Noname manuscript No.
(will be inserted by the editor)

Learning Hierarchical Video Representation for Action Recognition

Qing Li · Zhaofan Qiu · Ting Yao · Tao Mei · Yong Rui · Jiebo Luo

Received: date / Accepted: date

Abstract Video analysis is an important branch of computer vision due to its wide applications, ranging from video surveillance, video indexing and retrieval to human computer interaction. All of the applications are based on a good video representation, which encodes the content of video into a feature vector with a fixed length. Most existing methods treat video as a flat image sequence, but from our observations we argue that video is an information-intensive media with intrinsic *hierarchical* structure, which is largely ignored by previous approaches. Therefore, in this work we represent the hierarchical structure of video with multiple granularities including, from short to long, single *frame*, consecutive frames (*motion*), short *clip*, and the entire *video*. Furthermore, we propose a novel deep learning framework to model each granularity individually. Specifically, we model the *frame* and *motion* granularities with 2D convolutional neural networks, and model the *clip* and *video* granularities with 3D convolutional neural networks. Long Short-Term Memory (LSTM) networks are applied on the *frame*, *motion* and *clip* to further exploit the long-term temporal clues. Consequently, the whole framework utilizes multi-stream CNNs to learn a hierarchical representation that captures spatial and temporal information of video. To validate its effectiveness in video analysis, we apply this video representation to the action recognition task. We adopt a distribution-based fusion strategy to combine the decision scores from all the granularities, which are obtained by using a *softmax* layer on the top of each stream. We conduct extensive experiments on three action benchmarks (UCF101, HMDB51 and CCV) and achieve competitive performance against several state-of-the-art methods.

Keywords Video Representation Learning · Action Recognition · Deep Learning

Qing Li · Zhaofan Qiu
University of Science and Technology of China, P.R. China
E-mail: {sealq, qiudavy}@mail.ustc.edu.cn

Ting Yao · Tao Mei · Yong Rui
Microsoft Research, Beijing, P.R. China
E-mail: {tiyao, tmei, yongrui}@microsoft.com

Jiebo Luo
University of Rochester, New York, USA
E-mail: jluo@cs.rochester.edu

1 Introduction

Video analysis is attracting more and more research attention in recent years. This is due partially to the explosive increasing amount of video and the wide range of potential applications based on video analysis such as video surveillance, video highlight detection, and human computer interaction [11,23,35]. Video analysis heavily relies upon a good video representation. However, devising a robust and discriminative video representation is very challenging due to not only the visual variance caused by camera motion, viewpoint changing or illumination conditions, but also the complex temporal structure of video itself. Traditional hand-crafted methods usually start by detecting spatial-temporal interest points and then represent these points with local descriptors. For instance, Wang *et al.* propose dense trajectory features in [30], which tracks densely-sampled local frame patches over time and extract several traditional features based on the trajectories. The dense trajectory features can achieve very good performance on video action recognition by simply training a SVM classifier on them.

In contrast to the hand-crafted features, there is recently a big surge of automatically learning a represen-

tation from the raw data using deep neural networks. Among these networks, two-dimensional convolutional neural networks have exhibited state-of-the-art performance in image analysis tasks like classification or detection [13,24,28]. For video analysis, Karpathy *et al.* [11] extend the 2D CNN into the temporal dimension by stacking frames over time and achieve promising results on action recognition task. Another important work is the two-stream CNN approach proposed by Simonyan *et al.* [23], which use two different 2D CNNs on individual frame and stack optical flows respectively to capture the spatial and motion information.

As we can see, what is in common for these existing methods is that they treat video as a flat image sequence. However, from our observation, video embeds its intensive information in a *hierarchical* structure. Concretely, if we focus on the content of a single frame from the video, we can only get information about the objects and contexts. And if we select two consecutive frames and compute the displacement between them, the motion of the objects in video can be observed. Furthermore, by inspecting more and more continuous frames, we can observe the more complex motion pattern of the objects. In short, *hierarchical* means that when we focus on different granularity of the video, we can get different kind of information.

To make full use of the intensive information from video, this paper proposes a multi-stream deep learning framework to learn a hierarchical video representation, which not only harness the spatial-temporal clues, but also consider the multiple granularities of video. To represent the hierarchical structure of video, we define four granularities from short to long, i.e., a single *frame*, consecutive frames (*motion*), a short *clip*, and the entire *video*. we model the *frame* and *motion* streams by 2D CNNs, while the *clip* and *video* streams are processed by 3D CNNs. Therefore the framework can learn both visual appearance and short-term motion information via the multi-stream CNNs. Furthermore, the Long Short-term Memory networks are utilized on the *frame*, *motion* and *clip* streams for the long-term temporal modeling, while the outputs of 3D CNN on each clip are combined by mean pooling as the representation of *video* stream.

To verify the power of this hierarchical video representation, we apply it to the action recognition task. First we equip each stream with a softmax layer to predict the classification scores. And as shown in Figure 1, an action may span different granularities in a video. Therefore, instead of mean pooling the classification scores from different streams, we adopt a novel fusion strategy based on the multi-granular score distribution to predict the final probabilities on every action class.

