset may cause over-fitting [14]. Considering that, we pre-train the TSF-CNN on ILSVRC-2012 [59] (just for the S-Net) and HMDB51 [56] (just for the T-Net) dataset.

In Table 2, the TSF-CNN is compared with some state-of-the-art methods, which can be categorized into two classes. Refs. [10,60–62] are all feature-designing-based methods using HoF, DCS descriptor, combined multi-descriptor or trajectory as basic features. Refs. [11–14,64] are all feature-learning-based methods employing CNN, 3D-CNN, heterogeneous-data CNN, or LSTM as core feature extractor. The results for UCF101 and HMDB51 dataset are reported in Table 2. For a given video, each TSF-CNN at t output a prediction. We report the dominant values to evaluate the performance of TSF-CNN.

As shown in Table 2, the TSF-CNNs' recognition accuracy outperforms state-of-the-art feature-designing-based and deep-learning-based methods. Specially, we examine the differences between deep-learning-based methods. Ref. [11] uses temporal convolution and the "Slow Fusion" technique, which is analogous to our S-Net and F-Net. Ref. [12] uses 3D convolution that is similar to the cube convolutions in T-Net although they are quite different in detail. However, [11,12] do not have motion information embedding. Although they have deeper network configurations, we outperform these methods by more than 8%. Besides,

both [14] and TSF-CNN adopt juxtaposed networks to accommodate motion information. TSF-CNN have a slightly better performance.

Compared with these methods, we speculate that optical flow (or prior-knowledge-based data preprocessing) plays a crucial role in TSF-CNN. To verify this speculation, we examine the contribution of motion information in TSF-CNN. Two scenarios are compared: (a) discarding T-Net's output in F-Net; (b) discarding S-Net's output in F-Net. Then, we retrain the TSF-CNN. The results are shown in Table 3.

From Table 3, we can observe that the TSF-CNN without T-NET is significantly weaker than integrated TSF-CNN. This test quantitatively verified T-Net's importance: motion information embedding is indispensable in our model. Some similar conclusions also can be found in [14,8].

5.1.2. Tests on the ARCH network—Furthermore, we evaluate the ARCH network, and take the UCF101 dataset as a benchmark test. We run seven rounds of stepwise training as introduced in Section 4.1.

Comparison with the state-of-the-art methods: We make a comparison with some state-of-the-art works, including "Improved Trajectories" [8], "Two-stream" ConvNet [14] and "Slow Fusion" CNN [11]. These methods could be categorized into two classes: hand-crafted feature methods and feature learning methods.

As can be seen in Table 4, ARCH outperforms all other state-of-the-art methods. Refs. [11–14,29] are deep-learning-based methods. Specially, [29] also follows a hybrid network architecture, which use CNN and LSTM as stacked feature extractor. Ref. [13] is a novel unsupervised method for video representation. Nevertheless, according to its report, its performance relies more on data supply. Generally, they both do not have motion information embedding. However, akin to ARCH network's architecture, they also take videos' global sequentiality into consideration.

Therefore, we investigate the contribution of each part in ARCH networks. In Fig. 11 we can observe that, in quality, the RNN's contribution is not so noticeable. However, we found that some samples that are misclassified by TSF-CNNs are corrected by RNN. And most of these samples contain some sub-actions which are easy to be confused in some individual time points. Hence, we argue that RNN plays a rectification role upon TSF-CNNs.

Transfer learning: To evaluate the generalization performance of ARCH networks, we perform cross-dataset evaluation. Leveraging the transfer learning techniques, we examine ARCH network's performance on three popular datasets: KTH [57], Hollywood2 [58] and HMDB51 [56] dataset.

We examine two distinct forms of approaches: (1) retraining all layers from scratch; (2) retraining the F-Net of TSF-CNN and the classification layer which is upon the RNN, then conducting global fine-tuning. To our surprise, the second method performs better than the first one. Hence we report the results achieved by the second method (listed in Table 5).

In Table 5, we can observe that ARCH network's performance is comparable or better than the state-of-the-art techniques.

5.2. Analysis

Recall the three problems which are stated at the beginning of the Introduction section.

5.2.1. Spatial domain variations—Many studies have demonstrated CNN's superiority on learning robust features from raw images [23,66]. A common strategy used for boosting the robustness of feature extractors and resisting over-fitting is "data augmentation" [23]. To make the feature extractor steadier to spatial domain interference, we add extra training data, which simulate the variations in spatial domain.

In ARCH network, we adopt two methods to artificially augment datasets. The first method is adding spatial interference (image blur, random occlusion and noises) on original samples. It also conducts image mirroring, scaling and cropping operations during training. The second method inserts fragments from other clips, which may belong to the same or different categories (1:1.5 in quantity). The former aims at boosting TSF-CNN's robustness, and the latter is used to reinforce the RNN's discrimination to stochastic destabilization. Fig. 12 demonstrates an example. Totally, we randomly processed 2525 clips (nearly 19% of the whole dataset) using the first method, and added 1500 fragments (5–8 frames per fragment) for artificial confusion.

As reported in Table 6, we can observe that data augmentation demonstrated effects to the recognition performance. By this method, TSF-CNN's robustness to spatial interference is reinforced. And the RNN is more stable to interferences in temporal domain.

5.2.2. Temporal domain polytropes—We employ SMSs to normalize speed and compress data. It should be noted that the SMSs are not used for achieving precisely aligned sequences. This is because precise alignments are time-consuming, however, our application scenario is rapidly generating videos. As a trade-off, the ARCH network is trained on weak alignment sequences.

We make a comparison with some precise alignment methods. We use the DTW [9] and the GTW [67] method to substitute the SMSs, and set the same sampling rate as SMSs. Then we retrain the ARCH network from scratch. The results are reported in Table 7.

Our experiments demonstrated that, for ARCH networks, temporal alignment is important for recognition, however, precise alignment is not very necessary. This is because we employ RNN to encode sub-actions, while precise alignments are more important to video pooling methods.

5.2.3. Inter-class and intra-class correlations—The fundamental reason of employing RNNs to enlarge interclass differences is that RNNs are used for modeling the probability distribution of sub-actions' relationships.

RNNs can be viewed as a directed graphical model [68]. The random variable is the sequence of vectors $X = (x_1, x_2, \ldots, x_T)$. For a classification problem, we want to achieve the joint probability of a sequence of variables.

A common strategy used for achieving video-level classification is some video pooling-based methods [69]. For these methods, generally, $P(X)$ is estimated using a subset of sequence states

$$P(X)=P(x_{s_1}, x_{s_2}, \ldots, x_{s_m}) \quad (14)$$

where s_i denotes a sampling point.

However, for standard RNNs, the graphical model is generally fully connected, not discarding any dependency a priori. Moreover, $P(X)$ is estimated by

$$P(X)=P(x_1)P(x_2|x_1)P(x_3|x_2, x_1)\ldots P(x_t|x_{t-1}, \ldots, x_1), \quad (15)$$

which decompose the joint probability of a sequence of variables into ordered conditionals precisely corresponds to the sequence of computations performed by an RNN [68]. Hence, the joint probability achieved by RNN is more discriminative to sub-actions' mutual relationships.

To quantitatively evaluate the discriminative ability, we substitute the RNN for a video pooling operation, and calculate the cosine similarity of feature vectors of intro-class and inter-class exemplars, separately using the video pooling and the RNN. The results are shown in Fig. 13.

In Fig. 13(a), we can observe that similarity distributions achieved by video pooling are far from 0, and distribute over a large range. These observations indicate that features of inter-class and intro-class exemplars are low discriminative. However, Fig. 13(b) demonstrates the results achieved by the RNN method. We can observe that the similarities of a large proportion of inter-class exemplars are very close to 0, which means that the inter-class variations are more significant. Meanwhile, the overlapping of these two distributions is smaller than Fig. 13(a), which means that the inter-class and intra-class correlations are enlarged.

