Action Recognition by Learning Deep Multi-Granular Spatio-Temporal Video Representation[*]

Qing Li [1], Zhaofan Qiu [1], Ting Yao [2], Tao Mei [2], Yong Rui [2], Jiebo Luo [3]
[1] University of Science and Technology of China, Hefei 230026, P. R. China
[2] Microsoft Research, Beijing 100080, P. R. China
[3] University of Rochester, NY 14627, USA
{sealq, qiudavy}@mail.ustc.edu.cn; {tiyao, tmei, yongrui}@microsoft.com;
jluo@cs.rochester.edu

ABSTRACT

Recognizing actions in videos is a challenging task as video is an information-intensive media with complex variations. Most existing methods have treated video as a flat data sequence while ignoring the intrinsic hierarchical structure of the video content. In particular, an action may span different granularities in this hierarchy including, from small to large, a single *frame*, consecutive frames (*motion*), a short *clip*, and the entire *video*. In this paper, we present a novel framework to boost action recognition by learning a deep spatio-temporal video representation at hierarchical multi-granularity. Specifically, we model each granularity as a single stream by 2D (for *frame* and *motion* streams) or 3D (for *clip* and *video* streams) convolutional neural networks (CNNs). The framework therefore consists of multi-stream 2D or 3D CNNs to learn both the spatial and temporal representations. Furthermore, we employ the Long Short-Term Memory (LSTM) networks on the *frame*, *motion*, and *clip* streams to exploit long-term temporal dynamics. With a *softmax* layer on the top of each stream, the classification scores can be predicted from all the streams, followed by a novel fusion scheme based on the multi-granular score distribution. Our networks are learned in an end-to-end fashion. On two video action benchmarks of UCF101 and HMDB51, our framework achieves promising performance compared with the state-of-the-art.

Keywords

Action Recognition; Video Analysis; Deep Learning.

1. INTRODUCTION

Recognizing actions in videos is one of the fundamental problems of computer vision for a wide variety of applications, ranging from video surveillance, indexing and retrieval, to human computer interaction [9, 23, 29, 32]. However, video is an information-intensive media with large variations and complexities, e.g., intra-class variations caused by camera motion, cluttered background, illumination conditions, and so on. These have made action recognition a very challenging task. Moreover, as shown in Figure 1, an action may span different granularities in a video including, from small to large, a single *frame*, consecutive frames (*motion*), a short *clip*, and the entire *video*. Therefore, recognizing actions in videos therefore should take the hierarchical multi-granularity and spatio-temporal properties into consideration.

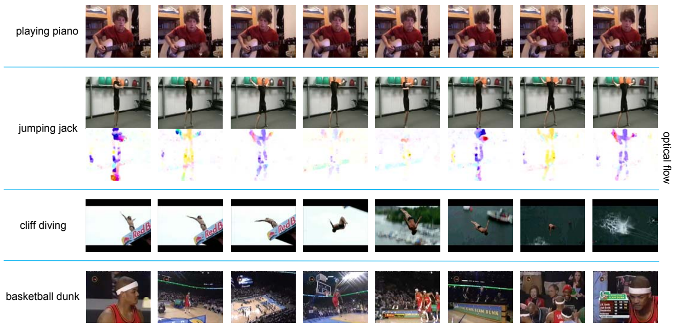

Figure 1: An action may span different granularities. For example, the action of "playing piano" can be recognized from individual *frames*, "jumping jack" may have high correlation with the optical flow images (*motion* computed from consecutive frames), "cliff diving" should be recognized from a short *clip* since this action usually lasts for few seconds, while "basketball dunk" can be reliably identified at the video granularity due the complex nature of this action. Recognizing actions therefore should take the hierarchical multi-granularity and spatio-temporal properties into consideration.

There has been extensive research on video action recognition, including hand-crafted feature-based methods [14, 15, 22, 28, 29, 33, 34, 36] or deep learning of video representations [9, 20, 16, 26, 27, 32]. The first category of research predominantly focuses on the detection of spatio-temporal interest points followed by the description of these points with local representations, while the deep learning methods heavily rely on using the Convolutional Neural Networks (CNNs) to learn visual appearance or the Recurrent Neural Networks (RNNs) to model the temporal dynamics in the video. However, most these methods treat video as a flat

[*] This work was performed at Microsoft Research Asia. The first two authors made equal contributions to this work.

ICMR'16, June 06-09, 2016, New York, NY, USA
© 2016 ACM. ISBN 978-1-4503-4359-6/16/06…$15.00
DOI: http://dx.doi.org/10.1145/2911996.2912001

data sequence while ignoring the aforementioned intrinsic hierarchical structure of the video content deeply.

In this work, we aim at investigating a multi-granular architecture to learn the deep spatio-temporal video representation for action recognition. A video is represented by a hierarchical structure with multiple granularities, including from small to large, a single *frame*, consecutive frames (*motion*), a short *clip*, and the entire *video*. We model each video granularity as a single stream by 2D CNN (for *frame* and *motion* streams) or 3D CNN (for *clip* and *video* streams). The framework therefore learns both the spatial and temporal representations via the multi-stream 2D or 3D CNNs. Furthermore, we employ the Long Short-Term Memory (L-STM) networks on the *frame*, *motion*, and *clip* streams to exploit long-term temporal dynamics. With a *softmax* layer on the top of each stream, the classification scores can be predicted from each stream. A novel fusion scheme based on the multi-granular score distribution is proposed to predict the final recognition results, where the weights of each individual stream are learnt on the score distribution. This is an effective way to reflect the importance of each stream (and its components) to the overall action recognition. It is worth noting that the entire architecture is trainable in an end-to-end fashion.