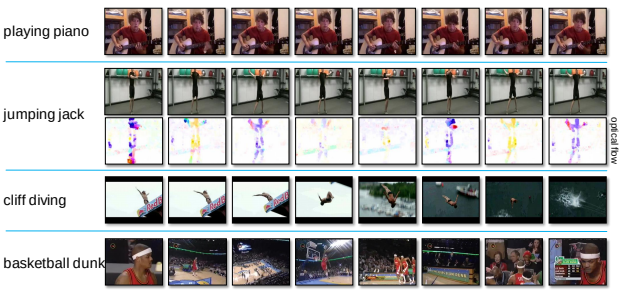

Fig. 1: An action may span different granularities. For example, the action of "playing piano" can be recognized from individual *frames*, "jumping jack" may have high correlation with the optical flow images (*motion* computed from consecutive frames), "cliff diving" should be recognized from a short *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 spatial-temporal properties into consideration.

We train classifiers on the score distributions to learn the weights of each individual component, which can effectively reflect the importance of each stream and its components to the overall recognition result. And what is worth to mention is that all the granularities are processed in parallel and the whole framework is end-to-end trainable.

The main contributions of this work can be summarized as follows:

- we propose an end-to-end hybrid deep learning framework, which exploits the multiple granularities of video to learn a hierarchical representation for video. The framework can model not only the spatial and short-term motion patterns, but also the long-term temporal clues of the video.
- We adopt the LSTM to model long-term temporal clues on the top of *frame*, *motion* and *clip* streams. We show that all the streams work well with the LSTM, which is complementary to the traditional methods without considering the temporal order of frames in video.
- We apply the hierarchical video representation to action recognition task by integrating all the granularities with a novel distribution-based fusion strategy. The fusion scheme not only can reflect the importance of different streams, but also is computationally efficient in training and testing.
- Through an extensive set of experiments, we demonstrate that our proposed framework outperforms several alternative methods with clear margins. On three

popular benchmarks (UCF-101, HMDB-51, and CCV), we obtain the state-of-art results.

The rest of this paper is organized as follows. Section 2 reviews related work on video representation learning with hand-crafted methods and deep learning architecture. Section 3 describes the proposed multi-granular framework for learning hierarchical video representation in detail, while Section 4 formulates the novel fusion scheme based on multi-granular score distribution for action recognition. The experiment settings and implementation details are given in Section 5. Section 6 provides experimental results and analyses on three well-known benchmarks (UCF101, HMDB51 and CCV), followed by the conclusions and future work in Section 7.

2 Related Work

As aforementioned, video analysis has been an active research topic in multimedia and computer vision. A good video analysis system relies heavily on the extracted video features so significant efforts have been paid to design discriminative and robust video representations [11,30,20]. And here we just focus on the review of recent work in the context of action recognition task of video analysis.

Hand-crafted Representations. There has been numerous work focusing on developing discriminative features by hand that are expected to be able to distinguish different categories and be robust to the large intra-class variances [30,37,12]. Designing video representations usually consists of two steps: detecting spatial-temporal interest points, representing these points with local descriptors. For the design of interest points detectors, Laptev and Lindeberg [16] extend the 2D Harris corner detector into 3D space to find the space-time interest points (STIP). To describe the found interest points, we can utilize some image-based descriptors, such as HOG and SIFT, to extract visual appearance information from individual frames of video. In addition to the static appearance information, the motion information are also very crucial for understanding video contents so a lot of efforts are paid to design descriptors taking into account the object movements. A popular way to extract motion descriptors is to extend the frame-based local descriptors into 3D space. For example, Klaser *et al.* propose HOG3D by extending the idea of integral images for fast descriptor computation [12]. Besides, SIFT-3D [22], Extended SURF [33], and Cuboids [2] are also good choices as the local spatial-temporal descriptors. 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 [30]. This method has demonstrated very competitive performances on several popular benchmarks. In addition, the further improvements can be achieved by the compensation of camera motion, and the use of advanced feature encoding methods like Fisher Vectors. What is worth noting is that these spatial-temporal video descriptors can only represent local motion pattern within a very short period, and the popular descriptors encoding methods like Bag-of-Words (BoW) just discard the temporal order information of the descriptors totally.

Deep Learning Representations. Motivated by the great success of deep neural networks (especially the ConvNets) on image classification tasks [13,24,28], there are recently a lot of attempts to devise deep architectures for learning video representations. Ji *et al.* extend 2D CNN architecture into spatial-temporal space by operating on stacked video frames [8]. Karparthy *et al.* compare several different fusion architectures for video classification [11] Later in [29], Tran *et al.* propose to train 3D ConvNets on a large labelled video dataset Sports-1M to learn generic spatial-temporal features which can be computed very efficiently. Xu *et al.* adopt advanced feature encoding strategies VLAD to make the CNN features generalize better. Zha *et al.* leverage both spatial and temporal pooling on the CNN features computed on patches of video frames [39]. Simonyan *et al.* propose an novel two-stream approach, where two different ConvNets are trained on individual frame and stacked optical flows respectively to more explicitly capture the spatial and short-term motion information. Final predictions can be obtained by mean-pooling the decision scores of the two ConvNets or train a SVM classifier on the concatenation of the outputs of the two ConvNets. The late fusion is then exploited to combine spatial-temporal representations. A recent work by Wang *et al.* [31] incorporate the two-stream method with the traditional dense trajectory by pooling the local ConvNet responses over the spatial-temporal tubes centered at the trajectories as the video representation. Similar to the hand-crafted features, the CNN-based representations are also not able to model the long-term temporal information and the proposed several fusion schemes don't consider the temporal order of the different parts of the video.

Temporal Modeling in Video. As aforementioned, both the hand-crafted and CNN features cannot model the long-term information. So there is also extensive work to explore the long-term temporal dynamics in video. For instance, Fernando *et al.* propose to learn a function capable of ordering the frames of a video, which can capture well the evolution of the appearance

within the video. And in the recent years, RNN attracts a lot of research attention on many sequential learning tasks like speech recognition [4] and machine translation [27]. RNN can deal with sequential data with variable length so theoretically it can be utilized to model the long-term temporal dynamics in video. In [19], 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 in an unsupervised manner, which consists of the encoder and decoder LSTMs [26].