5.2.4. Further discussions

(a) Accuracy vs. sequence length: We found that ARCH networks are good at recognizing long sequences. Fig. 14 reports recognition results on samples of different lengths. Fig. 14(a) exhibits detailed results on split 1, including samples' number, proportion and recognition accuracy. Fig. 14(b) records accuracies' tendency, and samples' quantitative distribution across three test splits.

To the best of our knowledge, the relationship between recognition precision and sequences length was not studied adequately in previous works. We reached out the authors of other methods to obtain comparison results. Unfortunately, these results were unavailable. Hence, we could not give a comprehensive evaluation so far. However, it can be observed in Fig. 14 that recognition accuracy on long sequences is better than short ones, although former's samples are relatively less. The best results achieve 90%. To some extent, this refiected ARCH network's advantage on long sequence recognition.

(b) Error analysis: Then we investigate the error cases. Some misclassification cases are listed in Fig. 15(a). It could be found that most of error cases share some common characteristics, e.g., actions and scenes are both very similar (row 1 vs. row 2), actions are short and fast repeated (row 3), or actions' region is very small and complexity is relatively low (row 4). On this point, we believe that our SMS selection method still has improvement space.

Interestingly, we also found some good performance on several hard decided cases (Fig. 15(b)), such as fine-grained classifications and long-term actions. For example, row 1 and row 2 are "Breaststroke" and "FrontCrawl", where their senses are almost the same, and actions have finely differences (similar cases are "Dive" and "TrampolineJumping"). Another case is "HighJump" and "Javelin", which are both long-term actions and share the similar "run-up" progress (about 2/3 length of the whole sequence). On this point, we argue that SMS and the RNN have a positive effect.

(c) Data consumption: Another key point is that achieving this result we use less data (Fig. 16).

For example, the original number of frames in the class "*ApplyEyeMakeup*" is 24,358. With the help of SMS mechanism, only 10,437 frames (about 43%) are enough for our method. This means that it just needs to store SMS data instead of complete videos, which could be regarded as a summary-like storage. This character is meaningful in production environment, especially in internet scale.

On this point, we make a statistics to compare the amount of original data and SMS data. The details are listed in Fig. 16. It can be observed that the amount of data of SMSs are much less at all classes. The average data usage is 40.95% of the original video data.

5.3. Visualization

In this part, we provide a visual demonstration on features learned by TSF-CNN.

In Fig. 17, the first row includes 7 image frames, which are the image channels (C_r, C_g, C_b) of one SMS. The second row is correspondingly 7 feature maps derived from the first convolutional layer. They come from one volume filter kernel.

Fig. 18 shows the T-Net features. The first row lists 7 frames which are the optical flow channels (C_x, C_y) of a SMS. The second row lists corresponding 6 feature maps derived

from one cube convolution kernel. The temporal stride of cube convolutional kernel is 2. They are normalized to range(0, 255) for a better display.

6. Conclusion

This paper focuses on improving the long-term action recognition. Our idea is both on aspects of data preparation and feature extraction. Utilizing integrated information and efficient feature learning method to solve several important problems in action recognition domain.

Our work can be concluded as follows:

- **A.** We use unsupervised cluster and data reduction to address the data amount and temporal multiscale problem. This idea helps us manage the challenge from long-term and high-dimensional data.
- **B.** We design an efficient architecture, the TSF-CNN, an upgraded convolutional neural network, for local static and dynamic information abstraction. Furthermore, a “network up networks” structure makes an RNN rules a group of TSF-CNNs, which seamlessly connects local feature extraction and global sequence pattern modeling.
- **C.** We develop a training method to optimize this big model, and utilize some tricks to boost its generalization. Besides, we develop a distributed training architecture.

Finally, we conduct a suit of experiments. The result demonstrated that we achieve a leading result at the UCF101 dataset.

On methodology, we believe that a problem-oriented learner designing and efficient optimization method are both very important.

In future directions, we plan to train our model on larger datasets (e.g., the Sport1M dataset) which contain more action categories and rich changes. This could help the model achieve better generalization ability on big data. Besides, we will establish a long-term action dataset for the further studies. We also have the plan to dock the ARCH network with an object tracking system.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant no. 61272351, and National Institutes of Health Grants R01CA165255 and R21CA172864 of the United States.

We gratefully acknowledge the support of NVIDIA Corporation and Perry Deng (NVIDIA) with the donation of the GPUs used for this research. Thanks to Haoran Li, Zehao Huang and Xin Liu for fruitful discussions.

References

1. Weinland D, Ronfard R, Boyer E. A survey of vision-based methods for action representation, segmentation and recognition. Comput Vis Image Underst. 2011; 115(2):224–241.
2. Aggarwal J, Ryoo MS. Human activity analysis: a review. ACM Comput Surv. 2011; 43(3):16.

3. Bourdev, L., Malik, J. Poselets: body part detectors trained using 3d human pose annotations. International Conference on Computer Vision (ICCV); 2009.

4. Bourdev, L., Maji, S., Malik, J. Describing people: a poselet-based approach to attribute classification. 2011 IEEE International Conference on Computer Vision (ICCV); Barcelona, Spain: IEEE; 2011. p. 1543-1550.

5. Schindler, K., Van Gool, L. Action snippets: How many frames does human action recognition require?. IEEE Conference on Computer Vision and Pattern Recognition, 2008, CVPR 2008; Alaska, USA: IEEE; 2008. p. 1-8.

6. Li, L-J., Fei-Fei, L. What, where and who? classifying events by scene and object recognition. IEEE 11th International Conference on Computer Vision, 2007, ICCV 2007; Rio de Janeiro, Brazil: IEEE; 2007. p. 1-8.

7. Wang, H., Klaser, A., Schmid, C., Liu, C-L. Action recognition by dense trajectories. 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); Colorado Springs, USA: IEEE; 2011. p. 3169-3176.

8. Wang, H., Schmid, C. Action recognition with improved trajectories. 2013 IEEE International Conference on Computer Vision (ICCV); Sydney, Australia: IEEE; 2013. p. 3551-3558.

9. Veeraraghavan, A., Chellappa, R., Roy-Chowdhury, AK. The function space of an activity. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition; New York, NY, USA: IEEE; 2006. p. 959-968.

10. Soomro K, Zamir AR, Shah M. Ucf101: a dataset of 101 human actions classes from videos in the wild. arxiv:hepth/1212.0402.

11. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L. Large-scale video classification with convolutional neural networks. IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2014.

12. Tran D, Bourdev L, Fergus R, Torresani L, Paluri M. C3d: generic features for video analysis. Eprint Arxiv.

13. Srivastava N, Mansimov E, Salakhutdinov R. Unsupervised learning of video representations using lstms. arxiv:hepth/1502.04681.

14. Simonyan K, Zisserman A. Two-stream convolutional networks for action recognition in videos. Advances in Neural Information Processing Systems. 2014:568–576.

15. Aggarwal, JK., Cai, Q. Human motion analysis: a review. Nonrigid and Articulated Motion Workshop, 1997, Proceedings; San Juan, Puerto Rico: IEEE; 1997. p. 90-102.

16. Blank, M., Gorelick, L., Shechtman, E., Irani, M., Basri, R. Actions as space–time shapes. Tenth IEEE International Conference on Computer Vision, 2005, ICCV 2005; Beijing, China: IEEE; 2005. p. 1395-1402.

17. Weinland D, Ronfard R, Boyer E. Free viewpoint action recognition using motion history volumes. Comput Vis Image Underst. 2006; 104(2):249–257.

18. Nguyen, NT., Phung, DQ., Venkatesh, S., Bui, H. Learning and detecting activities from movement trajectories using the hierarchical hidden Markov model. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, CVPR 2005; San Diego, CA, USA: IEEE; 2005. p. 955-960.

19. Zhang D, Gatica-Perez D, Bengio S, McCowan I. Modeling individual and group actions in meetings with layered hmms. IEEE Trans Multimed. 2006; 8(3):509–520.