The contributions of this paper are as follows.

- We propose a novel framework to learn a deep multi-granular spatio-temporal representation for action recognition. The learned representation can capture not only the spatio-temporal nature of video sequence but also the contributions from different granularities for action recognition.
- We adopt the LSTM to model long-term temporal dynamics on the top of frame, motion and clip streams, and optimize the recognition results from all the streams through a novel fusion scheme based on the multi-granular score distribution.
- We conduct extensive experiments and show that our framework outperforms several state-of-the-art methods by clear margins on two well-known benchmarks.

The remaining sections are organized as follows. Section 2 describes related work on action recognition. Section 3 presents our multi-granular architecture for action recognition, while Section 4 formulates the importance of each component over the predicted score distribution. Implementation details are given in Section 5. Section 6 provides empirical evaluations on two popular datasets, i.e., UCF101 and HMDB51, followed by the conclusions in Section 7.

2. RELATED WORK

Video action recognition has attracted intensive research attention. We briefly group the methods for action recognition into two categories: hand-crafted feature-based and deep learning-based methods.

Hand-crafted feature-based methods usually start by detecting spatio-temporal interest points and then describe these points with local representations. Many video representations are derived from image domain and extended to measure the temporal dimension of 3D volumes. For example, Laptev and Lindeberg propose space-time interest points (STIP) by extending the 2D Harris corner detector into 3D space [14]. The global color moment feature [34], Histogram of Gradient (HOG) and Histogram of Optical Flow (HOF) [15], 3D Histogram of Gradient (HOG3D) [10], SIFT-3D [21], Extended SURF [31], and Cuboids [2] are good descriptors as the local spatio-temporal features. Recently, Wang *et al.* propose dense trajectory features, which densely sample local patches from each frame at different scales and then track them in a dense optical flow field [28, 29]. The further improvements are achieved by the compensation of camera motion [6], and the use of advanced feature encoding methods such as Bag-of-Words (BoW) [15] and Fisher Vectors [19, 22].

The most recent approaches to action recognition are to devise deep architectures for learning video representations. Qiu *et al.* perform action recognition using the Support Vector Machine with mean pooling of the CNN-based representations over frames [20]. Karparthy *et al.* extend the CNN-based architecture by stacking visual features in a fixed size of windows and using spatio-temporal convolutions for video classification [9]. Later in [27], Tran *et al.* employ 3D ConvNets trained on the Sports-1M dataset to learn video descriptors. Zha *et al.* leverage both spatial and temporal pooling on the CNN features computed on patches of video frames [35]. The late fusion is then exploited to combine spatio-temporal representations. In the work by Wang *et al.* [30], the local ConvNet responses over the spatio-temporal tubes centered at the trajectories are pooled as the video descriptors. Fisher vector is then used to encode these local descriptors to a global video representation. Most recently, the LSTM-RNN networks have been successfully employed for modeling temporal dynamics in videos. In [16], temporal pooling and LSTM are used to combine frame-level (optical flow images) representation and discover long-term temporal relationships. Srivastava *et al.* further formulate the video representation learning as an autoencoder model, which consists of the encoder and decoder LSTMs [26]. Wu *et al.* present an approach using regularization in neural networks to exploit feature and class relationships [32].

It can be observed that most existing methods treat video as a flat data sequence while ignoring the aforementioned intrinsic hierarchical structure of video content deeply. The most closely related work is the two-stream CNN approach [23]. The work applies the CNN separately on visual frames and stacked optical flows. Our method is different from [23] in that we extend two-stream to hierarchical multi-granular streams, employ 3D CNN to learn the spatio-temporal representation of video, and further utilize LSTM networks to model long-term temporal cues. In addition, our work is able to derive the fusion weights of each component from all the streams in a principled way.

3. MULTI-GRANULAR ARCHITECTURE FOR ACTION RECOGNITION

Video is essentially an information-intensive media with multiple granularities. For example, a video can be represented by a hierarchical structure including, from large to small, the entire *video*, short *clips*, consecutive frames (called *motion*, or optical flow), and individual *frames*. Different granularity depicts distinct capability to describe different actions. Recognizing actions from videos should take this intrinsic structure into account. Motivated by the above observations, we exploit such hierarchical nature of video structure and decompose video representation into multi-