It can be observed that most existing methods treat video as a flat data sequence while ignoring the aforementioned intrinsic hierarchical structure of the video content deeply. The most closely related work is the two-stream CNN approach proposed by Simonyan *et al.* [23]. The work applies the CNN separately on individual frame and stacked optical flows. Our method is different from [23] in that we extend two-stream to the multi-granular streams, employ 3D CNN to learn the spatial-temporal representation of video, and further utilize LSTM networks to model long-term temporal cues. Besides, our framework adopts a more principled fusion scheme to integrate each component from all the streams.

3 Hierarchical Video Representation

Compared with image, video contains more intensive information, which is essentially embedded in a hierarchical structure. So a good video representation should cover all the aspects of the hierarchical structure. Then how can we define the hierarchical structure of video? In this paper, we represent the hierarchical structure by defining multiple granularities in video from short to long, i.e., single *frame*, consecutive frames (*motion*), short *clip*, and the whole *video*. And we devise a multi-stream deep learning architecture to model each granularity. Figure 2 gives an overview of our framework, and next we will introduce the implementation details of each stream respectively.

3.1 Modeling Frame Stream

For representing video, individual frames can provide some static useful clues like particular scenes and objects. In recent years, convolution neural network has proved its surprising power in image analysis, so CNN is very good choice to make full use of the static frame appearance. Among the proposed CNN architectures in recent work, we choose the VGG_19 [24], a superior CNN architecture for image classification, to extract the high level visual features from each sampled individual frame. VGG_19 is a deep convolutional network with up to 19 weight layers (16 convolutional layers and 3 fully-connected layers). And considering the relatively small size of labelled video dataset, training the network from scratch will cause a very heavy overfitting problem. In view of the great similarity between the video frames and the images from ImageNet, it is reasonable to pre-train the network on ImageNet, which is a much larger dataset with labelled images. Then we finetune the network on the frames from the video dataset to get the final model. Thanks to the power of this transfer learning, we alleviate the overfitting problem to a great extent. Finally, we utilize the outputs of the fully-connected layer of the finetuned VGG_19 model as the representation of individual frame, which contains the scenes and objects information of the video.

3.2 Modeling Motion Stream

Complementary to frame, motion is another important clue for video representation. The crucial difference between video and static image is the movement of the objects in video so it is very necessary to consider the motion information when representing video. Following [23], we compute the optical flow [1] to explicitly measure the displacement between two consecutive frames. Furthermore, to alleviate the effect of camera motion in video, we subtract from every optical flow its mean vector. This preprocessing can be viewed as a very rough estimation for the camera motion and more advanced techniques can be explored for compensating the camera motion, but that topic is just out of scope of our work. After getting the measurements of the objects' motion in video, we need to encode these measurements into a fixed length representation. A smart way proposed in [3] is to convert the optical flow into "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. Having converted flow into "image", What we do next is very similar to the frame stream, i.e., to finetune the pre-trained VGG_19 on the extracted optical flow "images." And then we compute the outputs of the fully-connected layer of the finetuned VGG_19 model as the representation of motion in video.

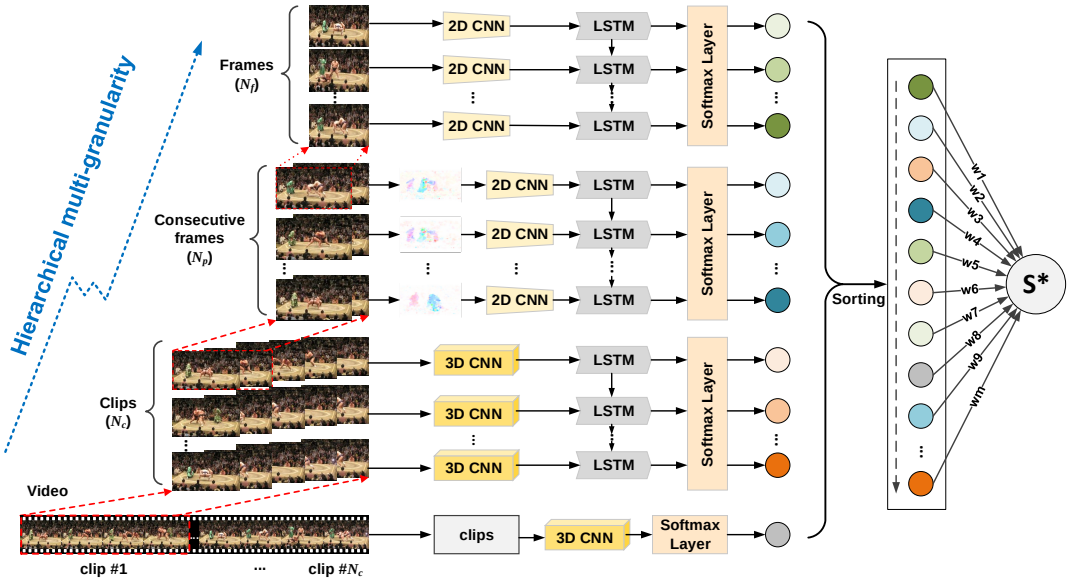

Fig. 2: Multi-granular spatial-temporal architecture for video action recognition. A video is represented by the hierarchical structure with multiple granularities including, from short to long, *frame*, consecutive frames (*motion*), *clip*, and *video*. Each granularity is modeled as a single stream. 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 learnt on the score distribution. Note that this deep architecture is trainable in an end-to-end fashion.