20. Liang, X., Lin, L., Cao, L. Learning latent spatio-temporal compositional model for human action recognition. Proceedings of the 21st ACM International Conference on Multimedia; Barcelona, Catalunya, Spain: ACM; 2013. p. 263-272.

21. Ryoo, M., Aggarwal, J. Recognition of high-level group activities based on activities of individual members. IEEE Workshop on Motion and Video Computing, 2008, WMVC 2008; Cooper Mt, USA: IEEE; 2008. p. 1-8.

22. Gong, S., Xiang, T. Recognition of group activities using dynamic probabilistic networks. Ninth IEEE International Conference on Computer Vision, 2003, Proceedings; IEEE; 2003. p. 742-749.

23. Krizhevsky, A., Sutskever, I., Hinton, GE. Imagenet classification with deep convolutional neural networks. In: Pereira, F.Burges, C.Bottou, L., Weinberger, K., editors. Advances in Neural

Information Processing Systems. Vol. 25. Curran Associates, Inc; South Lake Tahoe, Nevada, US: 2012. p. 1097-1105.

24. He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. arxiv:hepth/1502.01852.
25. Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R, LeCun Y. Overfeat: integrated recognition, localization and detection using convolutional networks. arxiv:hepth/1312.6229.
26. Girshick, R., Donahue, J., Darrell, T., Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); Columbus, OH, USA: IEEE; 2014. p. 580-587.
27. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Van-houcke V, Rabinovich A. Going deeper with convolutions. arxiv:hepth/1409.4842.
28. Farabet C, Couprie C, Najman L, LeCun Y. Learning hierarchical features for scene labeling. IEEE Trans Pattern Anal Mach Intell. 2013; 35(8):1915–1929. [PubMed: 23787344]
29. Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T. Long-term recurrent convolutional networks for visual recognition and description. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2015. p. 2625-2634.
30. Ji S, Xu W, Yang M, Yu K. 3d convolutional neural networks for human action recognition. IEEE Trans Pattern Anal Mach Intell. 2013; 35(1):221–231. [PubMed: 22392705]
31. Taylor, GW., Fergus, R., LeCun, Y., Bregler, C. Computer VisionECCV 2010. Springer; Crete, Greece: 2010. Convolutional learning of spatio-temporal features; p. 140-153.
32. Robinson, T., Hochberg, M., Renals, S. Automatic Speech and Speaker Recognition. Springer US; 1996. The use of recurrent neural networks in continuous speech recognition; p. 233-258.
33. Tompson J, Jain A, LeCun Y, Bregler C. Joint training of a convolutional network and a graphical model for human pose estimation. arxiv:hepth/1406.2984.
34. Sutskever, I. PhD thesis. University of Toronto; 2013. Training recurrent neural networks.
35. Richard, S. PhD thesis. Stanford University; 2014. Recursive deep learning for natural language processing and computer vision.
36. Sutskever, I., Martens, J., Hinton, GE. Generating text with recurrent neural networks. Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, ICML 2011; Bellevue, WA, USA. 2011. p. 1017-1024.
37. Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., Baskurt, A. Human Behavior Understanding. Springer US; 2011. Sequential deep learning for human action recognition; p. 29-39.
38. Grossman ED, Blake R. Brain areas active during visual perception of biological motion. Neuron. 2002; 35(6):1167–1175. [PubMed: 12354405]
39. Grossman E, Donnelly M, Price R, Pickens D, Morgan V, Neighbor G, Blake R. Brain areas involved in perception of biological motion. J Cogn Neurosci. 2000; 12(5):711–720. [PubMed: 11054914]
40. Rodriguez A, Laio A. Clustering by fast search and find of density peaks. Science. 2014; 344(6191):1492–1496. [PubMed: 24970081]
41. Candès EJ, Wakin MB. An introduction to compressive sampling. IEEE Signal Process Mag. 2008; 25(2):21–30.
42. Foucart, S., Rauhut, H. A Mathematical Introduction to Compressive Sensing. Springer; 2013.
43. Zhang, K., Zhang, L., Yang, M-H. Computer Vision-ECCV 2012. Springer; Firenze, Italy: 2012. Real-time Compressive Tracking; p. 864-877.
44. Achlioptas D. Database-friendly random projections: Johnson–Lindenstrauss with binary coins. J Comput Syst Sci. 2003; 66(4):671–687.
45. Liebe S, Hoerzer GM, Logothetis NK, Rainer G. Theta coupling between v4 and prefrontal cortex predicts visual short-term memory performance. Nat Neurosci. 2012; 15(3):456–462. [PubMed: 22286175]
46. Georgiou HV. Estimating the intrinsic dimension in fmri space via dataset fractal analysis-counting the cpu cores' of the human brain. arxiv:hepth/1410. 7100.

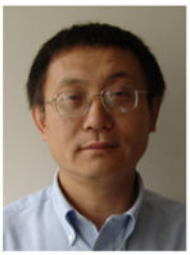
47. Baird, JC. Sensation and Judgment: Complementarity Theory of Psychophysics. Psychology Press; 2014.
48. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998; 86(11):2278–2324.
49. Hammer B. On the approximation capability of recurrent neural networks. Neurocomputing. 2000; 31(1):107–123.
50. Martens, J., Sutskever, I. Learning recurrent neural networks with Hessian-free optimization. Proceedings of the 28th International Conference on Machine Learning (ICML-11); 2011. p. 1033-1040.
51. Martens, J. Deep learning via Hessian-free optimization. Proceedings of the 27th International Conference on Machine Learning (ICML-10); 2010. p. 735-742.
52. Bodén, M. In the Dallas Project, SICS Technical Report T2002:03. SICS; 2002. A guide to recurrent neural networks and backpropagation.
53. Williams RJ, Zipser D. Gradient-based learning algorithms for recurrent networks and their computational complexity. Back-propagation: Theory, Architectures and Applications. 1995:433–486.
54. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T. Caffe: convolutional architecture for fast feature embedding. arxiv: hepth/1408.5093.
55. Mikolov, T. PhD thesis. Brno University of Technology; 2012. Statistical language models based on neural networks.
56. Kuehne, H., Jhuang, H., Stiefelhagen, R., Serre, T. High Performance Computing in Science and Engineering. Vol. 12. Springer US; 2013. Hmdb51: a large video database for human motion recognition; p. 571-582.
57. Schuldt, C., Laptev, I., Caputo, B. Recognizing human actions: a local svm approach. Proceedings of the 17th International Conference on Pattern Recognition, 2004, ICPR 2004; Cambridge, UK: IEEE; 2004. p. 32-36.
58. Marszałek, M., Laptev, I., Schmid, C. Actions in context. IEEE Conference on Computer Vision & Pattern Recognition; 2009.
59. Deng, J., Berg, A., Satheesh, S., Su, H., Khosla, A., Fei-Fei, L. Large scale visual recognition challenge. ⟨http://www.image-net.org/challenges/LSVRC/2012⟩
60. Jain, M., Jégou, H., Bouthemy, P. Better exploiting motion for better action recognition. 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); Portland, OR, USA: IEEE; 2013. p. 2555-2562.
61. Oneata, D., Verbeek, J., Schmid, C. Action and event recognition with fisher vectors on a compact feature set. 2013 IEEE International Conference on Computer Vision (ICCV); Sydney, Australia: IEEE; 2013. p. 1817-1824.
62. Cai, Z., Wang, L., Peng, X., Qiao, Y. Multi-view super vector for action recognition. 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); Columbus, OH, USA: IEEE; 2014. p. 596-603.
63. Wang H, Kläser A, Schmid C, Liu CL. Dense trajectories and motion boundary descriptors for action recognition. Int J Comput Vis. 2013; 103(1):60–79.
64. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014 arxiv:hepth/1409.1556.
65. Sapienza M, Cuzzolin F, Torr PH. Feature sampling and partitioning for visual vocabulary generation on large action classification datasets. arxiv:hepth/1405.7545.
66. Aggarwal J, Ryoo MS. Visualizing and understanding convolutional networks. ACM Comput Surv. 2011; 43(3):16.
67. Zhou, F. Generalized time warping for multi-modal alignment of human motion. 2013 IEEE Conference on Computer Vision and Pattern Recognition; 2012. p. 1282-1289.
68. Bengio, Y., Goodfellow, IJ., Courville, A. Deep learning, book in preparation for MIT Press. 2015. URL http://www.iro.umontreal.ca/~bengioy/dlbook/
69. Aly R, Arandjelovic R, Chatfield K, Douze M, Fernando B, Harchaoui Z, McGuinness K, O'Connor NE, Oneata D, Parkhi OM, et al. The axes submissions at trecvid 2013. TrecVid. 2013

Biographies

Miao Xin received the B.S. degree in Computer Science from HeFei University of Technology, China, in 2009. He is currently a Ph.D. candidate in Image Research Center, School of Astronautics at Beihang University, China. His research interests include deep learning, image understanding and action recognition.

