Temporal Sparse Feature Auto-combination Deep Network for Video Action Recognition[y]

Qicong Wang*[1] | Dingxi Gong[1] | Man Qi[2] | Yehu Shen[3] | Yunqi Lei[1]

[1]Department of Computer Science, Xiamen University, Fujian, China
[2]School of Law, Criminal Justice and Computing, Canterbury Christ Church University, Canterbury, UK
[3]College of Mechanical Engineering, Suzhou University of Science and Technology, Jiangsu, China

Correspondence

*Qicong Wang, Department of Computer Science, Xiamen University, Xiamen 361005, Fujian, China. Email: qcwang@xmu.edu.cn

Summary

In order to deal with action recognition for large-scale video data, we present a spatio-temporal auto-combination deep network which is able to extract deep features from short video segments by making full use of temporal contextual correlation of corresponding pixels among successive video frames. Based on conventional sparse encoding, we further consider the representative features in adjacent nodes of the hidden layers according to activation states similarities. A sparse auto-combination strategy is applied to multiple input maps in each convolution stage. An infor-mation constraint of the representative features of hidden layer nodes is imposed to handle the adaptive sparse encoding of the topology. As a result, the learned features can represent the spatio-temporal transition relationships better and the number of hidden nodes can be restricted to a certain range. We conduct a series of experiments on two public data sets. The experimental results show that our approach is more effective and robust in video action recognition compared with traditional methods.

KEYWORDS:
action recognition, deep learning, spatio-temporal convolution, feature map, sparsity

1 INTRODUCTION

Most of the existing video based human action recognition approaches use handcrafted features. An important and difficult problem with above mentioned methods is how to design more effective features to represent the actions. As early as 1973, Johansson et al. [1] studied how to perceive the action from the perspective of neuroscience. Some small light sources were tied to main joints of a body. Eventually, the human actions could be inferred just from the motion of those bright spots. Later many approaches used similar technologies to model human body for action recognition [2,3].

Rao et al. [4] thought that human actions could be expressed by the rapid changes in direction and speed of movement trajectory. Therefore they described the actions through spatio-temperol changes of the trajectory. Their method was view-invariant for action recognition. For a single static image, Mori et al. [5] modeled a human body with geometry shapes. So their action classification method was based on contextual shapes. Ramanan et al. [6] applied some rectangular blocks to model a human body. Although these modeling methods are simple and intuitive, only rough shape features can be extracted and the recognition result is not satisfactory.

Optical flow is presented by Gibson [7,8] to describe object motion in images. As a classic motion feature, it can be computed according to pixel changes caused by object movement. Bobick et al. [9] proposed the motion history image (MHI) and motion energy image (MEI) based on the contours of the human body. They can extract the shape and motion information of a body directly from video. Chen et al. [10] used several points obtained with the optimal clustering of convex contours to represent a human body. These points are connected as a star whose changes can describe the human action. Scovanner et al. [11] extended the SIFT feature from 2D image to 3D video and proposed a 3D-SIFT descriptor that can extract

~~[y]This is an example for title footnote.~~

[0]**Abbreviations:** ANA, anti-nuclear antibodies; APC, antigen-presenting cells; IRF, interferon regulatory factor

spatio-temporal features from video at the same time. It performs better than many traditional features when combined with the bag-of-words model. Wang et al. [12] tracked dense sampled points in video to form dense trajectories which can also represent the human action effectively. Li et al. [13] sampled a small amount of points from 3D images to represent the body poses. Furthermore, these poses are viewed as points of a motion graph model. In order to improve the robustness of video action recognition, Sun et al. [14] used SIFT features and Zernike moments to extract the local and global features from video frames respectively.

Unfortunately, the handcrafted features have the following disadvantages. Most of them are only effective on some particular applications or data. Generally speaking, they have a certain degree of one-sidedness, blindness, and poor transferability. The process of feature extraction is computationally demanding. In machine learning, some methods using automatic feature learning have gained better performances for different tasks [15] [16]. These feature learning methods have good generalization ability. Without any prior knowledge, they can learn discriminative features automatically from raw data. As a kind of automatic feature learning method, convolutional neural network has been widely applied in image classification [22] [23]. Recently, it has attracted more and more attention in video action recognition [17] [18] . In convolutional neural network, raw images or feature maps are convolved using learnable kernels in convolution layers and their weights can be adjusted by error back propagation. As a result, the classification features can be extracted automatically. The network model can be directly applied to raw images which eliminates the complexity and blindness of traditional handcrafted features. However, unlike single image recognition, we can extract more discriminative features from serval successive frames in video action recognition. To this end we should make full use of temporal contextual relationships of corresponding pixels among successive video frames.

In convolutional neural network, we can compute an output feature map by summing up all convolution values after convolving with multiple input feature maps. In previous studies [17] [18], some feature maps are selected manually to generate a feature map, which lacks flexibility and is not beneficial to extract the essential features of human action from video. Inspired by the sparse auto-encoder algorithm, we introduce sparse constraints into the convolution layers and propose an automatic combination strategy for input feature maps. We aim to learn these optimal combinations in the training process of the convolutional neural network. It can make feature representation of video segments which reflect spatio-temporal intrinsic change law and correlation more effectively.

2 2D CONVOLUTION AND SPATIO-TEMPORAL CONVOLUTION

2.1 Convolutional Neural Network

Convolutional neural network is a special neural network that can be applied directly to two-dimensional maps. Therefore it is also called 2D convolutional neural network borrowing the concept of the receptive field of visual nervous system. The main difference from the traditional neural network is that input neurons can be arranged to form a two-dimensional map which exists relative spatial relationships among neurons. This kind of two-dimensional map is called a feature map in convolutional neural network. Input neurons are not fully connected to the output neurons, i.e. each output neuron is only connected to some local input neurons. This local domain is the receptive field of the neuron. Figure 1 shows the special connection of convolutional neural network. Because the features extracted by convolution are local and they can be organized as two-dimensional maps, it provides more complex features for the next step and local spatial topological structures of image features are preserved effectively.

In traditional neural network, the number of parameters is equal to the number of connections. However, convolutional neural network adopts the strategy of sharing weights so that the number of parameters is not necessarily related to the number of connections. This greatly reduces the training parameters and difficulties. The so-called sharing weights means that the same convolution kernels are used when all neurons in the same feature map are convolved with input neurons.