3.3 Modeling Clip and Video Streams

As we can see above, the frame only contains the scenes and objects information, while the motion is considered between only two consecutive frames. Then we try to figure out an approach than can explore the motion pattern of the objects in multiple consecutive frames. 3D CNN is just a very good option, which takes a video clip (multiple continuous frames) as the input and conduct 3D convolutions in both spatial and temporal dimensions. In our work, we adopt the superior 3D CNN architecture proposed in [29], named C3D, which takes 3D convolution and 3D pooling alternatively. C3D is pre-trained on a large-scale labeled video dataset, i.e., Sports-1M, to learn a generic spatial-temporal video representation. Following the spirit of frame and motion streams, we also finetune the pre-trained C3D model on clips extracted from the video dataset to get the final model. Then we take the outputs of fully-connected layer of C3D as the representations for the sampled clips, and we think these representations can capture the appearance and motion pattern information simultaneously. To represent the video globally, we just adopt mean pooling of the features of all the clips as the video-level representations. We have to admit that this is simple as well as rough, and a better way to get the video representation can be explored in the future.

3.4 Temporal Modeling with LSTM

During the modeling of the frame, motion and clip streams, each CNN architecture just takes one component (i.e., one frame, optical flow "image," or clip) in video, and the temporal order of the components is totally discarded. To learn the long-term dependencies between different components of video, we employ the Long Short-Term Memory (LSTM) on frame, motion, and clip streams. LSTM is a type of RNN with special memory cells and controllable gate mechanism and has achieved great success in many long range sequential modeling task, like speech recognition [4] and machine translation [27]. when dealing with long sequential data, LSTM doesn't suffer the "vanish gradients" issue like the traditional RNN. In general, LSTM recursively maps the input representations at the current step to the output representation based on the current hidden state, and thus the training process of LSTM should be in a sequential manner. At last, we can compute a decision score at each time step with a softmax layer using the hidden states from the LSTM layer.

More formally, given a sequence of representations $(\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^T)$, LSTM maps the input sequence to an output sequence of hidden states $(\mathbf{h}^1, \mathbf{h}^2, ..., \mathbf{h}^T)$ by updating the hidden state in the network with following formula recursively from $t = 1$ to $t = T$:

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

where $\mathbf{x}^t$ and $\mathbf{h}^t$ are the input and output hidden state vector respectively with the superscript t denoting the t-th time step, $\mathbf{i}^t$, $\mathbf{f}^t$, $\mathbf{c}^t$, $\mathbf{o}^t$ are the activation vectors of input gate, forget gate, memory cell and output gate, 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 $\bigoplus$ respectively. Figure 3 is an illustration of an LSTM unit.

The core idea of LSTM is to introduce a new structure called *memory cell*, which can store information over time to explore long-range dynamics as well as alleviate the "vanish gradients" effect. As we can see 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 which is crucial for controlling the gradient flow in back-propagation through time to avoid the gradients vanish. 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 [6] and we adopt the LSTM architecture as described in [38].

In order to obtain the decision scores over C classes at the time step t, we apply a softmax layer on the

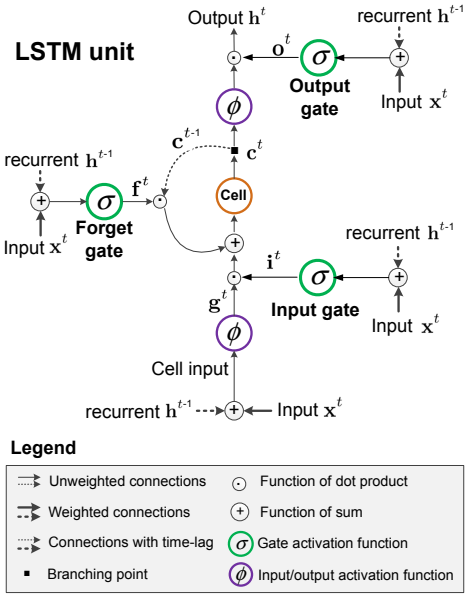

Fig. 3: A diagram of a LSTM memory cell.

hidden state of time step t of LSTM to compute the probabilities as:

$$
s_c = \frac{exp(\mathbf{w}_c^T \mathbf{h}^t + b_c)}{\sum_{c' \in C} exp(\mathbf{w}_{c'}^T \mathbf{h}^t + b_{c'})} \tag{1}
$$

where s_c, $\mathbf{w}_c$ and b_c are the prediction score, the corresponding weight vector and bias term of the c-th class, respectively. The LSTM is trained with the Back-propagation Through Time (BPTT) algorithm [32], which unrolls the model along the time dimension into a feed forward neural nets and back-propagate the gradients.

4 Multi-Granular Score Distribution Fusion for Action Recognition (MSD)

In the section 3, we have learnt a hierarchical video representation from the multiple granularies of video, and then through a softmax layer, we can compute the decision score for each component from each stream over each action class. Actually we can simply use the *mean* or *max* pooling, but as we have explained in the introduction section, different action may span different granularity. Therefore, we try to figure out a more adaptive fusion scheme to fuse all the decision scores, which is expected to be able to attach different importance to different component. Inspired by the idea of addressing the temporal ambiguity of actions by learning score distribution in [5], we devise a novel fusion scheme based on the multi-granular score fusion. Specifically, 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.