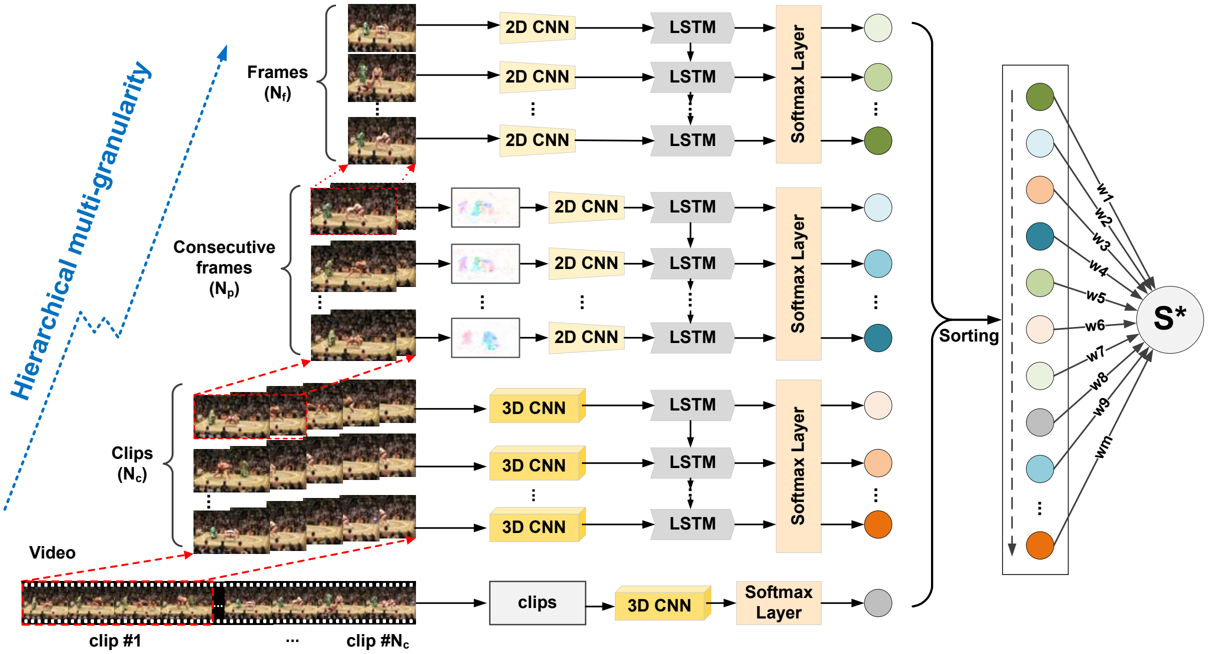

Figure 2: Multi-granular spatio-temporal architecture for video action recognition. A video is represented by the hierarchical structure with multiple granularities including, from small to large, *frame*, consecutive frames (*motion*), *clip*, and *video*. Each granularity is modeled as a single stream. 2D CNNs are used to model the *frame* and *motion* (optical flow images) streams, while 3D CNNs are used to model the *clip* and *video* streams. LSTMs are used to further model the temporal information in the *frame*, *motion*, and *clip* streams. A *softmax* layer is built on the top of each stream to obtain the prediction from each component. Suppose we have N_c clips, N_p motions (consecutive frame pairs), and N_f frames, then we have $N_c + N_p + N_f + 1$ components. The final action recognition result of the input video is obtained by linearly fusing the prediction scores from all the components with the weights learned on the score distribution. Note that this deep architecture is trainable in an end-to-end fashion.

granular streams. We design a deep architecture consisting of modeling of individual streams, 2D/3D CNNs with LSTM networks for learning spatio-temporal representations, and a novel multi-granularity score distribution scheme for fusion. Figure 2 shows the overall framework of our proposed multi-granular architecture for action recognition.

3.1 Modeling Frame Stream

For action recognition, individual video frames can provide useful characteristics as some actions are strongly associated with particular scenes and objects. To make full use of static frame appearance, VGG_19 [24], the recent superior CNN architecture for image classification, is adopted to extract high level visual features for each sampled video frame. VGG_19 is a very deep convolutional network with up to 19 weight layers (16 convolutional layers and 3 fully-connected layers). Thanks to the pre-train process using large dataset from ImageNet challenge, the VGG_19 model can be used to extract plentiful visual concepts like scenes and objects. Thus, we choose the outputs of the fully-connected layer in VGG_19 as the deep representation of a single frame.

3.2 Modeling Motion Stream

To model the displacement of consecutive frames, optical flow is extracted to describe the motion between consecutive frames. As the input to 2D CNN, the optical flow is firstly computed between a temporal window of consecutive frames with [1], and then converted to flow "image" by centering horizontal (x) and vertical (y) flow values around 128 and multiplying by a scalar such that flow values fall between 0 and 255. By this transformation, we can obtain two channels of optical flow "image," while the third channel is created by calculating the flow magnitude. Furthermore, to suppress the displacement caused by camera motion, which may introduce additional noise into CNN, a global motion component is estimated by the mean vector of each flow and then subtracted from the flow [23]. Although the input image is generated from optical flow, we also choose the pre-trained VGG_19 architecture as used for frame stream, and then fine-tune on the extracted optical flow "images."

3.3 Modeling Clip and Video Streams

Besides static frame and motion information between consecutive frames, 3D CNN is used to construct video clip features from both spatial and temporal dimensions by utilizing 3D convolutions. Unlike traditional 2D CNN, 3D CNN architecture takes video clip (multiple continuous frames) as the inputs and consists of alternating 3D convolutional and 3D pooling layers, which are further topped by a few fully-connected layers as described in [7]. For describing video clip by more powerful feature using 3D CNN, we choose the superior architecture in [27], named C3D, which aims to learn spatio-temporal features for videos using 3D CNN trained on Sports-1M video dataset [9]. As the deep network architecture is pre-trained on a large-scale video dataset, C3D can model general appearance and motion information simultaneously, which is important for action recognition. Similar to 2D CNN used for single frame and optical flow "image," we regard fully-connected layer outputs of C3D as the ex-

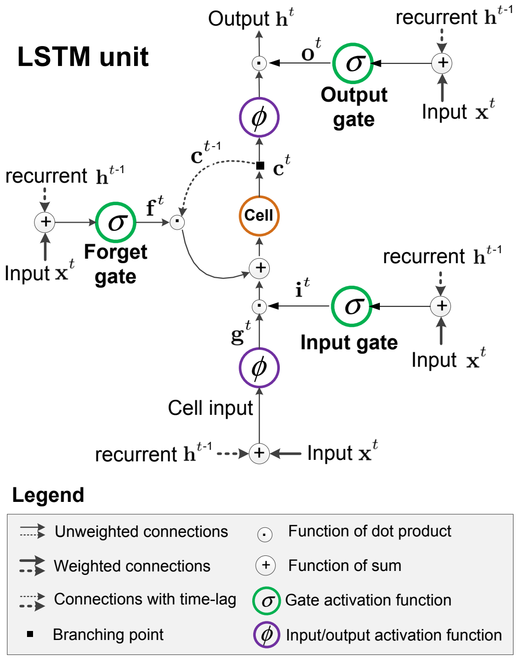

Figure 3: A diagram of a LSTM memory cell.

tracted features for sampled video clips. With the extracted features of each sampled video clip from a video by C3D, we simply perform mean pooling over all the video clips to generate video-level representations.