Hong Zhang received the B.S. degree, M.S. degree and Ph.D. degree in Electrical Engineering from Hebei University of Technology, China, Harbin University of Science and Technology, China, and Beijing Institute of Technology, China, in 1988, 1993 and 2002, respectively. She is currently a Professor of the School of Astronautics, Beihang University, Beijing, China. She was in the Department of Neurosurgery at the University of Pittsburgh as a visiting scholar from 2007 to 2008. Her research interests include activity recognition, image restoration, image indexing, object detection and stereovision.

Helong Wang is the chief research fellow of Luoyang electro-optical equipment research institute. His research interests include image enhancement, image restoration and object detection.



Mingui Sun received the B.S. degree from Shenyang Chemical Engineering Institute, Shenyang, China, in 1982, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Pittsburgh, Pitts-burgh, PA, in 1986 and 1989, respectively. In 1991, he joined the faculty of University of Pittsburgh, where he is currently a Professor of neurosurgery, electrical and computer engineering, and bioengineering. His current research interests include advanced biomedical electronic devices, biomedical signal and image processing, sensors and transducers, biomedical instruments, brain–computer interface, electro-neurophysiology, implantable devices, radio-frequency systems for biomedical applications, electronic and data processing systems for diet and physical activity assessment, and wearable computers. He has authored or coauthored more than 300 papers.

Ding Yuan received her B.Eng. degree in Measurement, Control Technology and Instrument, from Harbin Institute of Technology, China, in 2001, her M.Phil. degree and Ph.D. in Mechanical and Automation Engineering from the Chinese University of Hong Kong in 2004 and 2008 respectively. She joined the School of astronautics, Beihang University in September 2009. She is now an assistant professor of Image processing Center at Beihang University. Her research interests include stereo vision, 3D reconstruction, camera calibration, and camera’s ego-motion estimation.

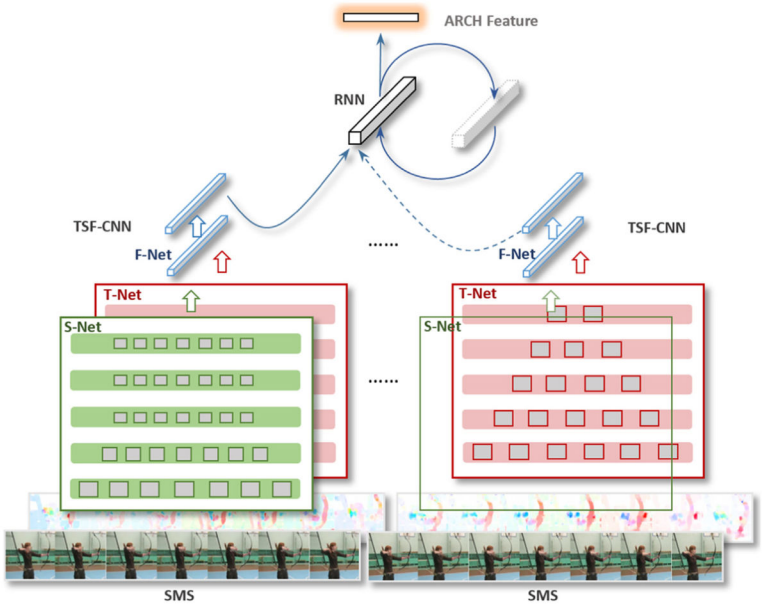


Fig. 1.
Overview of an ARCH network.

Fig. 2.

Key-frames of the video sample “vArcheryg01c03”. The first row is key-frames selected from original video. The second row is corresponding optical flow diagram.

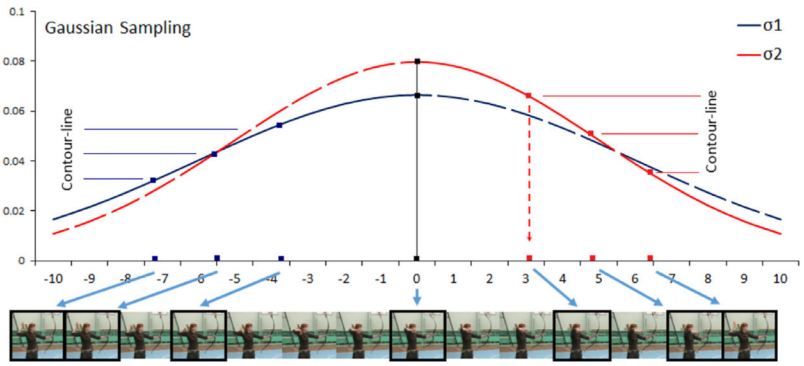


Fig. 3.

Gaussian contour sampling. The *x*-axis is frame number. The frame at *x*=0 is the current key-frame. Given the key-frame, we sample frames from each side of the key-frame.

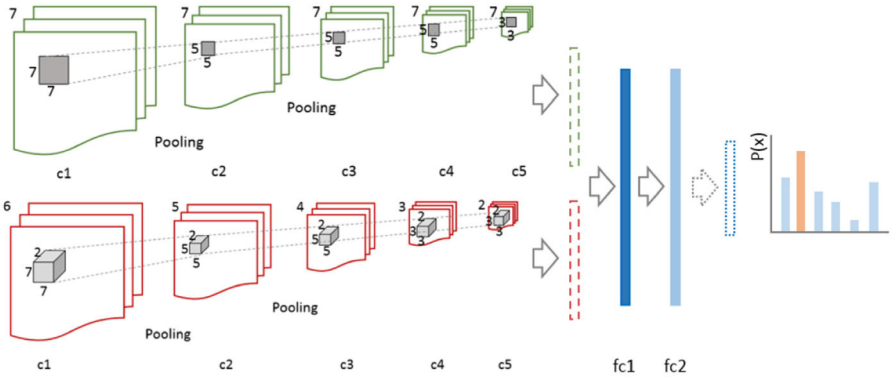


Fig. 4.
Basic structure of TSF-CNN.

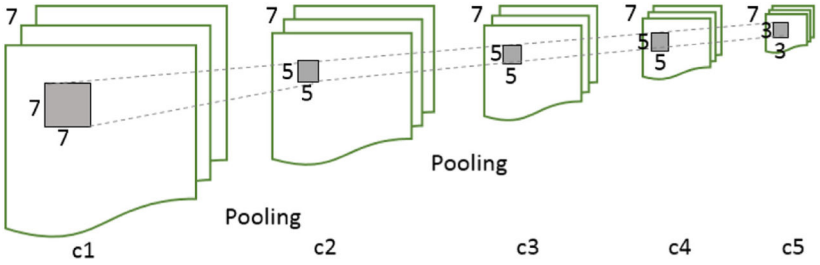


Fig. 5.
S-Net of the TSF-CNN.