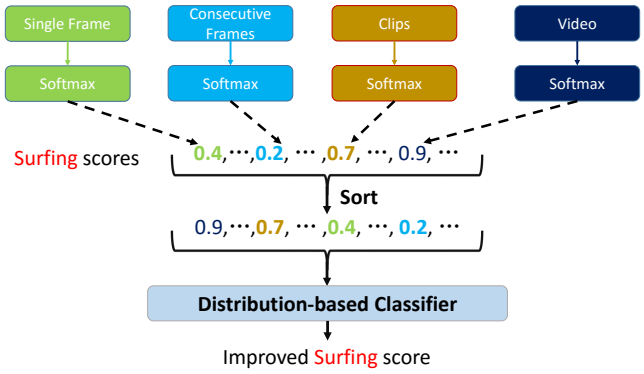

Fig. 4: An example of Multi-granular score fusion over class "Surfing."

4.1 Formulation

Firstly, we put together the decision scores of all components from the mutiple streams to form a distribution matrix as:

$$\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_C) \in \Re^{L \times C}, \tag{2}$$

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

$$sort(\mathbf{S}) = (sort(\mathbf{s}_1), sort(\mathbf{s}_2), ..., sort(\mathbf{s}_C)) \in \Re^{L \times C}, \tag{3}$$

where $sort(\mathbf{s}_c) \in \Re^L$ is to reorder all elements of the vector $\mathbf{s}_c$ in descending order. With L large enough, we can safely presume that $sort(\mathbf{s}_c)$ can approximate the score distribution of this video over the c^{th} action class.

Then what we are going to do next is to train a binary classifier for each action class to distinguish whether a score distribution belongs to this class. The merit of training such a distribution classifier is that we can make full use of the information from all the components, rather than a summary statistic (*mean*) or a extreme one (*max*). And as we observe, the score distributions vary from class to class. When training the distribution classifier for a certain class, the positive samples consist of the score distributions from this class, while the negative ones come from the other classes. Figure 4 gives an example on the "Surfing" class to illustrate how multi-Granular score fusion works.

More formally, we can formulate the training process as the following optimization problem:

$$\min_{\mathbf{w}_c, b_c} \quad \sum_{i=1}^{N}\sum_{c=1}^{C} \max\{1 - y_i^c(\mathbf{w}_c \cdot sort(\mathbf{s}_c) + b_c), 0\} \tag{4}$$

$$s.t. \quad \sum_{l=1}^{L} w_c^l = 1, \quad c = 1, ..., C \ , \tag{5}$$

$$w_c^1 \geq w_c^2 \geq ... \geq w_c^L \geq 0, \quad c = 1, ..., C. \tag{6}$$

For the c-th action class, the weight vector $\mathbf{w}_c$ and the bias item b_c form a classifier to compute a confidence score over this action class for every score distribution. The objective function (4) is to minimize the sum of Hinge Loss. The constraint (5) requires the weights to have unit sum because we are assigning weights to each component. The reason for the constraint (6), which requires the weights to be monotonic and non-negative, is that $sort(\mathbf{s}_c)$ are actually classification scores in descending order and naturally we want to attach more importance to the components with high classification scores. By inspecting the two constraints, we can infer that the feasible set of $\mathbf{w}_c$ contains two special cases:

1. $w_c^1 = 1, w_c^2 = ... = w_c^L = 0$: *max* pooling.
2. $w_c^1 = w_c^2 = ... = w_c^L = \frac{1}{L}$: *mean* pooling.

4.2 Solution

Although we can train the distribution classifier in a standalone mode, we try to integrate the training process with the former video representation learning to form an "end-to-end" framework. But the two constraints make it difficult to train the framework in an "end-to-end" manner. To do this, we relax the two constraints into the objective function J by appending two penalty terms as

$$J = \mathcal{L} + \alpha \sum_{c=1}^{C} \|\mathbf{w}_c\|^2 + \beta \sum_{c=1}^{C} (1 - \sum_{l=1}^{L} w_c^l)^2 + \gamma \sum_{c=1}^{C}\sum_{l=1}^{L} m_c^l, \tag{7}$$

$$m_c^l = \begin{cases} w_c^{l+1} - w_c^l, if \ w_c^{l+1} > w_c^l \\ 0, \qquad\quad if \ w_c^{l+1} \leq w_c^l \end{cases}, l = 1, ..., L \text{ and } w_c^{L+1} = 0, \tag{8}$$

where the first part $\mathcal{L}$ is the Hinge loss in Eq. (4), the second is a regularization term preventing over-fitting, followed by two penalty terms. α, β, γ are the tunable hyperparameters.

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

$$\frac{\partial J}{\partial w_c^l} = \frac{\partial \mathcal{L}}{\partial w_c^l} + 2\alpha w_c^l - 2\beta(1 - w_c^l) + \gamma\Big(\frac{\partial m_c^l}{\partial w_c^l} + \frac{\partial m_c^{l-1}}{\partial w_c^l}\Big), \tag{9}$$

$$\frac{\partial m_c^l}{\partial w_c^l} = \begin{cases} -1, & if \ \ w_c^{l+1} > w_c^l \\ 0, & if \ \ w_c^{l+1} \leq w_c^l \end{cases} \quad l = 1, ..., L \ , \tag{10}$$

$$\frac{\partial m_c^{l-1}}{\partial w_c^l} = \begin{cases} 1, & if \ \ w_c^l > w_c^{l-1} \\ 0, & if \ \ w_c^l \leq w_c^{l-1} \end{cases} l = 2, 3, ..., L \text{ and } \frac{\partial m_c^0}{\partial w_c^1} = 0. \tag{11}$$

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%

After the optimization of J in Eq. (7), we can obtain the optimal $\{\mathbf{w}_c\}_{c=1}^{C}$. With this, we compute the final improved action score for the video as

$$p_c = \mathbf{w}_c \cdot sort(\mathbf{s}_c) + b_c. \tag{12}$$

The gradient is also backpropagated to the score distribution. As the order of the decision scores have been changed by the *sort* function, we need to store the index of the sorted scores in original vector in the forward process and propagate the gradients to the corresponding element in the original vector when backpropagating. This practice is very similar to that of max pooling layer. After incorporating the learning of distribution classifier into the previous framework, we can train the whole architecture in an "end-to-end" fashion using the mini-batch SGD and standard backpropagation algorithm, which are implemented in the deep learning framework *Caffe* [9].