3.4 Modeling Temporal Dynamics with LSTM

To model the long-term temporal information in videos, we apply the Long Short-Term Memory (LSTM) on the frame, optical flow and video clip streams. The standard LSTM is a variant of RNN, which can capture long-term temporal information in the sequential data. To address the *vanishing/exploding gradients* issues when training traditional RNN, LSTM introduces a new structure called a *memory cell*. As illustrated in Figure 3, a memory cell is composed of four main elements: an input gate, a neuron with a self-recurrent connection, a forget gate and an output gate. The self-recurrent connection has a weight of 1.0 and ensures that, barring any outside interference, the state of a memory cell can remain constant from one timestep to another. The gates serve to modulate the interactions between the memory cell itself and its environment. The input gate can allow incoming signal to alter the state of the memory cell or block it. On the other hand, the output gate can allow the state of the memory cell to have an effect on other neurons or prevent it. Finally, the forget gate can modulate the memory cell's self-recurrent connection, allowing the cell to remember or forget its previous state, as needed. Many improvements have been made to the LSTM architecture since its original formulation [5] and we adopt the LSTM architecture as described in [17].

For timestep t, $\mathbf{x}^t$ and $\mathbf{h}^t$ are the input and output vector respectively, $\mathbf{T}$ are input weights matrices, $\mathbf{R}$ are recurrent weight matrices and $\mathbf{b}$ are bias vectors. *Logic sigmoid* $\sigma(x) = \frac{1}{1+e^{-x}}$ and *hyperbolic tangent* $\phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ are element-wise non-linear activation functions, mapping real values to $(0,1)$ and $(-1,1)$ separately. The dot product and sum of two vectors are denoted with $\odot$ and $\oplus$ respectively. Given inputs $\mathbf{x}^t$, $\mathbf{h}^{t-1}$ and $\mathbf{c}^{t-1}$, the LSTM unit updates for timestep t are:

$$\begin{array}{lr}
\mathbf{g}^t = \phi(\mathbf{T}_g\mathbf{x}^t + \mathbf{R}_g\mathbf{h}^{t-1} + \mathbf{b}_g) & \textit{cell input} \\
\mathbf{i}^t = \sigma(\mathbf{T}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{h}^{t-1} + \mathbf{b}_i) & \textit{input gate} \\
\mathbf{f}^t = \sigma(\mathbf{T}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{h}^{t-1} + \mathbf{b}_f) & forgetgate \\
\mathbf{c}^t = \mathbf{g}^t \odot \mathbf{i}^t + \mathbf{c}^{\mathbf{t-1}} \odot \mathbf{f}^{\mathbf{t}} & \textit{cell state} \\
\mathbf{o}^t = \sigma(\mathbf{T}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{h}^{t-1} + \mathbf{b}_o) & \textit{output gate} \\
\mathbf{h}^t = \phi(\mathbf{c}^t) \odot \mathbf{o}^t & \textit{cell output}
\end{array}$$

4. FUSION WITH MULTI-GRANULAR SCORE DISTRIBUTION (MSD)

Given the multi-granular streams in a video, we can predict the action score for each component from each stream. Inspired by the idea of addressing the temporal ambiguity of actions by learning score distribution in [4], we develop a multi-granular score fusion architecture in our deep networks rather than only using *max* or *mean*. Consequently, an improved action recognition score will be obtained by automatically aligning the relative importance to each component from all the streams based on the score distribution.

4.1 Formulation

Given a video, we combine the scores of all the components from multi-granular streams in a distribution matrix as

$$\mathbf{S} = (\mathbf{s}_1, ..., \mathbf{s}_c, ..., \mathbf{s}_C) \in \Re^{L\times C}, \tag{1}$$

where $\mathbf{s}_c \in \Re^L$ denotes the score column vector of L components from all the streams on the c^{th} action class. Next, we will define a *sort* function on the score distribution matrix as

$$sort(\mathbf{S}) = \big(sort(\mathbf{s}_1), ..., sort(\mathbf{s}_c), ..., sort(\mathbf{s}_C)\big) \in \Re^{L\times C}, \tag{2}$$

where $sort(\mathbf{s}_c) \in \Re^L$ is a function to reorder all elements of the vector $\mathbf{s}_c$ in descending order. With large L, $sort(\mathbf{s}_c)$ can represent the score distribution of all the components on the c^{th} action class.

Given N videos, each represented by the score distribution matrix $sort(\mathbf{S})$, we can learn a MSD classifier by formulating the optimization problem as

$$\min_{\mathbf{w}_c, b_c} \quad \sum_{i=1}^{N}\sum_{c=1}^{C} \max\{1 - y_i^c(\mathbf{w}_c \cdot sort(\mathbf{s}_c) + b_c), 0\} \tag{3}$$

$$s.t. \quad \sum_{l=1}^{L} w_c^l = 1, \quad c = 1, ..., C \quad , \tag{4}$$

$$w_c^1 \geq w_c^2 \geq ... \geq w_c^L \geq 0, \quad c = 1, ..., C. \tag{5}$$

For the c^{th} action class, the weight vector $\mathbf{w}_c$ and the bias item b_c can separate score distributions of positive and negative data. The loss function (3) is the sum of Hinge Losses. The constraint (4) requires the weights to have unit sum because we are learning weights for each component. The constraint (5) requires the weights to be monotonic and non-negative, because $sort(\mathbf{s}_c)$ are classification scores in descending order and we want to emphasize the relative importance of the components with high classification scores.