In visual neural system, specific neuron responds to specific stimuli. For example, the direction-selective neuron only responds to a specific light direction which is consistent with the sharing weight strategy. The neurons of the same feature map can only extract the same kind of features from different positions because they share one convolution kernel. This means they can only be activated by a certain feature. It is not enough to extract only one class of feature when we classify the images. Therefore, in convolutional neural network, the neurons of the same local domain are always connected to different feature maps. It means that different features can be extracted by convolving with different convolution kernels.

As shown in Fig. 1, the feature maps 1 and 2 use different convolution kernels, so they extract two different features. The dotted line and solid line represent different convolution kernels in this figure. From the view of feature map 1, it means that one kind of features are extracted from local domain 1 and local domain 2 using the same convolution kernel and then they are put in different neurons of the same feature map. From the view of the local area 1, it means that two kind of features are extracted using different convolution kernels and they are put in the neurons of different feature maps.

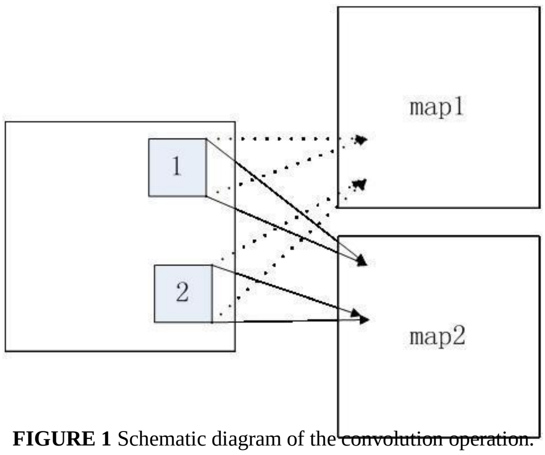


FIGURE 1 Schematic diagram of the convolution operation.

2.2 Spatio-temporal Convolution

In convolutional neural network, convolution is mainly performed for generating feature maps which only contain spatial information. For video analysis, we need to extract temporal motion information from multiple successive video frames [24] . Therefore, the convolution phase of convolutional neural network should perform spatial and temporal convolution simultaneously and get spatio-temporal features.

In the convolution layers of spatio-temporal convolutional neural network, different kernels are convolved with multiple feature maps from the previous layer so that we can obtain multiple different output feature maps. The jth feature map of the tth layer can be expressed as the following formula.

$$X_{jt} = f\Big(\sum_{i \in M_j} X_{it1} \ W_{ijt} + b_{jt} \Big) \tag{1}$$

where M $_j$ represents a combination of some feature maps selected from all the feature maps according to a preset strategy. In order to generate different feature maps, it is necessary to construct different input combinations which are convolved with different learnable convolution kernels. When we construct the combinations of the input feature maps, different strategies can be adopted according to data characteristics. Sometimes random selection can also be used. Meanwhile all input feature maps should be utilized effectively. Table 1 shows the combination strategy of LeNet-5 [27]. From the table we can conclude that different combinations are chosen from six input feature maps to generate sixteen feature maps. In this paper, we will explore a novel method for automatic learning of input combinations for video action recognition.

3 IMPLEMENTATION OF SPATIO-TEMPORAL CONVOLUTIONAL NEURAL NETWORK

3.1 The Convolution Layer

Forward propagation of spatio-temporal convolutional layer is formulated in Eq. 1. Backward propagation uses the steepest descent method to guide the training of neural networks through error back propagation. Assume that the output of spatio-temporal convolutional neural network is a_L for one sample, and the error $_L$ of the output layer can be computed as:

$$^{L} = (y \qquad a^{L}) \ f'(z^{L}) \tag{2}$$

$$^{L} = (y \qquad a^{L}) \ a^{L} \ (1 \quad a^{L}) \tag{3}$$

For a multi-class problem, y and a_L are vectors. For a c-class classification problem, y uses one-hot encoding where the value of the corresponding class of input sample is set to 1 and others are set to 0. The output value of a_L is between 0 and 1, so the goal of training is to minimize the error between y and a_L.

Each convolution layer C_l is followed by a pooling layer S_{l+1} . In order to compute the updated values $\triangle W_l$ of the corresponding weights W_l which belong to the neurons of the lth layer, the sensitivity $_l$ of the lth layer could be solved by the following formula.

$$\nabla_{W^{l}} J(W; b) = {}^{l+1} \ (a^{l})^{T} \tag{4}$$

In order to obtain the sensitivity $_l$, we need to calculate $_{l+1}$. Then we multiply it with the corresponding weights W_{l+1} of the connection between the layer l and l + 1. Finally we multiply it with the derivative f′(z) of the activation function f for the input z of the lth layer to compute layer l and get $_l$ by:

$$ {}^{l} = \left((W^{l})^{T}\ {}^{l+1} \right) f'(z^{l}) \tag{5}$$

The weights of each layer are updated using the method of backward propagation. In spatio-temporal convolutional neural network, due to the existence of the pooling layer, i.e. the convolution layer C_l is followed by a subsampling layer S_{l+1} , the sensitivity of a neuron of the pooling layer S_{l+1} corresponds to a block in a feature map of the convolution layer C_l . In order to calculate the residual $_l$ for the jth feature map of the C_l layer, we need to up-sample $_j^{l+1}$ and make the size of $_j^{l+1}$ consistent with the size of feature map of C_l . Then ^{j}we are able to deal with matrix operations. The jth feature map of the l layer is calculated by:

$$ {}_{j}^{l} = {}_{j}^{l+1} \left(f'(z_{j}^{l})\ \ up({}_{j}^{l+1}) \right) \tag{6}$$

where up( ) is up-sampling operation. Suppose that the down-sampling factor is n, it copies a pixel n n times to form a matrix which restores the size of feature map before sub-sampling. The up-sampling operation can be defined as a Kronecker product:

$$ up(x)\quad x\ \ 1_{n\ n} \tag{7}$$

In order to compute $_j^l$, we perform element-wise multiplication between the residual up($_j^{l+1}$) obtained from the up-sampling and the derivative $f'(z_j^l)$ of the activation function f for the input z_j^l of the jth sub-sampling feature map of the layer l + 1 and multiply the weights W_j^l between C_l and S_{l+1}. Because in the down-sampling step, all the weights of the same feature map are the same, i.e. W_j^l = . For each feature map of the convolution layer, we use the above calculation process to get the residual $_l$ of all feature maps in the convolution layer C_l. For the jth feature map, we can calculate the gradient ∇b_l of the bias term b_l through summing up the residuals of all the neurons in each feature map of the C_l layer.

