Action Recognition with Coarse-to-Fine
Deep Feature Integration and Asynchronous Fusion

Weiyao Lin,[1*] **Yang Mi,**[1] **Jianxin Wu,**[2] **Ke Lu,**[3] **Hongkai Xiong**[1]
[1] Department of Electronic Engineering, Shanghai Jiao Tong University, China
[2] National Key Laboratory for Novel Software Technology, Nanjing University, China
[3] University of Chinese Academy of Sciences, China
{wylin, deyangmiyang, xionghongkai}@sjtu.edu.cn, wujx2001@nju.edu.cn, luk@ucas.ac.cn

Abstract

Action recognition is an important yet challenging task in computer vision. In this paper, we propose a novel deep-based framework for action recognition, which improves the recognition accuracy by: 1) deriving more precise features for representing actions, and 2) reducing the asynchrony between different information streams. We first introduce a coarse-to-fine network which extracts shared deep features at different action class granularities and progressively integrates them to obtain a more accurate feature representation for input actions. We further introduce an asynchronous fusion network. It fuses information from different streams by asynchronously integrating stream-wise features at different time points, hence better leveraging the complementary information in different streams. Experimental results on action recognition benchmarks demonstrate that our approach achieves the state-of-the-art performance.

1 Introduction

Action recognition, which aims at identifying the action class label for an input action video, has attracted much attention due to its importance in many applications. Although the recent advances in deep convolutional networks (ConvNets) have brought some improvements on action recognition (Tran et al. 2015; Zhu et al. 2016), it remains challenging due to the large variation of video scenarios and the interferences from noisy contents unrelated to the video topic.

In this paper, we focus on two key issues for improving the performance over the existing ConvNet frameworks: (1) deriving more precise features to better represent actions, (2) reducing the asynchrony among information streams to better leverage the stream-wise complementary information.

First, good features are crucial to reliable action recognition. Although features automatically learned from ConvNets have shown big improvements in many domains (Liu et al. 2016; Deng et al. 2009; Song et al. 2017), they make
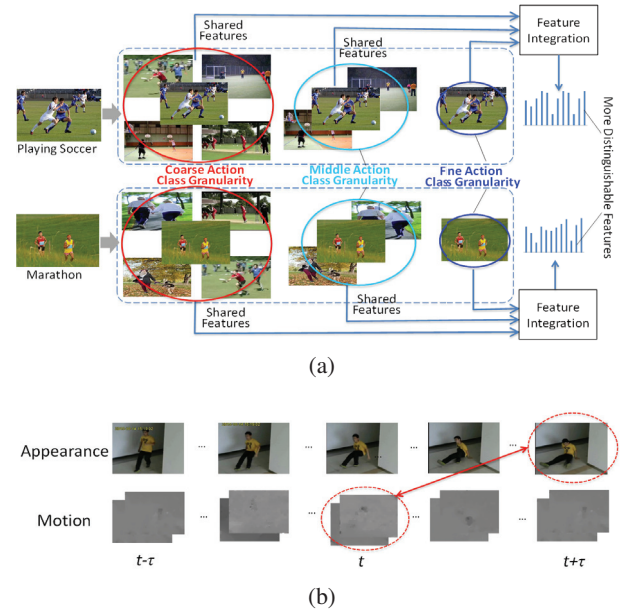
(a)

(b)

Figure 1: (a) Illustration of different action class granularity. (b) Illustration of the asynchronous pattern between streams: The appearance stream is most indicative about "fall down" after the object has lied down, while the motion stream shows the strongest "fall down" pattern when the object is in the process of going down. (Best viewed in color)

less progress in action recognition due to the high complexity of video data. Some recent studies attempted to improve the deep feature representation of an action by including additional information sources (Duta et al. 2017; Shi et al. 2017; Kataoka et al. 2016), selecting spatial-temporal attention parts (Kar et al. 2017; Sharma, Kiros, and Salakhutdinov 2015; Zhu et al. 2016), or incorporating more proper temporal information (Wang et al. 2016b; Cherian et al. 2017). However, since most of them focus on learning features to directly describe actions' individual action classes, they have limitations in precisely differentiating the ambiguity among action classes due to the large intra-class variations and subtle inter-class differences of actions.

In this paper, we introduce the idea of *action class granularity* where a coarser action class granularity includes more

action classes and a finer action class granularity contains fewer action classes. We argue that features learned for different action class granularities can provide useful information in discriminating action classes. For example, in Fig. 1a, since the input action clip "marathon" is visually similar to "playing soccer", they will be easily confused if directly deriving features to recognize their individual action classes. However, if we relax the recognition requirement from individual action classes to action class groups (i.e., coarser action class granularities), we are able to obtain shared features for representing a set of action classes. These shared features are able to provide more discriminative power as ambiguous action clips may correspond to different groups of action classes in coarser action class granularities (cf. Fig. 1a).