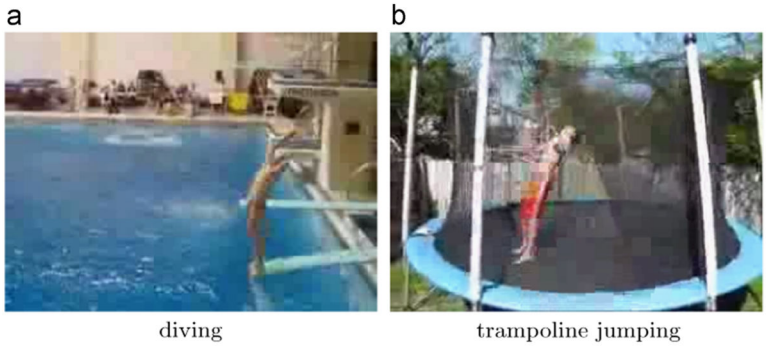


Fig. 6.

(a) is Diving and (b) is TrampolineJumping. Motion patterns are quite similar, but owing to the differences in scene, they can be easily distinguished by human.

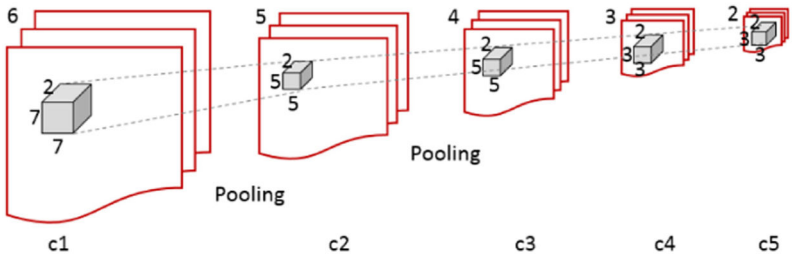


Fig. 7.
T-Net of TSF-CNN.

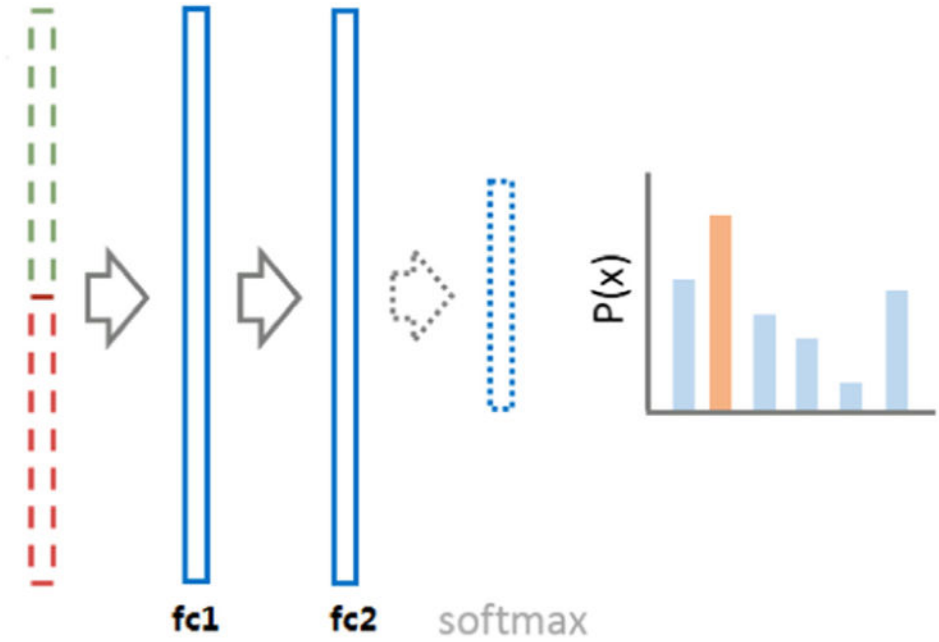


Fig. 8.
F-Net of TSF-CNN.

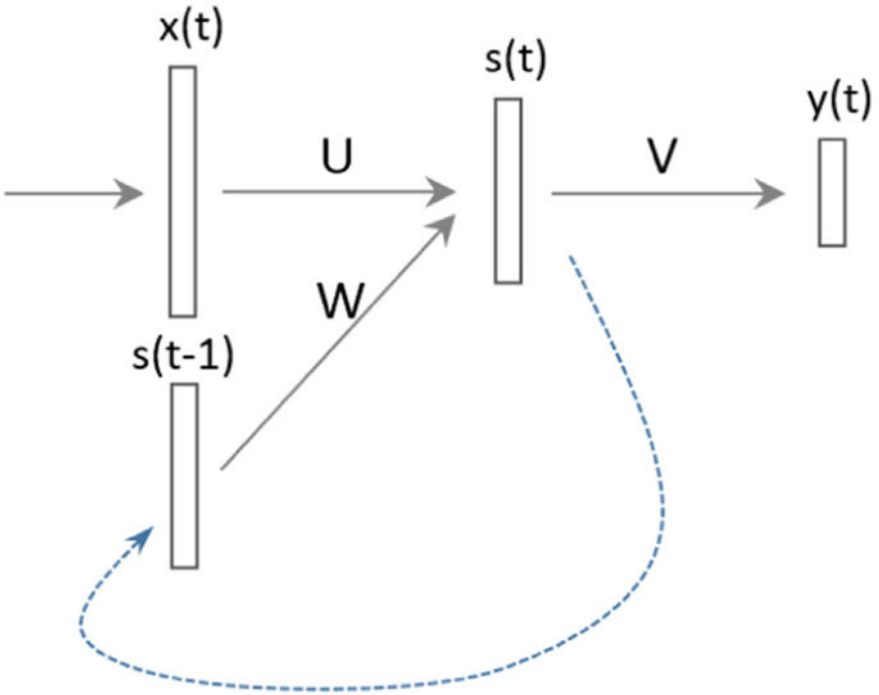


Fig. 9.
A standard recurrent network. Parameters (*U, W* and *V*) are shared for all time steps.