So the gradient ∇b_j^l is computed by: $\quad{}^{j}\qquad\qquad{}^{j}$

$$ \nabla b_{j}^{l} = \sum_{u;v} ({}_{j}^{l})_{uv} \tag{8}$$

According to Eq. 4, through multiplying the corresponding activation values of the previous layer S_{l1} , we can get the gradient ∇W_j^l of weights of the jth feature map in the C_l convolution layer. Due to the reason that weights are sharing values, for a given weight W_j^l, many map blocks are connecting with it. During the calculation process, we need to solve the gradient of all the elements connected with the weight. We use the above method which can calculate the gradient of the bias term to sum up the gradients of all elements. The gradient of W_{ij}^l can be expressed as:

$$ \nabla W_{ij}^{l} = \sum_{u;v} ({}_{j}^{l})_{uv} (p_{i}^{l1})_{uv} \tag{9}$$

where $(p_i^{l1})_{uv}$ are map blocks of a_i^{l1} multiplied by W_{ij} element-wise in convolution.

3.2 The Pooling Layer

The pooling layer aims to scale feature maps and their main characteristics can be preserved when the resolution of the feature maps is reduced. It is similar to a fuzzy filter which can enhance invariability for image scaling, translation and deformation while reducing the complexity of convolutional neural network. For a local sub-block x, output neuron y can be computed by:

$$ y = f(w\sum_{x} x_{i} + b) \tag{10}$$

where x_i are all neurons which belong to x. In order to calculate y by the pooling process, first all neurons belonging to x are summed up and multiplied by a parameter w and then add a bias term b into the activation function. According to the weight sharing strategy, the sampling parameters w and b of any sub-block of the feature map are the same when sub-sampling a feature map.

For the pooling layer, the number of output feature maps is N when there are N input feature maps. Each output feature map becomes smaller.

The jth feature map X_j^l of the pooling layer S_l can be calculated by:

$$ X_{j}^{l} = f\left({}_{j}^{l}\, down(X_{j}^{l1}) + b_{j}^{l} \right) \tag{11}$$

where down( ) is down-sampling step, is a weight coefficient and b is a bias term.

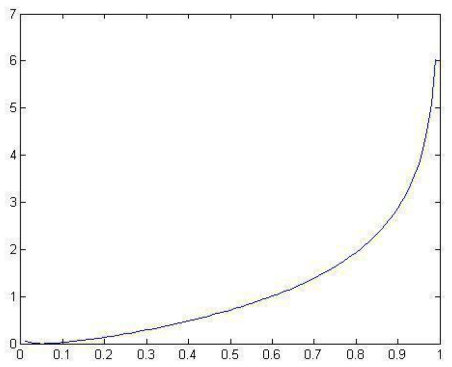


FIGURE 2 Sample curve of a relative entropy function.

For backward propagation of the pooling layer, first we need to determine the corresponding relationships of connection between the pooling layer S_l and the following convolution layer which can propagate back the residual $^{l+1}$ of the following layer. The calculation process of the bias term b is similar to the convolution layer. We sum up all the elements of the residual $_j^l$ by:

$$\nabla b_j = \sum_{u,v} (_j^l)_{uv} \tag{12}$$

For the weight parameter , first we define a down-sampling function down( ) as:

$$d_j^l = down(X_j^{l1}) \tag{13}$$

Thus we can compute the gradient of by:

$$\nabla_j = \sum (_j^l d_j^l)_{uv} \tag{14}$$

4 TEMPORAL SPARSE FEATURE AUTO-COMBINATION

In order to learn some more essential features from x for a given data set x = x1; x2; x3; x4; x5, we design an auto-combination network containing three layers which are an input layer, a hidden layer, and a reconstruction layer respectively. During training process, we try to minimize the error between the input layer and the output of the reconstruction layer. To this end we can mine the essential information of input data from the hidden layer which could be viewed as an expression of the input data.

In fact, the above structure is to learn a function $h_{W;b}$ x, which can always unveil some intrinsic characteristics from data by limiting the number of neurons of the hidden layer. This is analogous to a dimensionality reduction method under the circumstances where there exists fewer neurons in the hidden layer. If the number of neurons in the hidden layer is relatively large, more essential characteristics can also be mined from data by imposing a sparse constraint. Assume that the jth neuron belongs to the hidden layer and its activation value is a_j, we can make the structure sparse by the following equation:

$$_j = \frac{1}{m}\sum_{i=1}^{m} a_j x_i \tag{15}$$

where m is the number of neurons in the input layer, and $_j$ is a sparse parameter determined by the right item of Eq. 15. It is not a variable. $_j$ should be close to a preset sparsity constant . When we design the hidden layer, $_j$ can be optimized by computing the following relative entropy:

$$KL(\,/\!/\,_j) = \log\frac{}{_j} + (1 -)\log\frac{1-}{1-_j} \tag{16}$$

In Fig. 2, we show a relative entropy function. The minimum value can be obtained when $_j$ is equal to .

In the convolution layer, spatio-temporal convolution kernels are used to specify the input of each feature map. This method can be implemented simply without additional calculation. However, due to the way of presetting, self-learning ability of the network is limited which is not beneficial to mine the most essential information. To effectively enhance the learning ability of the network, a novel sparse auto-combination strategy is proposed so that each convolution layer can use the learned combinations of feature maps as its input.

To this end, we introduce a sparse constraint parameter a_{ij} into Eq. 1. It represents the weight or the contribution of the ith input feature map when we calculate the jth output feature map. Thus it can be computed by:

$$X_j^l = f\Big(\quad {}^{N}{}_{in} \qquad W_{ij}^{l}) + b_j^{l} \qquad \Big) \tag{17}$$

Meanwhile, a_{ij} must comply with the following constraints: $\sum$

$$\sum_{i} a_{ij} = 1; \; and \; 0 \quad a_{ij} \quad 1 \tag{18}$$

In sparse auto-encoding neural network, sparse constraints are imposed on the output. So an over-complete basis is used to represent the input data which is equivalent to extracting the low-level features of the input data and then reproducing the input data. However, in order to realize the compact expression of input data, namely extracting their high-level features, we place the constraints to the input. So only a small amount of inputs are used to activate a neuron of the output layer.

If a_{ij} is viewed as a dependent variable, it can be further expressed by a set of corresponding independent variable c_{ij}, i.e. the following softmax function of c_{ij}.

$$a_{ij} = \frac{\exp(c_{ij})}{\sum_k \exp(c_{kj})} \tag{19}$$

For a fixed feature map ^{j}, its corresponding sparse coefficients c_{ij} are independent of the other feature maps. For convenience, we remove the subscript j, i.e. only consider the update of a feature map. The updates of other feature maps are the same.