From the constraints, we can see that the feasible set of $\mathbf{w}_c$ includes two special cases: 1) $w_c^1 = 1, w_c^2 = ... = w_c^L = 0$, 2) $w_c^1 = w_c^2 = ... = w_c^L = \frac{1}{L}$. The former corresponds to *max* pooling, while the later refers to *mean* pooling.

4.2 Solution

As our architecture is an "end-to-end" neural network which is learnt by using standard backpropagation, we solve the optimization problem (3) by using stochastic gradient descent in a deep learning framework *Caffe* [8]. However, the two constraints (4) and (5) make the optimization difficult to be solved. To address this problem, we relax the two constraints by appending two penalty terms to the cost function J as

$$J = \mathcal{L} + \alpha \sum_{c=1}^{C} \|\mathbf{w}_c\|^2 + \beta \sum_{c=1}^{C} \big(1 - \sum_{l=1}^{L} w_c^l\big)^2 + \gamma \sum_{c=1}^{C}\sum_{l=1}^{L} m_c^l, \quad (6)$$

$$m_c^l = \begin{cases} w_c^{l+1} - w_c^l, \ if\ w_c^{l+1} > w_c^l \\ 0, \qquad\qquad if\ w_c^{l+1} \leq w_c^l \end{cases} \quad l = 1, ..., L \text{ and } w_c^{L+1} = 0, \quad (7)$$

where the first part $\mathcal{L}$ is the loss function in Eq. (3), the second is a regularization term preventing over-fitting, the rest are two penalty terms and α, β, γ are the tradeoff parameters.

Finally, the above cost function J is minimized with respect to $\{\mathbf{w}_c\}_{c=1}^{C}$ and the gradients are calculated by

$$\frac{\partial J}{\partial w_c^l} = \frac{\partial \mathcal{L}}{\partial w_c^l} + 2\alpha w_c^l - 2\beta(1 - w_c^l) + \gamma\Big(\frac{\partial m_c^l}{\partial w_c^l} + \frac{\partial m_c^{l-1}}{\partial w_c^l}\Big), \quad (8)$$

$$\frac{\partial m_c^l}{\partial w_c^l} = \begin{cases} -1, \ \ if\ \ w_c^{l+1} > w_c^l \\ 0, \quad\ \ if\ \ w_c^{l+1} \leq w_c^l \end{cases} \quad l = 1, ..., L\ , \quad (9)$$

$$\frac{\partial m_c^{l-1}}{\partial w_c^l} = \begin{cases} 1, \ \ if\ \ w_c^l > w_c^{l-1} \\ 0, \ \ if\ \ w_c^l \leq w_c^{l-1} \end{cases} \quad l = 2, 3, ..., L \text{ and } \frac{\partial m_c^0}{\partial w_c^1} = 0. \quad (10)$$

Each gradient is calculated upon a sorted score distribution. As the score order changes in each SGD iteration, backpropagation of the sorting operator is similar to that of max pooling layer. We store the index of sorted score in original vector and propagate the gradients to the corresponding element in the original vector when backpropagation. After the optimization of J in Eq. (6), we can obtain the optimal $\{\mathbf{w}_c\}_{c=1}^{C}$. With this, we compute the final improved action score for the video as

$$p_c = \mathbf{w}_c \cdot sort(\mathbf{s}_c) + b_c. \quad (11)$$

5. IMPLEMENTATIONS

Frame stream. We uniformly select 25 frames per video and adopt the VGG_19 [24] to extract frame features. The VGG_19 is first pre-trained with the ILSVRC-2012 training set of 1.2 million images and then fine-tuned by using the video frames, which is observed to be better than training from scratch. Following [23], we also use data augmentation like cropping and flipping. The learning rate starts from 10^{-3} and decreases to 10^{-4} after 14,000 iterations, then to 10^{-5} after 20,000 iterations. For temporal modeling, we extract the outputs of 4096-way fc6 layer from VGG_19 as inputs and adopt one-layer LSTM. We conduct experiments with different number of hidden states in LSTM. The LSTM weights are learnt by using the BPTT algorithm with a mini-batch size of 10. The learning rate starts from 10^{-2} and decreases to 10^{-3} after 100K iterations. The training is stopped after 150,000 iterations.

Table 1: The accuracy of *frame* and *motion* streams on UCF101 (split 1).

(a) The accuracy of different 2D CNN and LSTM used on *frame* and *motion* streams. The results are reported for late fusion.

Training setting	Frame	Motion
AlexNet	67.1%	68.4%
AlexNet + LSTM	69.3%	70.3%
VGG_19	77.9%	70.6%
VGG_19 + LSTM	79.3%	73.8%
VGG_19 + LSTM + Augmentation	**80.2%**	**74.6%**

(b) The effect of hidden layer size in the LSTM (VGG_19).

Hidden layer size	Frame	Motion
128	78.2%	71.2%
256	78.8%	72.6%
512	79.1%	73.5%
1024	**79.3%**	**73.8%**
2048	78.5%	73.1%

Motion stream. We compute the optical flow between consecutive frames using the GPU implementation of [1] in OpenCV toolbox. The optical flow is converted to a flow "image" by linearly rescaling horizontal (x) and vertical (y) flow values to [0, 255] range. The transformed x and y flows are the first two channels for the flow image and the third channel is created by calculating the flow magnitude. Moreover, the settings of VGG_19 and LSTM are the same with frame stream.