5 Experimental Setup

5.1 Datasets

We adopt three popular datasets to evaluate the proposed hierarchical video representation and multi-granular score fusion scheme.

HMDB51 [14] and **UCF101** [25]. The UCF101 dataset is one of the most popular action recognition benchmarks. It consists of 13,320 videos clips 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 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. What needs noting is that the average length of videos in UCF101 and HMDB51 is about 6 seconds, which is much shorter than that of the videos from the real world.

Columbia Consumer Videos (CCV) [10]. The CCV dataset contains 9,317 YouTube videos annotated according to 20 classes, which are mainly events like "Soccer", "Cat", "WeddingCeremony", "Beach". We follow the convention defined in [10] to use a training set of 4,659 videos and a test set of 4,658 videos. The results are evaluated by average precision (AP) over each class, and we report the mean AP (mAP) as the overall measure when compared with the baselines. Comparing with UCF101 and HMDB51, the videos in CCV usually last about several minutes and the categories are relatively more complex and high-level than the simple actions in UCF101 and HMDB51.

5.2 Implementation details

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 much 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.

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 [29]. The C3D is exploited to model video clip, which is pre-trained on Sports-1M [11] 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 Results and Analyses

In this section, we show the experiment results and analyses. As we can see, each granularity of the hierarchical representation of video can be used for action recognition solely. So we first evaluate the performance of different streams on UCF101 and HMDB51, followed by the evaluation of multi-granular score fusion. Then we compare the performance of our whole framework with the state-of-the-art approaches on these two dataset. Next we evaluate the proposed multi-granular framework on the more complex dataset CCV for video classification. Last but not the least, we report the running time of each stream and the whole framework to justify the feasibility of our framework in the real applications.

6.1 Evaluation of Frame and Motion Streams

Table 1 shows the results and comparisons of frame and motion streams on UCF101 (split 1). First we examine the influence of using 2D CNNs with different depths (AlexNet and VGG_19) on the frame and motion stream. As we can see in the first and third rows in Table 1a, compared with AlexNet [13] (8 weight layers), VGG_19 [24] (19 weight layers) exhibits significantly better performance (more than 10 percent gain) on frame stream. The boosting performance should be credited to the increased depth on the CNN architecture, which can learn a better representation for the scene and object information in the frames. But this case doesn't apply to the motion stream, and we can only get very marginal improvement when using the deeper CNN. We speculate that deeper CNN suffers from the overfitting problem on the motion stream, which can be explained by two reasons: 1) optical flow "image" is relatively simple and doesn't contain as many details as natural images, 2) the big gap between the optical flow "image" and natural images from ImageNet makes the transfer learning work not very well. We also evaluate the LSTM to investigate the significance of leveraging the long-term temporal clues for action recognition. On UCF101, LSTM can improve both the frame and motion 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.

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%

Besides, we evaluate the influence of hidden state size of LSTM as seen in the 1b. 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.

6.2 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 UCF101 and HMDB51. We extract activations of the C3D layers: fc6, fc7, and prob for each clip. The recognition score is computed by late fusing the predicted score on each 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

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%**

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.

6.3 Evaluation of Multi-Granular Score Fusion

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 5 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.

6.4 Comparisons with the 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 spatial-temporal architecture exhibits the highest performance on UCF101 dataset. It makes the improvement over [31] by 0.5%, which is generally considered as a significant progress on this dataset. On the HMDB51, the works [15,31] 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, [23] exploit UCF101 as additional training data through multi-task learning while our architecture is only trained on the HMDB51 data. In addition, the recent works in [3,19,26] also use the LSTM to exploit the temporal information. Our method achieves more promising results as more dimensions of cues are taken into account.

Table 4: The performance in terms of mean accuracy (over three splits) on UCF101 and HMDB51. Please note that the methods in [30,21,15] are based on traditional dense trajectory which is computationally expensive, while the methods in [39,31] combine dense trajectory and deep learning based algorithms. Our approach outperforms the deep learning-based methods without combination of dense trajectory [3,23,19] with a large margin. "–" means that the authors did not report their performance on this dataset. IDT: improved dense trajectory [30]; MIFS: Multi-skip Feature Stacking [15]; LRCN: Long-term Recurrent Convolutional Networks [3]; C3D: Convolutional 3D [29]; TDD: Trajectory-pooled Deep-convolutional Descriptor [31].