According to Eq. 20, the partial derivative of a_k with respect to c_i is computed by:

$$\frac{@c_i}{@a_k} = {}_{ki} a_i \; a_i a_k \tag{20}$$

where $_{ki}$ is the Kronecker Delta Function [25]. The independent variables of the Kronecker Delta Function are two integers, namely k and i. If they are equal, then the output is 1, otherwise the output is 0. The partial derivative of error cost function $J(W; b)$ with respect to the sparse coefficient a_i of the jth feature map of the lth layer is calculated by:

$$\frac{@a_i}{@J(W;b)} = \frac{@z^l}{@J(W;b)\,@\,z} \frac{@a_i^l}{} \sum \Big({}^{l}(X_i^{l1} \qquad W_i^l)\Big)_{uv} \tag{21}$$

According to the derivation rules of compound function, the partial derivative of error cost function $J(W; b)$ with respect to c_i is computed by:

$$\frac{@c_i}{@J(W;b)} = \sum_k \frac{@a_k}{@J(W;b)\,@a_k} \frac{@c_i}{} = a_i \Big(\frac{@a_i}{@J(W;b)} \sum_k \frac{@a_k}{@J(W;b)} a_k \Big) \tag{22}$$

A constraint term $!(\,)$ of sparse coefficients is imposed on the weight attenuation term of error cost function. The error cost function can be rewritten as:

$$J(W\,;b) = J(W;b) + \sum_{i;j} j(a)_{ij}j \tag{23}$$

where the second item is $!(\,)$. f

The partial derivative of $!(\,)$ with respect to a_i is calculated by:

$$\frac{@a}{@!} = sign(a_i) \tag{24}$$

According to the derivation rules of compound function, the derivative of $!(\,)$ to c_i is calculated by:

$$\frac{@!}{@c_i} = \sum_k \frac{@!}{@a_k} \frac{@a_k}{@c_i} = \Big(ja_ij \quad a_i \sum_k ja_kj \Big) \tag{25}$$

Combining Eq. 22 and Eq. 25, we can compute the gradient of the modified error cost function $J(W; b)$ e with respect to c_i by the following formula:

$$\frac{@J(W\,;b)}{@c} = \frac{@J(W;b)}{@c} + \frac{@!}{} \tag{26}$$

$$f_i \qquad i \qquad @c_i$$

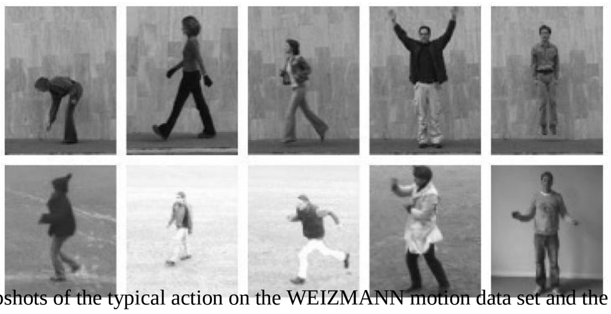


FIGURE 3 The video snapshots of the typical action on the WEIZMANN motion data set and the KTH motion data set.

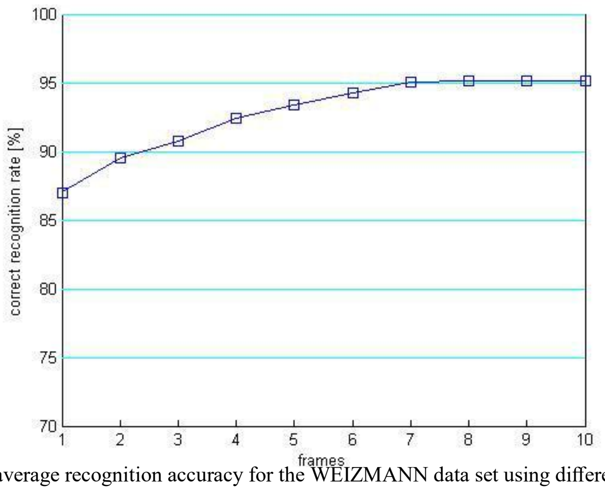


FIGURE 4 The average recognition accuracy for the WEIZMANN data set using different frame lengths.

5 EXPERIMENTAL RESULTS AND ANALYSIS

For video action recognition, an important parameter of spatio-temporal convolutional neural network is the size of temporal dimension which can determine the optimal number of successive frames to recognize video actions.

In this paper, we will use one frame or several frames for action recognition. Experimental data set are long videos, so we split each long video into some short video segments first. Under the extreme circumstance, a video segment may only has one frame.

Generally speaking, if the input frames have different length, we need to configure different frameworks for spatio-temproal convolutional neural network. In order to compare the impacts induced by different input length, the differences among all network frameworks are only between the input layer and the C1 layer. The configuration of remaining layers are unchanged. In analogy to this, no matter how many frames the input layer takes, only the size of the convolution kernel is different.

We use the WEIZMANN data set in the experiment. It is the standard data set for video action recognition [28] . This data set includes nine actions, namely bending, jumping jack (jack), jumping, jumping in place (pjump), gallop sideways (side), running, walking, one-hand waving (wave1) and two-hands waving (wave2). To collect the data set, each action was performed once by nine subjects under the same static background respectively. It contains nine categories and each category has nine samples. The videos in the WEIZMANN data set are some periodic actions. In the experiments, we take several successive frames starting from the beginning frame of the video as a sample. Because the number of the collected samples are small. For all experimental evaluations we use a leave-one-out cross validation method which takes one sample for testing and the other samples for training. According to different data combinations, the experiment was repeated 81 times and we calculate the mean for all the results.

In Fig. 4, it shows the average recognition accuracy for different input frame lengths. The average training time is 97854ms. We can see that the recognition accuracy can reach 87.25% when taking one frame. As the number of frames increases, the recognition accuracy rises almost linearly. When the number of frames reaches seven or more, accuracy becomes saturate. It means that the impact of the number of frames on recognition accuracy becomes less significant.