Clip stream. We define a *clip* as consecutive 16 frames, which is the same setting as [27]. The C3D is exploited to model video clip, which is pre-trained on Sports-1M [9] dataset with 1.1 million sports videos and then fine-tuned on UCF101 and HMDB51, respectively. As designed in C3D architecture, the input of C3D model is 16-frame clip and we uniformly sample 20 clips in each video. The learning rate starts from 10^{-4} and decreases to 10^{-5} after 10,000 iterations, then the training is stopped after 20,000 iterations. Again, the LSTM setting is the same with frame stream.

Video stream. The settings of video stream are similar to the clip stream. The only difference is that we do not involve LSTM after C3D and simply fuse the features of all video clips by *mean* pooling to generate the video-level representations.

6. EXPERIMENTS

6.1 Datasets

We empirically evaluate our multi-granular framework on the UCF101 [25] and HMDB51 [12] datasets. The UCF101 dataset is one of the most popular action recognition benchmarks. It consists of 13,320 videos from 101 action categories. The action categories are divided into five groups: Human-Object Interaction, Body-Motion Only, Human-Human Interaction, Playing Musical Instruments, and Sports. The HMDB51 dataset contains 6,849 video clips divided into 51 action categories, each containing a minimum of 101 clips. The experimental setup is the same for both datasets and three training/test splits are provided by the dataset organisers. Each split in UCF101 includes about 9.5K training and 3.7K test video, while a HMDB51 split

Table 2: The accuracy of *clip* and *video* streams on UCF101 (split 1) and HMDB51 (split 1).

(a) The comparisons of using features from different layers of C3D on *clip* stream.

Dataset	fc6	fc7	prob	fc6+LSTM
HMDB51	50.36%	48.65%	38.97%	**51.3%**
UCF101	83.11%	81.23%	69.81%	**83.9%**

(b) The comparisons of using the features from different layers of C3D on *video* stream.

Dataset	fc6	fc7	prob
HMDB51	**51.09%**	48.52%	39.10%
UCF101	**83.77%**	80.76%	67.01%

contains 3.5K training and 1.5K test videos. Following [23], we conduct our analyses of different streams on the first split of the UCF101 and HMDB51 datasets. The average accuracy over three splits on both datasets are reported when compared with the state-of-the-art techniques.

6.2 Evaluation of Frame and Motion Streams

We first examine the recognition performances of frame and optical flow streams from three aspects: 1) when different 2D CNN are used, 2) when LSTM is utilized to explore longer-term temporal information, and 3) how performance is affected by the size of hidden layer in LSTM learning.

The results and comparisons on UCF101 (split 1) are summarized in Table 1. Table 1a compares the accuracy of different CNN and LSTM on frame and optical flow stream, respectively, while Table 1b compares the performances with the hidden layer size of LSTM in the range of 128, 256, 512, 1024, and 2048. Compared to AlexNet [11], VGG_19 [24] with a deeper CNN exhibits significantly better performance on frame stream. Interestingly, using VGG_19 on optical flow stream gives only marginal performance gain. By additionally utilizing LSTM to explore longer-term temporal information, the improvements can be expected on both frame and optical flow streams. Furthermore, when augmenting the test frame (flow image) by cropping and flipping four corners and the center of the frame and averaging the scores across the frame and its crops, the performance can achieve 80.2% and 74.6% on frame and optical flow, respectively.

In general, increasing the hidden layer size of LSTM can lead to the improvement of the accuracy. When the hidden layer size reaches 1024 in our case, no further improvement can be obtained on both frame and optical flow streams. Note that the performances are reported based on the original frame or optical flow image with only cropping center and no flipping operation in this comparison.

6.3 Evaluation of Clip and Video Streams

Next, we turn to measure the performance of the clip and video streams in terms of features extracted from different layers of 3D CNN (C3D) on both datasets. We extract C3D features: fc6, fc7, and prob for each video clip. The recognition score is computed by late fusing the predicted score on each video clip and the accuracy comparison by using the outputs from these three different layers is shown in Table 2a. As indicated by our results, the recognition using the C3D feature of fc6 layer leads to a larger performance boost against the C3D features of fc7 and prob layers. Furthermore, the accuracy by using the feature of fc6 can achieve 51.3% and 83.9% on HMDB51 and UCF101 after longer-term temporal modeling with LSTM networks, respectively. The features for video stream are computed by averaging the video clip features separately for each type of feature and Table 2b reports the comparison of different C3D features on video stream. Similar to the observations on video clip stream, the features of fc6 layer achieves the best performance among all the three layers with a large margin.

Table 4: The performance in terms of mean accuracy (over three splits) on UCF101 and HMDB51. Please note that the methods in [29, 18, 13] are based on traditional dense trajectory which is computationally expensive, while the methods in [35, 30] combine dense trajectory and deep learning based algorithms. Our approach outperforms the deep learning-based methods without combination of dense trajectory [3, 23, 16] with a large margin. "–" means that the authors did not report their performance on this dataset. IDT: improved dense trajectory [29]; MIFS: Multi-skip Feature Stacking [13]; LRCN: Long-term Recurrent Convolutional Networks [3]; C3D: Convolutional 3D [27]; TDD: Trajectory-pooled Deep-convolutional Descriptor [30].