Method	UCF101	HMDB51
IDT [30]	85.9%	57.2%
IDT w/ Encodings [21]	87.9%	61.1%
MIFS [15]	89.1%	**65.1%**
"Slow Fusion" ConvNet [11]	65.4%	–
LRCN [3]	82.9%	–
C3D [29]	85.2%	–
Two-stream model [23]	88.0%	59.4%
Composite LSTM [26]	84.3%	–
CNN + IDT [39]	89.6%	–
Temporal Pooling + LSTM [19]	88.6%	–
TDD [31]	90.3%	63.2%
Ours	**90.8%**	**63.6%**

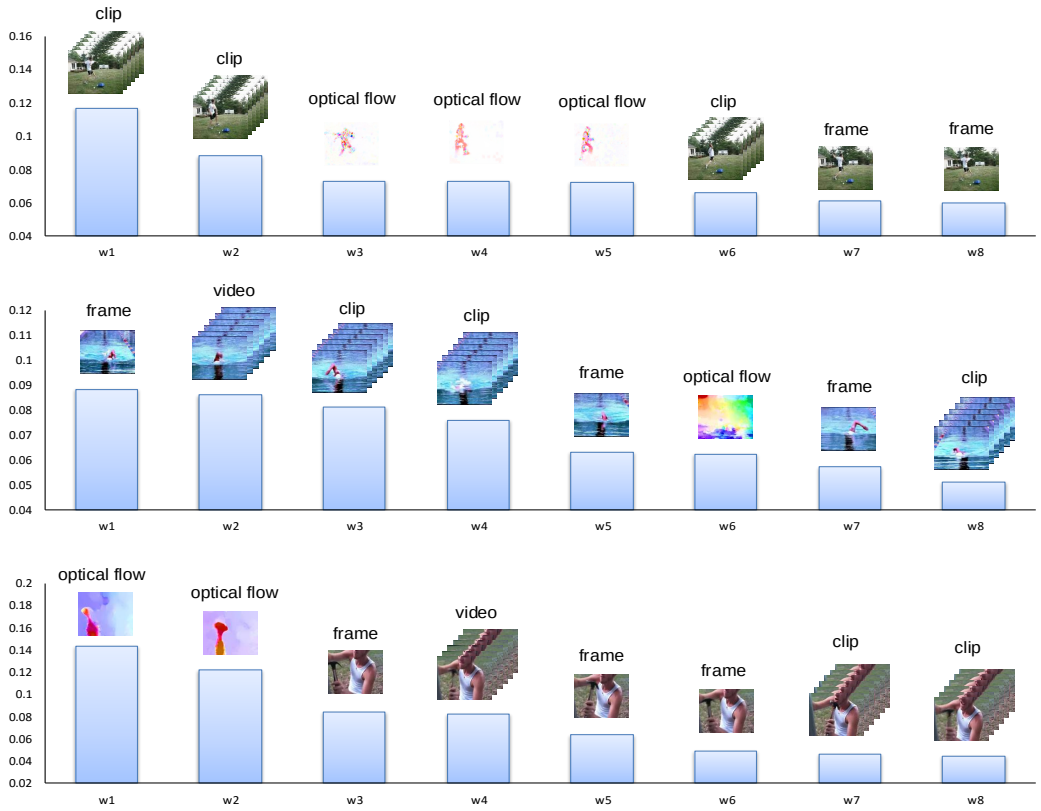

Fig. 5: Examples showing the top eight weights learned by the MSD and their corresponding components in a video (top: baseball pitch, middle: front crawl, bottom: hammering). We can see that MSD is able to learn the contributions from different components for particular actions. For example, two *clip* components play important roles for recognizing "baseball pitch," while two *motion* (optical flow) components contribute more to the recognition of "hammering."

6.5 Evaluation on CCV

Now we turn to evaluate our framework on the more challenging dataset CCV with longer video and more complicated categories. Table 5 shows the performance of each stream and the whole framework as well as the comparisons with the state-of-the-art results. As we can see, the performance of motion stream is much lower than that of frame stream, which is inconsistent with that of UCF101 and HMDB51. The reason is two-fold. On the one hand, many categories in CCV are of very high level and don't care much about the short-term motion in video, such as "graduation", "wedding reception" and "beach". On the other, since the average duration (around 80 seconds) of the videos of CCV is about 10 times longer than that of UCF101 and HMDB51 and the contents in CCV are more complex and noisy, the optical flows are easily disrupted by the big camera motion and useful motion information might be overwhelmed by many disrupted optical flows. When we comes to the fusion of all the streams, the performance of the whole framework is clearly better than each single stream, which verifies the effectiveness of the multi-granular score distribution fusion scheme. Table 5 also shows that our multi-granular framework is significantly better than all of the baselines, and we get around 10 percent performance gain over the best baseline [34].

To better understand the contribution of every stream in the multi-granular framework, we further report the performance on each class of CCV in Figure 6. From the Figure 6, *frame*, *clip* and *video* streams respectively get the best performance over different classes. For example, *clip* stream achieve the best performance on classes like "Basketball" and "Soccer", while the results of *frame* stream exceeds other streams a big margin on "Cat" class. This proves that different actions may span different granularities. And the multi-granular score fusion of all streams can significantly improve the performance for almost all the classes, although the performance of *motion* stream are relatively low.