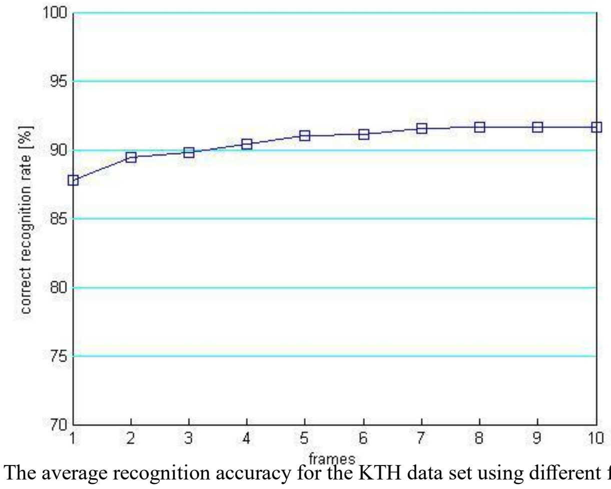


FIGURE 5 The average recognition accuracy for the KTH data set using different frame lengths.

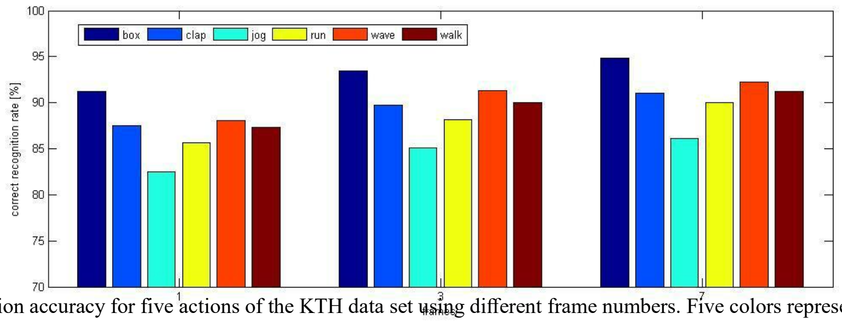


FIGURE 6 The recognition accuracy for five actions of the KTH data set using different frame numbers. Five colors represent five actions respectively.

The KTH data set contains six actions, which are boxing, clapping, jogging, running, waving and walking [29]. Each action was performed once by twenty-five subjects in four different scenarios respectively. The four scenarios include outdoors (s1), outdoors with scale variation (s2), outdoors with different clothes (s3), and indoor (s4). In the videos of jogging, running and walking, each action was performed several times. The subjects are out of the video completely in several intervals, thus we preprocess these videos and cut out the video segment of action executed once. For the KTH data, all experiments are evaluated using 5 fold cross-validation method. Five experiments are performed according to different combinations. Final experimental result is the average of the five results. The KTH dataset can be seen as either a large data set that has strongly individual changes or four different data sets according to different scenarios. So we carry out different experiments in two cases.

Figure 5 shows the average recognition accuracy for the KTH data set using different frame lengths. The average training time is 793412ms. We can see that the experimental results are similar to those of the WEIZMANN data set.

If the overall average recognition accuracy in Fig. 5 is decomposed into the recognition accuracy of each action, the experimental results are shown in Fig. 6. In order to demonstrate clearly, the recognition accuracy for each action is in the case of one frame, three frames, and seven frames. We can see that the recognition accuracy of each action has a growing tendency as the frame number increases. It is the same with the overall average recognition accuracy. In the case of the same number of frames, the recognition accuracy of jogging is the lowest and that of boxing is the highest. This means recognizing jogging is more difficult than recognizing boxing. Note that we mainly consider the overall recognition accuracy when determining the length of video segments for action recognition, i.e. it does not depend on the recognition accuracy of some specific action.

If figure 5 is converted to recognition results in four different scenarios, they are shown as Fig. 7. The results of outdoor scenarios (s1,s2,s3) are worse than those of indoor scenario (s4). The reason is that the illumination of outdoor scenarios often changes. The results of the s2 scenario is the worst. This is because s2 uses zoom lens which leads to larger viewpoint changes.

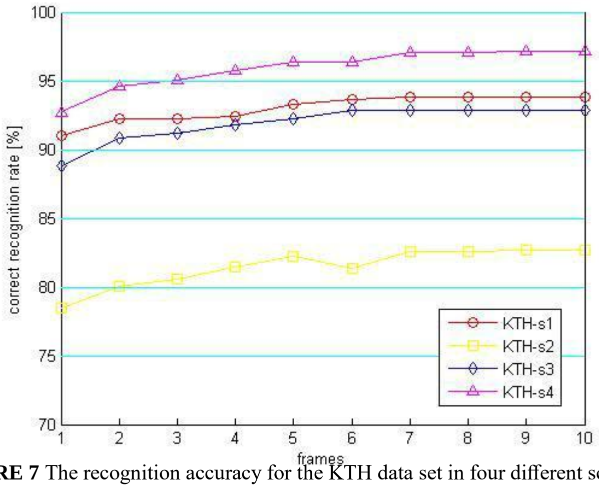
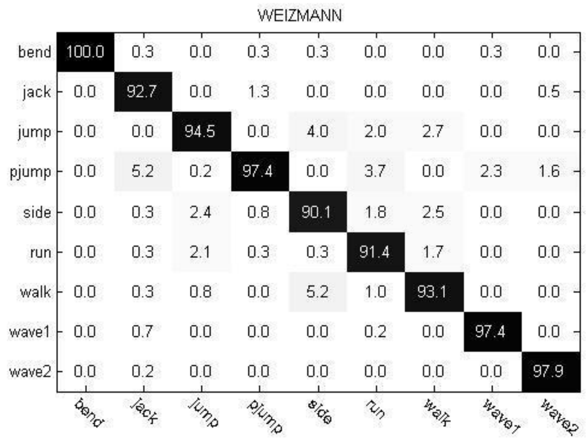


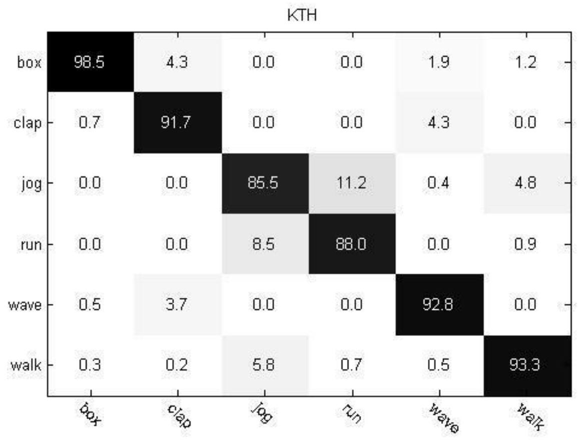
FIGURE 7 The recognition accuracy for the KTH data set in four different scenarios.