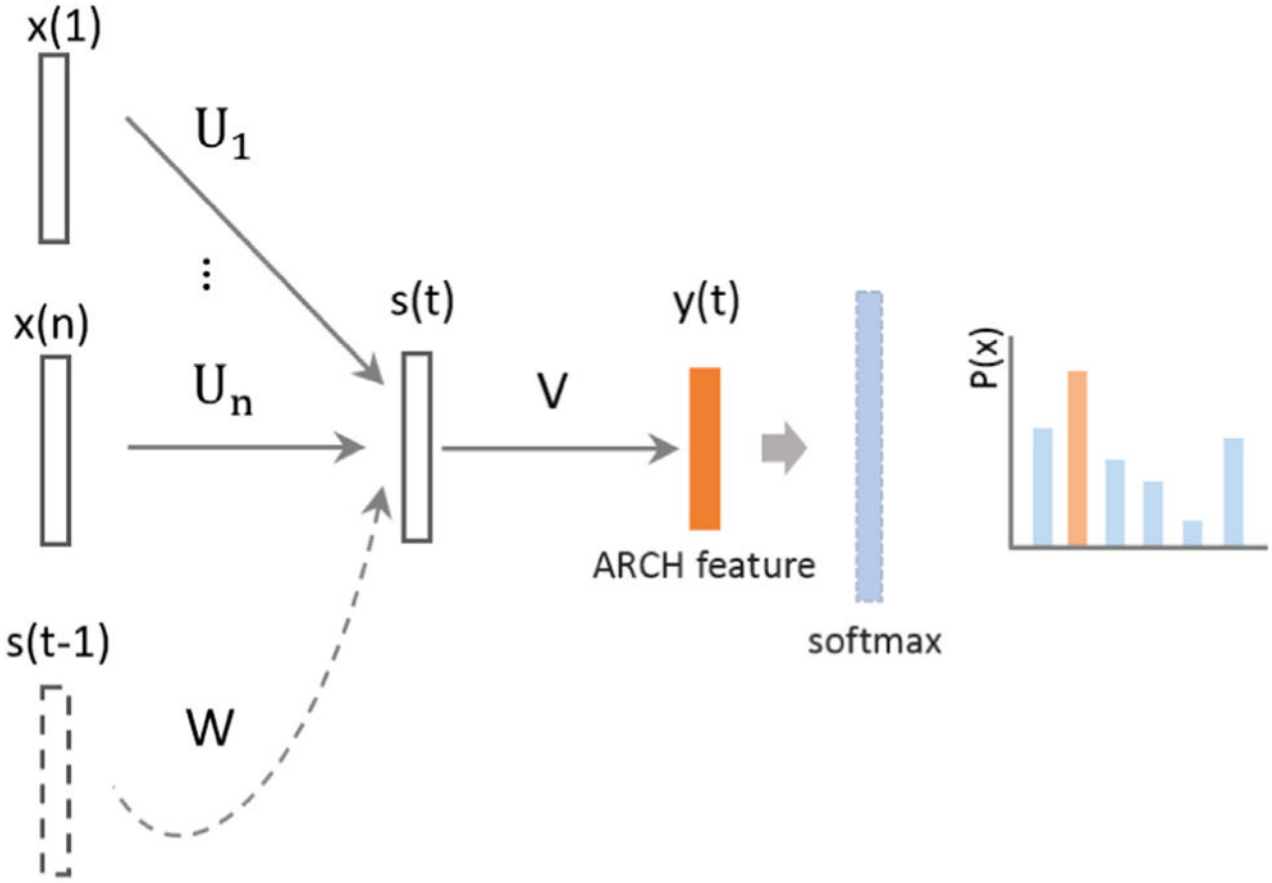


Fig. 10.
The memory-weights-independent RNN. U_t is non-shared with others.

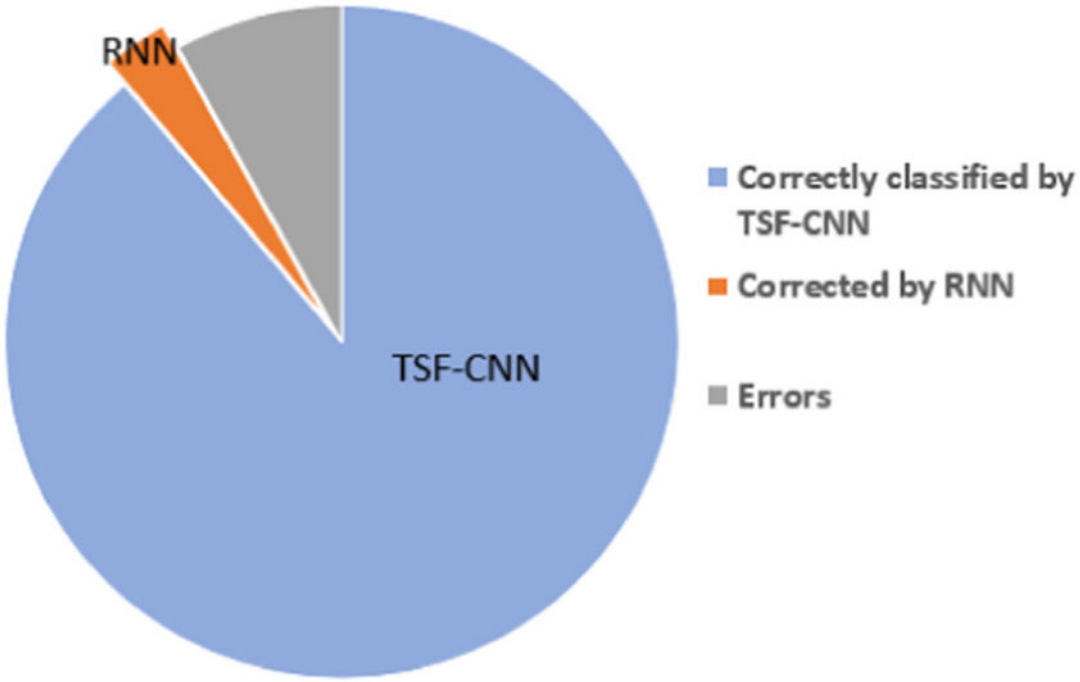


Fig. 11.

TSF-CNN and RNN's contributions on recognitor accuracy. This figure is best viewed in color. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

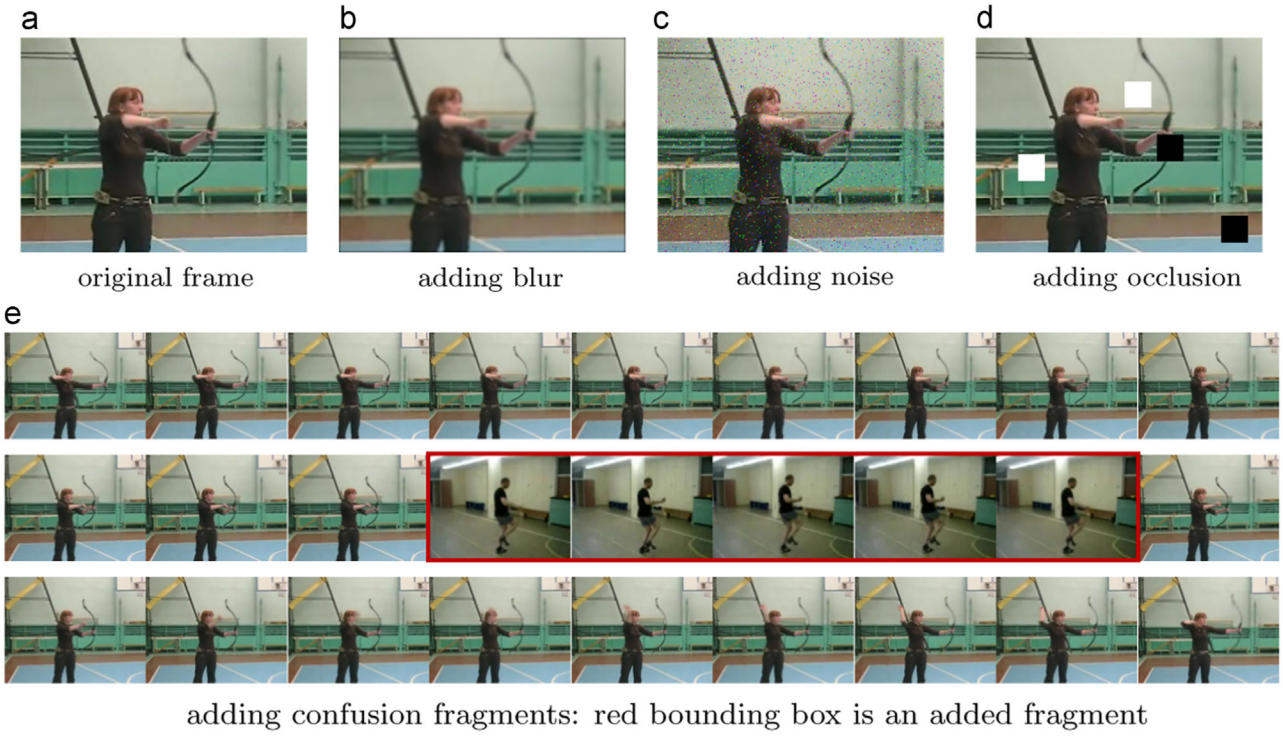


Fig. 12.