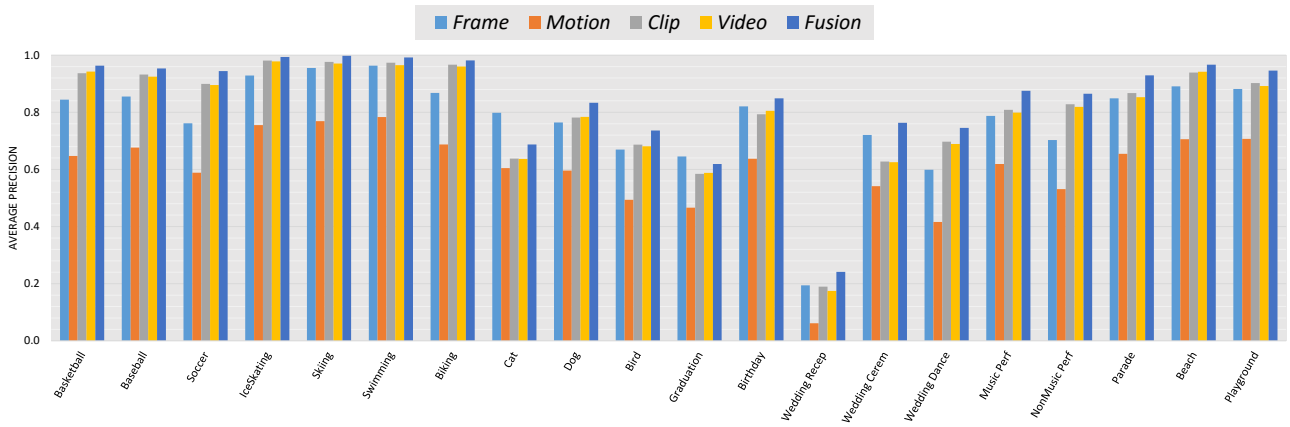

Fig. 6: Per-class performance on CCV, using *frame*, *motion*, *clip*, *video* and their *fusion*

Table 5: Comparisons with state-of-the-art results on CCV. The best performance is obtained by multi-granular score distribution fusion of our framework.

CCV			
Liu *et al.* [17]	68.2%	Ours (*frame*)	77.5%
Ye *et al.* [36]	64.0%	Ours (*motion*)	59.3%
Jhuo *et al.* [7]	64.0%	Ours (*clip*)	80.9%
Ma *et al.* [18]	63.4%	Ours (*video*)	80.1%
Wu *et al.* [34]	70.6%	Ours (*fusion*)	83.2%

Table 6: 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 Efficiency

In addition to obtaining the superior classification accuracy, our framework also enjoys high computational efficiency. Table 6 listed the details 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 response, the average prediction time of our multi-granular 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 for each video in UCF101.

7 Conclusions and Future Work

We have proposed a multi-stream deep learning framework to learn a hierarchical video representation, which can exploit the information in video from a multitude of granularities including *frame*, consecutive frames (*motion*), *clip* and the entire *video*. In the framework, we first apply two types of CNN at each granularity, i.e., 2D CNN on frame and motion streams, and 3D CNN on clip and video streams. The outputs of these CNNs are then used separately as inputs of the LSTM networks to model the long-term temporal dynamics. To verify the effectiveness of the proposed hierarchical video representation, we employ this representation for action recognition by integrating the information from all granularities with a novel score distribution fusion strategy. We conduct extensive experiments on two popular action recognition benchmarks with short videos (UCF101 and HMDB51) and another video classification dataset with complex and long videos (CCV). Our framework has achieved very impressive performance on all the three popular benchmarks. Results not only validate the effectiveness of each single stream, but also demonstrate that the multiple granularities of video are complementary and combining them can significantly boost the performance of action recognition.

There are several future directions. First, video action recognition can be enhanced by further considering audio information. The audio features can be exploited together with the current four streams to more comprehensively characterize the actions in videos. Second, the method for 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 action recognition performance.

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. A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.
5. M. Hoai and A. Zisserman. Improving human action recognition using score distribution and ranking. In *ACCV*, 2014.
6. S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
7. I.-H. Jhuo, G. Ye, S. Gao, D. Liu, Y.-G. Jiang, D. Lee, and S.-F. Chang. Discovering joint audio–visual codewords for video event detection. *Machine Vision and Applications*, 25(1):33–47, 2014.
8. 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.
9. 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.
10. Y.-G. Jiang, G. Ye, S.-F. Chang, D. Ellis, and A. C. Loui. Consumer video understanding: A benchmark database and an evaluation of human and machine performance. In *Proceedings of ACM International Conference on Multimedia Retrieval (ICMR), oral session*, 2011.
11. 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.
12. A. Kläser, M. Marszalek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008.
13. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.
14. H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011.
15. 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.
16. I. Laptev and T. Lindeberg. Space-time interest points. In *ICCV*, 2003.
17. W. Li, Z. Zhang, and Z. Liu. Expandable data-driven graphical modeling of human actions based on salient postures. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1499–1510, 2008.
18. A. J. Ma and P. C. Yuen. Reduced analytic dependency modeling: Robust fusion for visual recognition. *International Journal of Computer Vision*, 109(3):233–251, 2014.
19. 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.
20. Y. Pan, Y. Li, T. Yao, T. Mei, H. Li, and Y. Rui. Learning deep intrinsic video representation by exploring temporal coherence and graph structure.
21. 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.
22. P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM MM*, 2007.
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. I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
28. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.
29. 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.
30. H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.
31. L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015.
32. P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.
33. G. Willems, T. Tuytelaars, and L. J. V. Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *ECCV*, 2008.
34. 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 *Proceedings of the 22nd ACM international conference on Multimedia*, pages 167–176. ACM, 2014.
35. T. Yao, T. Mei, and Y. Rui. Highlight detection with pairwise deep ranking for first-person video summarization.
36. G. Ye, D. Liu, I.-H. Jhuo, and S.-F. Chang. Robust late fusion with rank minimization. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3021–3028. IEEE, 2012.
37. X. Yuan, W. Lai, T. Mei, X.-S. Hua, X.-Q. Wu, and S. Li. Automatic video genre categorization using hierarchical svm. In *2006 International Conference on Image Processing*, pages 2905–2908. IEEE, 2006.
38. W. Zaremba and I. Sutskever. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.
39. 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.