WEIZMANN

	bend	jack	jump	pjump	side	run	walk	wave1	wave2
bend	100.0	0.3	0.0	0.3	0.3	0.0	0.0	0.3	0.0
jack	0.0	92.7	0.0	1.3	0.0	0.0	0.0	0.0	0.5
jump	0.0	0.0	94.5	0.0	4.0	2.0	2.7	0.0	0.0
pjump	0.0	5.2	0.2	97.4	0.0	3.7	0.0	2.3	1.6
side	0.0	0.3	2.4	0.8	90.1	1.8	2.5	0.0	0.0
run	0.0	0.3	2.1	0.3	0.3	91.4	1.7	0.0	0.0
walk	0.0	0.3	0.8	0.0	5.2	1.0	93.1	0.0	0.0
wave1	0.0	0.7	0.0	0.0	0.0	0.2	0.0	97.4	0.0
wave2	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	97.9

FIGURE 8 The confusion matrix of each action class using seven frames for the WEIZMANN data set.



KTH

	box	clap	jog	run	wave	walk
box	98.5	4.3	0.0	0.0	1.9	1.2
clap	0.7	91.7	0.0	0.0	4.3	0.0
jog	0.0	0.0	85.5	11.2	0.4	4.8
run	0.0	0.0	8.5	88.0	0.0	0.9
wave	0.5	3.7	0.0	0.0	92.8	0.0
walk	0.3	0.2	5.8	0.7	0.5	93.3

FIGURE 9 The confusion matrix of each action class using seven frames for the KTH data set.

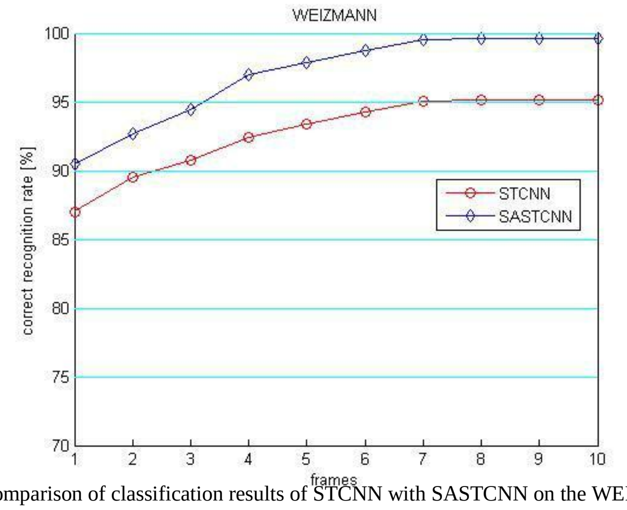


FIGURE 10 Comparison of classification results of STCNN with SASTCNN on the WEIZMANN data set.

Figure 8 shows the confusion matrix of each action class when taking seven frames for the WEIZMANN data set. Figure 9 shows the confusion matrix of each action class when taking seven frames for the KTH data set. We can see that the misclassification generally takes place between two similar actions. In the WEIZMANN data set, galloping sideways (side) is the easiest to be misclassified. 6% of it has been misclassified as walking and 4.52% of it has been misclassified as jumping. Besides this, it is also easy to be misclassified as other classes. In KTH data set, the misclassifications mainly take place among running, jogging, and walking, since these three actions are very similar. It is worth to mention that the misclassification accuracy between running and jogging is extremely high (more than 10%). For boxing, clapping and waving, it is also easy to be misclassified because of some similarities of their action modes.

The above experiments on the WEIZMANN and KTH data sets show that our approach is effective and stable. In order to further illustrate the superiority of the proposed approach, we will compare the method of this paper with the previous related researches. Table 2 shows the results of our approach compared with other methods using small video segments. Note that they can not be compared directly because of two different strategies. Our method and [19] both take a video segment as a unit, and assign an action label to each video segment. The methods of [20,21] use a time window to calculate the features, but only assign a label to the middle frame of the time window. Blank [19] assigns a label to each video segment which consists of 10 frames, features are calculated from these ten frames. Jhuang [20] assigns a label to each frame. Features are calculated from each adjacent nine frames. Although the recognition accuracy of our approach is not the highest in the best case. But only using one-frame, we can still reach a high recognition accuracy, which shows that this method has good stability. Furthermore, our method is more flexible in different application scenarios since it can adjust its own parameters according to different requirements.

In previous studies, many methods mainly deal with the whole videos. Compared with those methods, our experiments are performed with the video segments in different length. Then their results are compared with some classification results of those methods using the whole videos. For s2, [20] uses a scale normalized method, but our approach does not adopt this setting. We can see that when using the video segment containing seven frames, our approach is comparable with other methods using the whole videos. In most cases, it even achieves better results. The comparison shows that a short video segment also has rich information.

The above series of experiments show that it is enough to take a short segment from whole video for classification when using spatio-temporal convolutional neural network for video action recognition. Using seven successive frames can achieve an satisfactory result.

To analyze the effectiveness of sparse auto-combination strategy, we evaluate the classification performance of spatio-temporal convolutional neural network with and without sparse auto-combination strategy.

For the WEIZMANN data set, We still use the above mentioned experimental method, i.e. leave-one-out cross validation. According to different data combinations, the experiment is repeated 81 times to get the average of all the results. As shown in Fig. 10, the experimental results before and after adopting sparse auto-combination strategy are compared. From figure 10 we can see that the classification ability of the network using sparse auto-combination has been consistently improved from taking one frame to ten frames as the input. The experimental results shows that the action classification ability of using sparse auto-combination strategy is better than that without sparse auto-combination strategy on the WEIZMANN data set. When the length of video segment is 7 frames, the classification results can reach nearly 100%. This may be due to the fact that there is less data in the WEIZMANN data set which results in an over-complete model.

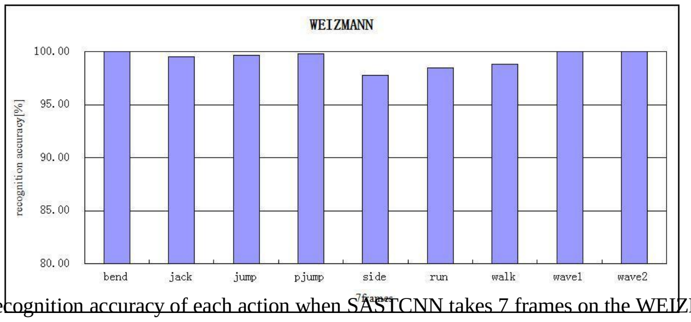


FIGURE 11 The recognition accuracy of each action when SASTCNN takes 7 frames on the WEIZMANN data set.