An example of data augmentation: (a) original frame; (b) blurred frame; (c) noised frame; (d) adding random occlusion; (e) adding confusion fragments: red bounding box is an added fragment. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

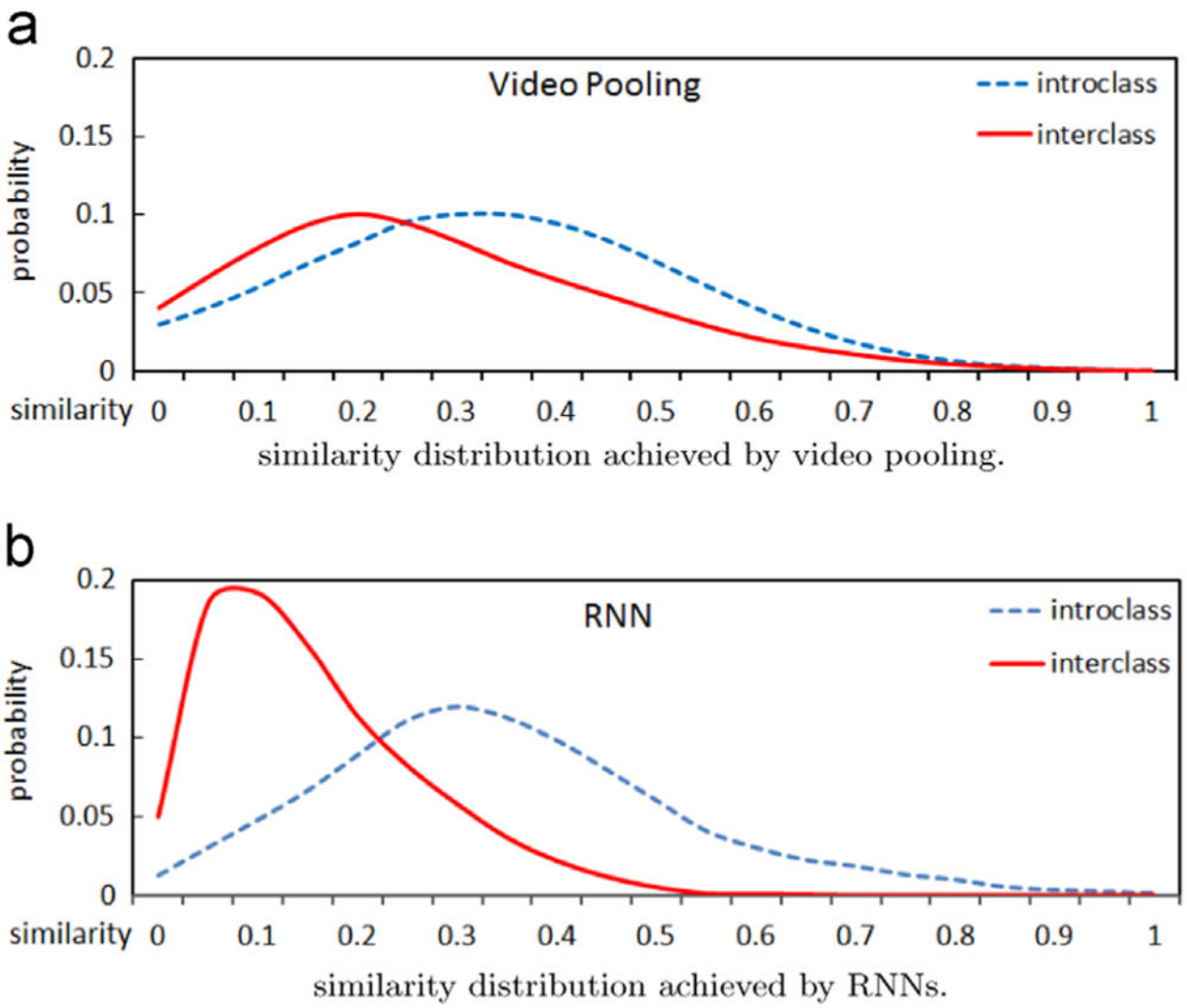


Fig. 13.

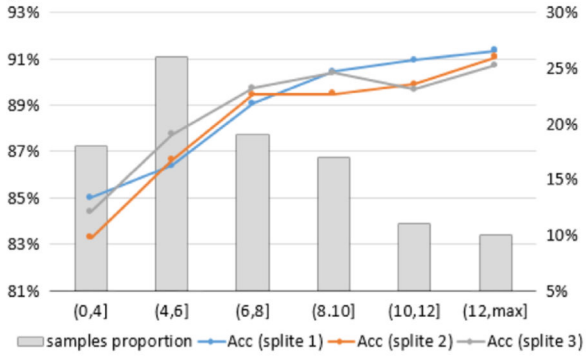
Cosine similarities achieved by the video pooling method and the RNN. Given two feature vectors, similarity=1 means they are exactly the same, while 0 indicates decorrelation. This figure is best viewed in color: (a) similarity distribution achieved by video pooling; (b) similarity distribution achieved by RNNs. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

a

Length (s)	Samples	Proportion	Acc
(0,4]	700	18.5%	85.0%
(4,6]	1006	26.6%	86.3%
(6,8]	704	18.6%	89.0%
(8,10]	617	16.3%	90.4%
(10,12]	397	10.5%	90.9%
(12,Max]	359	9.5%	91.3%

Test result on split 1.

b



Statistics over three splits.

Fig. 14.

Recognition results w.r.t. different lengths of videos in UCF101. (a) lists results on split 1, including numbers and proportions of samples in each range of lengths, and corresponding recognition accuracies. (b) Line chart demonstrates recognition results on three splits. Histogram is samples' quantitative distribution across all splits.

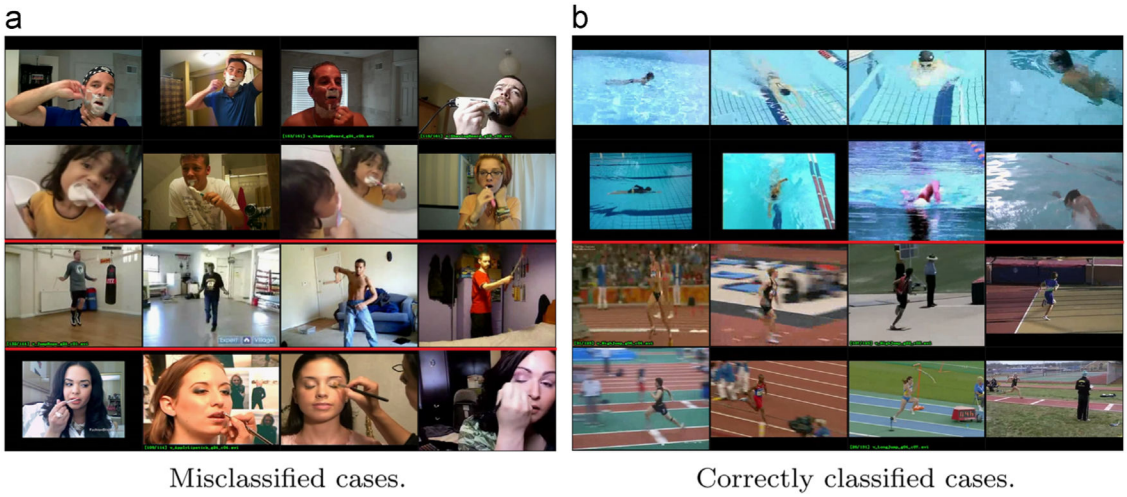


Fig. 15.

(a) lists some misclassification cases. Row 1 is “Shaving Beard” actions and row 2 is “Brush Teeth”. Senses and motions are both quite similar. Row 3 is some short-term and fast repeated actions such as “JumpRope” and “Nunchucks”. Row 4 is several actions that perform on small regions, such as “Apply Eye Makeup” and “Apply Lipstick”. (b) lists some hard but correctly classified cases, e.g., “Breaststroke” vs. “Front Craw” (row 1, row 2), and “High Jump” vs. “Javelin” (row 3, row 4).