Based on this intuition, we propose a *coarse-to-fine network* which first extracts deep features from different action class granularities, and then progressively integrates them from coarse granularities to fine ones to obtain a precise feature representation for input actions (cf. Fig. 1a). It should be noted that since the action classes in each granularity are only used to derive proper features, they are automatically determined and dynamic for different input video clips.

Second, combining multiple information streams (such as two-stream ConvNets (Simonyan and Zisserman 2014)) has shown strong performance and thus has become a mainstream framework in action recognition. However, most existing works only focus on introducing more information streams (Shi et al. 2017; Kataoka et al. 2016) or strengthening the correlation among streams (Wang et al. 2016b; Wu et al. 2015; Sun et al. 2017), while the asynchronous issue among different information streams is less studied.

We argue that many actions have asynchronous patterns in different information streams, which affects the performance of action recognition. For example, Fig. 1b shows two information streams for an action clip "fall down" (one appearance stream and one motion stream). Apparently, the appearance stream shows the most indicative pattern about "fall down" after the object has lied down on the floor. Comparatively, the motion stream shows the strongest "fall down" pattern when the object is in the process of going down. If we simply combine the overall information in both streams or fuse the stream-wise information at the same time point, the indicative patterns appear at different time cannot be fully utilized and the performance is restrained. Therefore, we further introduce an *asynchronous fusion network*, which asynchronously integrates stream-wise features from different time points, hence better leveraging the complementary information in multiple streams.

Overall, our contribution to action recognition are 3 folds:

1. We propose a coarse-to-fine network which extracts and integrates deep features from multiple action class granularities to obtain a more precise representation for actions.

2. We propose an asynchronous fusion network which integrates stream-wise features at different time points for better leveraging the information in multi-streams.

3. We combine the proposed coarse-to-fine and asynchronous fusion networks into an integrated framework which achieves the state-of-the-art performance.

2 Related Works

Action recognition has been studied for years. Early works focus on developing good hand-crafted features for representing actions, such as 3D SIFT (Scovanner, Ali, and Shah 2007) and dense trajectory (Wang et al. 2013). The performances for these methods are often restrained due to the limited differentiation capability of hand-crafted features.

With the development of deep ConvNets, many ConvNet-based methods were recently proposed for action recognition, which utilize ConvNets to automatically obtain the feature representation for actions. Ji et al. (Ji et al. 2013) utilize a 3D ConvNet to recognize actions in video. Simonyan and Zisserman (Simonyan and Zisserman 2014) propose a two-stream framework which uses two ConvNets to respectively extract features from two information streams (i.e., appearance and motion) and fuse them for recognition. Based on this framework, recent researches further improve the effectiveness of ConvNet features by including additional information sources (Shi et al. 2017; Kataoka et al. 2016), selecting spatial-temporal attention parts (Kar et al. 2017; Sharma, Kiros, and Salakhutdinov 2015; Zhu et al. 2016), or incorporating more proper temporal information (Wang et al. 2016b; Wu et al. 2015; Cherian et al. 2017; Bilen et al. 2016).

Most of the existing works are targeted at learning features for directly describing actions' individual action classes, while the shared characteristics in different action class granularities are less studied. This restrains them from precisely distinguishing the subtle difference among ambiguous actions. Although some methods (Wu et al. 2016) obtain different levels of generality by integrating features in multi-ConvNet layers, they still focus on directly representing the individual action classes and do not consider the shared characteristics in different action class granularities.

Besides the derivation of proper features, other researches focus on the proper combination of multiple information streams to boost the action recognition performance (Feichtenhofer, Pinz, and Wildes 2016; Wu et al. 2015; Feichtenhofer, Pinz, and Zisserman 2016; Sun et al. 2017). For example, Feichtenhofer et al. (Feichtenhofer, Pinz, and Wildes 2016) introduce residual connections between information streams to remedy the deficiency of late fusion strategy in the two-stream framework. Wu et al. (Wu et al. 2015) also improve the fusion efficiency of the two-stream framework by performing both sequence level fusion and video-level fusion over the information streams. However, most of these works fuse stream-wise information that happen simultaneously, which have limitations in handling the longer-term asynchronous pattern among information streams. As will be shown in this paper, the asynchrony among information streams is a non-trivial factor which can bring noticeable performance gains for action recognition.