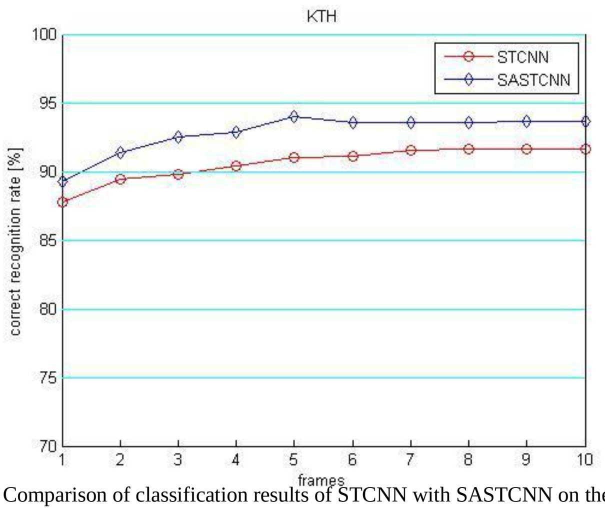


FIGURE 12 Comparison of classification results of STCNN with SASTCNN on the KTH data set.

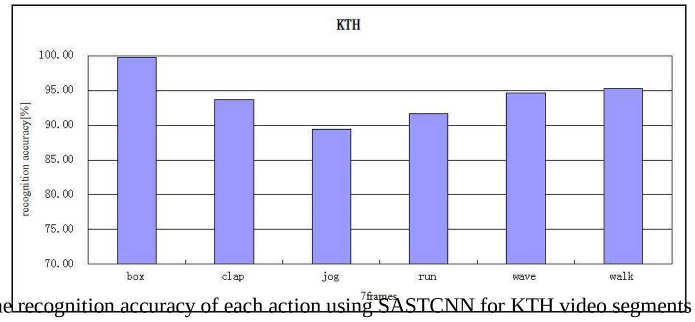


FIGURE 13 The recognition accuracy of each action using SASTCNN for KTH video segments with 7 frames.

When taking seven frames from the WEIZMANN data set, we get the recognition accuracy of each action using STCNN and SASTCNN respec-tively. The histograms of experimental results are shown in figure 11. We can see that the recognition accuracy of each action using SASTCNN has been improved, which is consistent with the trend of overall recognition accuracy performances.

For the KTH data set, we still adopt the above mentioned experimental method, namely the 5 fold cross-validation. According to different combinations, the experiment is repeated 5 times to get the average of the results. As shown in Fig. 12, comparative experiments on the KTH data set shows that the action classification ability of spatio-temporal convolutional neural network using sparse auto-combination strategy is better than that of the network without sparse auto combination strategy. It indicates the sparse auto-combination spatio-temporal convolutional neural network has robustness and generalization ability.

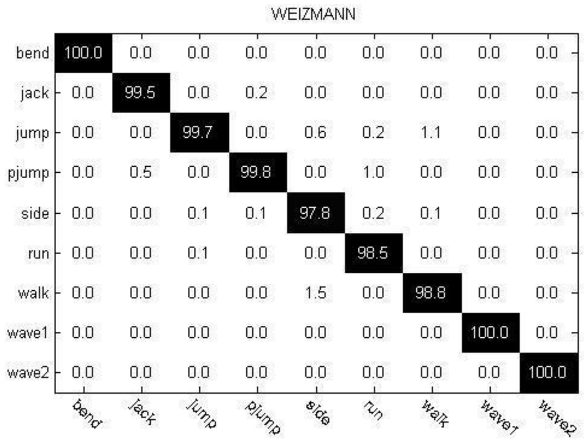


FIGURE 14 The confusion matrix using SASTCNN for WEIZMANN video segments with seven frames.

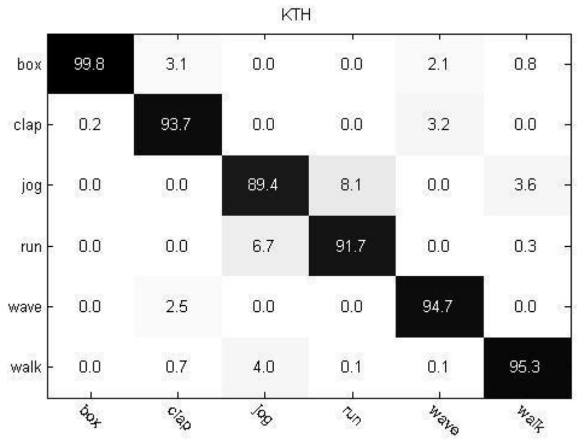


FIGURE 15 The confusion matrix using SASTCNN for KTH video segments with seven frames.

When taking seven frames for the KTH data set, we get the recognition accuracy of each action using STCNN and SASTCNN respectively. The histograms of experimental results are shown in Fig. 13. We can see that they are similar to the results on the WEIZMANN data set, i.e. the recognition accuracy of each action using SASTCNN has been consistently improved.

When we take 7 frames to form video segments, the obtained confusion matrix using SASTCNN on the WEIZMANN data set is shown in Fig. 14 and figure 15 shows the confusion matrix using SASTCNN on the KTH data set. We can see that the misclassification error of each action is very small. Compared with the confusion matrixs of STCNN in Fig. 8 and Fig. 9, the misclassification error of each action decreases. This indicates that SASTCNN is better than STCNN in distinguishing the subtle features.

Through the above series of experiments, the results show that the SASTCNN has better performance than STCNN for video action recognition. The main reason is that spatio-temporal convolution kernels are manually arranged in STCNN. Although this is simple, but it lacks flexibility so that it is not able to distinguish some subtle differences from similar actions. SASTCNN can learn optimal spatio-temporal convolution kernels automatically using sparse auto combination strategy which can extract the features and have better discriminative.

6 CONCLUSIONS

We propose a framework of spatio-temporal convolutional neural network for video action recognition so that we can extract spatial and temporal features from video cube consisting of multiple successive frames. Meanwhile, we adopt a sparse auto-combination strategy to combine multiple input feature maps. This makes spatio-temporal convolutional neural network more concise and flexible. As a result it is beneficial to improve its feature learning ability and classification ability. We carry out a series of experiments on the WEIZMANN and KTH data sets. The results show that the proposed method is effective and robust. Compared with some previous methods, it shows its advantages in video action recognition.