Method	UCF101	HMDB51
IDT [29]	85.9%	57.2%
IDT w/ Encodings [18]	87.9%	61.1%
MIFS [13]	89.1%	**65.1%**
"Slow Fusion" ConvNet [9]	65.4%	–
LRCN [3]	82.9%	–
C3D [27]	85.2%	–
Two-stream model [23]	88.0%	59.4%
Composite LSTM [26]	84.3%	–
CNN + IDT [35]	89.6%	–
Temporal Pooling + LSTM [16]	88.6%	–
TDD [30]	90.3%	63.2%
Ours	**90.8%**	**63.6%**

6.4 Evaluation of MSD

Here we evaluate the complete multi-granular architecture, which combines the four streams with the MSD fusion method. Table 3 details the accuracy across different fusion strategies on three splits of HMDB51 and UCF101, respectively. MSD consistently outperforms Max and Mean in every split of both two datasets. The improvement is observed in different types of actions. For instance, the actions "playing piano" and "biking" are better fused with Mean as the videos relevant to the two actions are consistent in content. On the other hand, the recognition of actions "cliff diving" and "basketball dunk" show much better results with Max fusion. In the experiment, MSD boosts the accuracy of these actions. Figure 4 shows the top eight weights learnt by MSD and their corresponding components of three exemplary videos from category "baseball pitch," "front crawl," and "hammering." We can easily see that all the eight components are highly related to each action. More importantly, the top eight components come from four different streams, which validates the effectiveness of MSD on fusing multi-granular information.

6.5 Comparisons with State-of-the-Art

We compare with several state-of-the-art techniques on three splits of UCF101 and HMDB51. As shown in Table 4, our multi-granular spatio-temporal architecture shows the

Table 3: The comparisons of the proposed MSD with Mean and Max fusion schemes in terms of accuracy on three splits of HMDB51 and UCF101.

Dataset	split 1			split 2			split 3		
	Mean	Max	MSD	Mean	Max	MSD	Mean	Max	MSD
HMDB51	61.5%	59.6%	**63.1%**	61.8%	59.5%	**63.5%**	62.1%	60.1%	**64.1%**
UCF101	89.6%	87.6%	**90.2%**	89.6%	87.4%	**90.3%**	91.2%	88.1%	**91.9%**

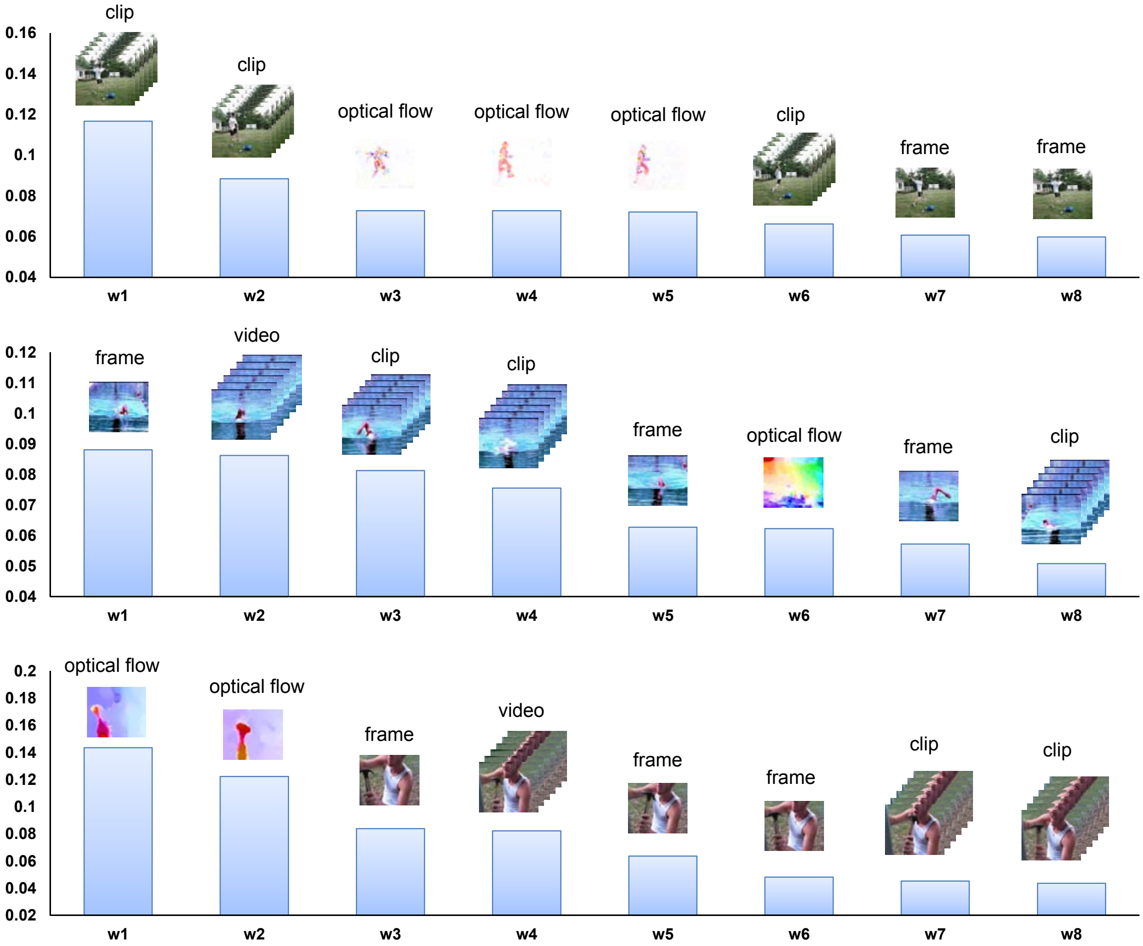

Figure 4: Examples showing the top eight weights learned by the MSD and their corresponding components in a video (top: baseball pitch, middle: front crawl, bottom: hammering). We can see that MSD is able to learn the contributions from different components for particular actions. For example, two *clip* components play important roles for recognizing "baseball pitch," while two *motion* (optical flow) components contribute more to the recognition of "hammering."