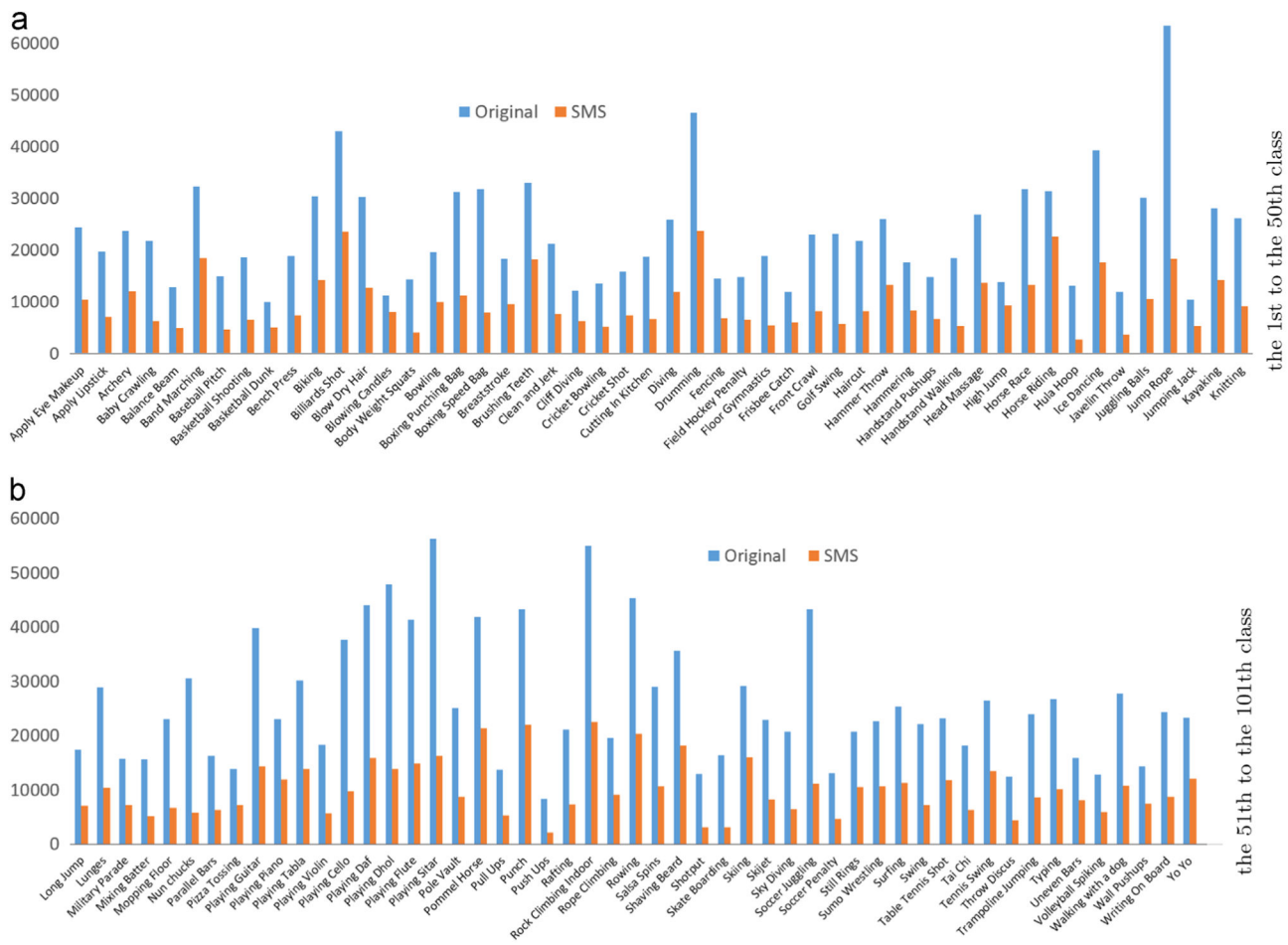


Fig. 16.

Comparisons on numbers of frames between original video and SMS on 101 classes: (a) the 1st to the 50th class, (b) the 51th to the 101th class.

Fig. 17.

S-NET features. The first row is the RGB channels of a SMS. The second row is their corresponding feature maps derived from the first volume convolutional layer.

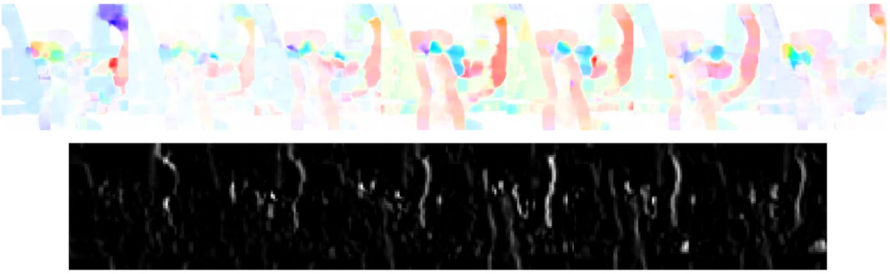


Fig. 18.

T-NET features. The first row is the XY channels of a SMS (visualized in Munsell color space). The second row is their corresponding feature maps derived from the first cube convolutional layer. Temporal stride is 2. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Table 1

The detailed configurations of the TSF-CNN.

	L1	L2	L3	L4	L5	L6	L7
S-Net	c[7*7]	c[5*5]	c[5*5]	c[5*5]	c[3*3]		
	p[2*2]	p[2*2]	a[relu]	a[relu]	a[relu]		
	a[relu]	a[relu]					
	n[lrn]						
T-Net	c[7*7*2]	c[5*5*2]	c[5*5*2]	c[3*3*2]	c[3*3*2]		
	p[2*2]	p[2*2]	a[relu]	a[relu]	a[relu]		
	a[relu]	a[relu]					
F-Net						fc(dp)	fc(dp)

Neurocomputing. Author manuscript; available in PMC 2017 December 29.

Table 2

TSF-CNN recognition accuracy on UCF101 and HMDB51.

Method	UCF101 (%)	Method	HMDB51 (%)
STIP+BoV [10]	44.5	ω-Flow+VLAD [60]	52.1
SFV [61]	–	SFV [61]	54.8
MVSV [62]	83.5	MVSV [62]	55.9
Slow Fusion CNN [11]	65.4	DT [63]	46.6
C3D+fc6 [12]	76.4	iDT [8]	57.2
Spatial ConvNet [14]	73.0	Two-stream ConvNet [14]	**59.4**
VGGA [64]	68.4	Composite-LSTM [13]	44.0
TSF-CNN (from scratch)	79.6	TSF-CNN (from scratch)	51.7
TSF-CNN + pre-training	**85.3**	TSF-CNN + pre-training	58.2

Table 3

Experimental result testing on ARCH.

Methods	MAcc (%)
TSF-CNN (5L) without T-Net	58.5
TSF-CNN (5L) without S-Net	69.7
TSF-CNN	**85.3**

Table 4

Comparison with state-of-the-art action recognition models.

Categories	Methods	MAcc (UCF101) (%)
Hand-crafted	iDT+FV [8]	87.9
	MVSV [62]	83.5
	Slow Fusion CNN [11]	65.4
	C3D+fc6 [12]	76.4
Learned feature	Two-stream ConvNet [14]	87.6
	LRCN [29]	82.9
	Composite-LSTM [13]	84.3
	ARCH	**88.2**

Table 5

Comparison with state-of-the-art action recognition models.

KTH		**Hollywood2**		**HMDB51**	
DT [63]	89.8%	DT [63]	51.9%	iDT+FV [8]	57.2%
convGRBM [31]	90.0%	MVSV [62]	62.5%	MVSV [62]	55.9%
3D-CNN [30]	90.2%	SFV [61]	**63.3**%	Two-stream CNN [14]	59.4%
CNN+LSTM [37]	94.3%	VVG [65]	59.6%	Composite-LSTM [13]	44.0%
ARCH	**95.2**%	**ARCH**	63.1%	**ARCH**	**61.1**%

Table 6

Experimental result testing on ARCH.

Training methods	MAcc (UCF101) (%)
Training on UCF101 from scratch	82.6
Training on UCF101 + data augmentation	83.5
Training on UCF101 + pre-training + data augmentation	**88.2**

Table 7

Experimental results using different preprocessing methods.

Preprocessing methods	MAcc (UCF101) (%)
No alignment	77.1
DTW	80.7
GTW	81.5
SMS	82.6