ACKNOWLEDGMENTS

This work was partially supported by NSFC under contract No. 61671397 and No. 61501451, and also by the scholarship from China Scholarship Council (CSC) under the Grant CSC No. 201606315022.

References

1. Johansson G. Visual perception of biological motion and a model for its analysis. *Perception psychophysics.* 1973: 201-211.
2. Campbell L, Bobick F. Recognition of human body motion using phase space constraints. *Proc IEEE Int Conf Comput Vis.* 1995: 624-630.
3. Guo Y, Xu G, Tsuji S. Understanding human motion patterns. *Proc Int Conf Pattern Recognit.* 1994: 325-329.
4. Rao C, Yilmaz A, Shah M. View-invariant representation and recognition of action. *Int J Comput Vis.* 2002: 203-226.
5. Mori G, Malik J. Estimating human body configurations using shape context matching. *Comput Vis ECCV 2002, Springer Berlin Heidelberg.* 2002: 666-680.
6. Ramanan D, Forsyth D. Automatic annotation of everyday movements. *Computer Science Division, University of California.* 2003
7. Gibson J. The perception of the visual world. 1950.
8. Gibson J. The senses considered as perceptual systems. 1966.
9. Bobick A, Davis J. The recognition of human movement using temporal templates. *IEEE Trans Pattern Anal Mach Intell.* 2001: 257-267.
10. Chen H, Chen H, Chen Y, et al. Human action recognition using star skeleton. *Proc ACM Int Workshop Vid Surveillance Sens Net.* 2006: 171-178.
11. Scovanner P, Ali S, Shah M. A 3-dimensional sift descriptor and its application to action recognition. *Proc ACM Int Conf Multimed.* 2007: 357-360.
12. Wang H, Klaser A, Schmid C, et al. Action recognition by dense trajectories. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit.* 2011: 3169-3176.
13. Li W, Zhang Z, Liu Z. Action recognition based on a bag of 3d points. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit Workshops.* 2010: 9-14.
14. Sun X, Chen M, Hauptmann A. Action recognition via local descriptors and holistic features. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit Workshops.* 2009: 58-65,
15. Lee H, Grosse R, Ranganath R, et al. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. *Proc Int Conf Mach Learn.* 2009: 609-616.
16. Norouzi M, Ranjbar M, Mori G. Stacks of convolutional restricted Boltzmann machines for shift-invariant feature learning. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit.* 2009: 2735-2742.
17. Ji S, Xu W, Yang M, Yu K. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Trans Pattern Anal Mach Intell.* 2013: 221-231.
18. Liu Z, Zhang C, Tian Y. 3D-based Deep Convolutional Neural Network for action recognition with depth sequences. *Image Vis Comput.* 2016: 93-100.

19. Blank M, Gorelick L, Shechtman E, et al. Actions as space-time shapes. *Proc IEEE Int Conf Comput Vis.* 2005: 1395-1402.

20. Jhuang H, Serre T, Wolf L, et al. A biologically inspired system for action recognition. *Proc IEEE Int Conf Comput Vis.* 2007: 1-8.

21. Niebles J, Li F. A hierarchical model of shape and appearance for human action classification. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit.* 2007: 1-8.

22. Krizhevsky A, Sutskever I, Hinton G. ImageNet classification with deep convolutional neural networks. *Proc Adv Neural Inf Process Syst.* 2012.

23. Cheron G, Laptev I, Schmid C. P-CNN: Pose-based CNN Features for Action Recognition. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit.* 2015: 3218-3226.

24. Wang Q, Zhao J, Gong D, et al. Parallelizing Convolutional Neural Networks for Action Event Recognition in Surveillance Videos, *Int J Parallel Program.* 2017: 1-26.

25. Heinbockel JH. Introduction to tensor calculus and continuum mechanics. *Trafford.* 2001.

26. Niebles J, Wang H, Li F. Unsupervised learning of human action categories using spatial-temporal words. *Int J Comput Vis.* 2008; 79(3): 299-318.

27. LeCun Y, Huang F, Bottou L. Learning methods for generic object recognition with invariance to pose and lighting. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit.* 2004: 97-104.

28. Gorelick L, Blank M, Shechtman E, et al. Actions as space-time shapes. *IEEE Trans Pattern Anal Mach Intell.* 2007; 29(12): 2247-2253.

29. Schuldt C, Laptev I, Caputo B. Recognizing human actions: A local SVM approach. *Proc Int Conf Pattern Recognit.* 2004: 32-36.

30. Lei J, Li G, Zhang J, et al. Continuous action segmentation and recognition using hybrid convolutional neural network-hidden Markov model. *IET Comput Vis.* 2016; 10(2): 537-544.

31. Lu M, Zhang L. Action recognition by fusing spatial-temporal appearance and the local distribution of interest points. *Proc Int Conf Future Comput Commun Eng.* 2014.

32. Escobar M, Kornprobst P. Action recognition via bio-inspired features: The richness of centerâĂŞsurround interaction. *Comput Vis Image Understand.* 2012; 116(5): 593-605.

33. Lu G, Kudo M. Learning action patterns in difference images for efficient action recognition. *Neurocomput.* 2014; 123(4): 328-336.

TABLE 1 The input combination strategy of feature maps of LeNet-5.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X			X
1	X	X				X	X	X			X	X	X	X		X
2		X	X	X			X	X	X			X		X	X	X
3			X	X	X		X	X	X	X			X		X	X
4				X	X	X		X	X	X	X		X	X		X
5					X	X	X		X	X	X	X		X	X	X

TABLE 2 Comparison of our approach with other methods using small video segments on the WEIZMANN data set. The last column indicates how many frames we use for classification.

	correct	frames
BLANK[19]	99.6%	10
NIEBLES[21]	55.0%	12
JHUANG[20]	93.8%	9
LU[33]	95.6%	7
LEI[30]	89.2%	5
Our work 1-frames	87.0%	1
Our work 3-frames	90.7%	3
Our work 7-frames	94.9%	7

TABLE 3 Comparison of our approach using different video segments with other methods using whole video on the KTH data set.

	1-frames	7-frames	Entire-sequence		
KTH-all	87.5%	91.6%	90.2%[17]	81.5%[26]	91.5%[31]
KTH-S1	90.9%	94.4%	96.0%[20]	92.0%[32]	95.3%[33]
KTH-S2	78.1%	82.3%	86.1%[20]		
KTH-S3	88.5%	93.2%	89.8%[20]	84.4%[32]	87.3%[33]
KTH-S4	92.2%	96.6%	94.8%[20]	92.4%[32]	90.4%[33]