best performance on UCF101 dataset. It makes the improvement over [30] by 0.5%, which is generally considered as a significant progress on this dataset. On the HMDB51, the works [13, 30] with competitive results are based on the motion trajectory, while our approach fully relies on the deep learning architecture and is trained end-to-end. Compared with the two-stream model [23], our architecture by additionally incorporating more temporal modeling and utilizing a sophisticated fusion strategy leads to a performance boost on both datasets. It is also worth noting that in the training of the HMDB51 dataset, the work in [23] exploits UCF101 as additional training data through multi-task learning, while our approach is purely trained on HMDB51 only. In addition, the recent works in [3, 16, 26] also use the LSTM to exploit temporal information. Our approach achieves more promising results as more dimensions of cues are included.

Table 5: Run time of different streams averaged over all test videos in UCF101 dataset (milliseconds).

Stream	2D/3D CNN	LSTM	SUM
frame	750	12	762
motion	750	12	762
clip	490	10	500
video	490	–	490

6.6 Run Time

Table 5 listed the detailed run time of each stream averaged over all test videos in UCF101 dataset. The experiments are conducted on a regular server (Intel Xeon 2.40GHz CPU and 256 GB RAM) with a single NVidia K80 GPU. As each stream could be executed in parallel and the fusion with MSD provides instant responses, the average prediction time of our architecture on each video in UCF101 is about 762 milliseconds, which is very efficient. This is much faster than trajectory-based approaches, e.g. IDT, which requires about seven minutes on each video in UCF101.

7. CONCLUSIONS

We have presented a multi-granular deep architecture for action recognition in videos, which is able to incorporate information at a multitude of granularity including frame, consecutive frames (motion), clip and the entire video. Specifically, we model each granularity with two types of CNNs (2D CNN trained on frame and motion streams, and 3D C-NN on clip and video streams). We employ LSTM networks to incorporate long-term temporal modeling based on the granularity features. To fuse the recognition scores of individual components in multiple streams, the distribution of scores is exploited to find the optimal weight for each component. We show that our approach outperforms several state-of-the-art methods on two benchmark datasets.

Our future works are as follows. First, video action recognition can be enhanced by further considering audio information. The audio features can be exploited together with current four streams to more comprehensively characterize the actions. Second, the method of learning the representations of the entire video could be explored by using RNNs in an encoder-decoder framework. In addition, we will continue to conduct more in-depth investigations on how fusion weights of individual streams can be dynamically determined to boost the action recognition performance.

8. REFERENCES

[1] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*. 2004.

[2] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, 2005.

[3] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *arXiv preprint arXiv:1411.4389*, 2014.

[4] M. Hoai and A. Zisserman. Improving human action recognition using score distribution and ranking. In *ACCV*, 2014.

[5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] M. Jain, H. Jégou, and P. Bouthemy. Better exploiting motion for better action recognition. In *CVPR*, 2013.

[7] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE Trans. on PAMI*, 35(1):221–231, 2013.

[8] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[9] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[10] A. Kläser, M. Marszalek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[12] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011.

[13] Z. Lan, M. Lin, X. Li, A. G. Hauptmann, and B. Raj. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. In *CVPR*, 2015.

[14] I. Laptev and T. Lindeberg. Space-time interest points. In *ICCV*, 2003.

[15] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[16] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets deep networks for video classification. In *CVPR*, 2015.

[17] Y. Pan, T. Mei, T. Yao, H. Li, and Y. Rui. Jointly modeling embedding and translation to bridge video and language. *arXiv preprint arXiv:1505.01861v3*, 2015.

[18] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *arXiv preprint arXiv:1405.4506*, 2014.

[19] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010.

[20] Z. Qiu, Q. Li, T. Yao, T. Mei, and Y. Rui. Msr asia msm at thumos challenge 2015. In *CVPR THUMOS Challenge Workshop*, 2015.

[21] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM MM*, 2007.

[22] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep fisher networks for large-scale image classification. In *NIPS*, 2013.

[23] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.

[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[25] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human action classes from videos in the wild. *CRCV-TR-12-01*, 2012.

[26] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. In *ICML*, 2015.

[27] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. *arXiv preprint arXiv:1412.0767*, 2014.

[28] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *CVPR*, 2011.

[29] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.

[30] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015.

[31] G. Willems, T. Tuytelaars, and L. J. V. Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *ECCV*, 2008.

[32] Z. Wu, Y.-G. Jiang, J. Wang, J. Pu, and X. Xue. Exploring inter-feature and inter-class relationships with deep neural networks for video classification. In *ACM MM*, 2014.

[33] T. Yao, T. Mei, C.-W. Ngo, and S. Li. Annotation for free: Video tagging by mining user search behavior. In *ACM MM*, 2013.

[34] X. Yuan, W. Lai, T. Mei, X.-S. Hua, and X.-Q. Wu. Automatic video genre categorization using hierarchical svm. In *ICIP*, 2006.

[35] S. Zha, F. Luisier, W. Andrews, N. Srivastava, and R. Salakhutdinov. Exploiting image-trained CNN architectures for unconstrained video classification. *arXiv preprint arXiv:1503.04144*, 2015.

[36] Z.-J. Zha, T. Mei, Z. Wang, and X.-S. Hua. Building a comprehensive ontology to refine video concept detection. In *ACM SIGMM Workshop on MIR*, 2007.