3 Overview

The framework of our approach is shown in Fig. 2. After obtaining appearance and motion streams from an input video, we first input each spatial frame from the appearance stream and each short-term optical flow stack from the motion stream into a coarse-to-fine network (detailed in
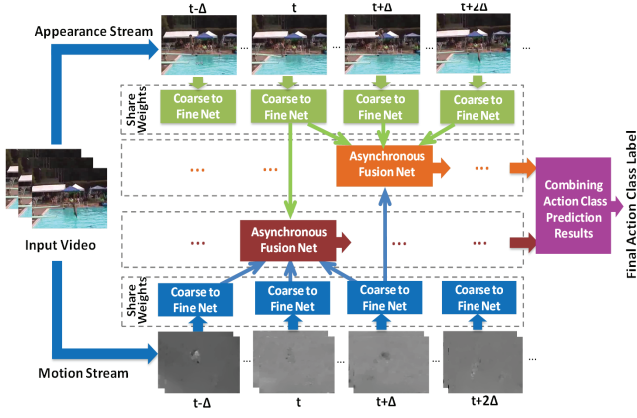
Figure 2: Framework of the approach. The coarse-to-fine network (detailed in Fig. 3) extracts a more precise feature representation for each frame/optical flow stack. These features are then fused by asynchronous fusion networks (detailed in Fig. 5) to obtain action prediction results.

Sec. 4), which integrates deep features from multiple action class granularities and creates a more precise feature representation. The extracted features are then fed into asynchronous fusion networks (detailed in Sec. 5), where each asynchronous fusion network integrates stream-wise features at different time points within a period and obtains an action class prediction result. Finally, action prediction results from different asynchronous fusion networks are combined to decide the final action class of the input video.

Note that the framework of our approach is integrated where the major components in the coarse-to-fine and asynchronous fusion networks can be jointly trained.

4 Coarse-to-Fine Network

The structure of the coarse-to-fine network is shown in Fig. 3. Basically, the network includes three major modules: first, a *multi-granularity feature extraction* module is applied over a ConvNet to extract deep features from different action class granularities. Second, in order to guarantee the extraction of proper features in the feature extraction module, an *adaptive class group forming* module is introduced. This module adaptively forms a suitable action class group for each action class granularity of an input frame/optical flow stack, so as to guide the feature extraction module to create the desired features. Third, a *coarse-to-fine integration* module is connected to the feature extraction module, which progressively integrates features from coarse action class granularities to fine ones and outputs a precise feature representation for the input frame/optical flow stack.

It should be noted that the *adaptive class group forming* module is only used in the training stage, while the *multi-granularity feature extraction* and *coarse-to-fine integration* modules are applied in both training and testing stages.

4.1 Multi-granularity feature extraction

The multi-granularity feature extraction module aims to extract deep features from different action class granularities.
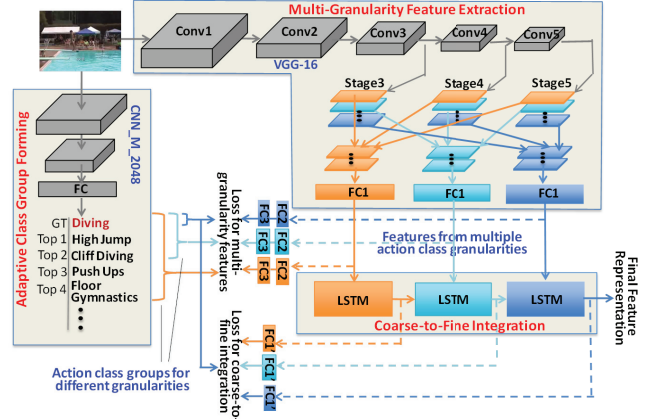

Figure 3: Structure of the coarse-to-fine network.

Since side output layers have shown their effectiveness in encoding multi-scale information in skeleton & boundary detection (Shen et al. 2017; Xie and Tu 2015), we borrow them into action recognition and construct three side output flows to extract features in three action class granularities.

Specifically, we derive side output maps from the last convolutional layer in stages 3, 4, and 5 of VGG-16 Conv-Net (i.e., conv3_3, conv4_3, conv5_3). The side output maps from different stages are then sliced and concatenated into three scale-specific side map groups (Shen et al. 2017), where each side map group corresponds to one action class granularity. In order to ensure output maps from different stages to have the same size, upsampling layers are applied on side output maps before map concatenating. Finally, the scale-specific side map groups are input into a fully connected (FC) layer respectively to obtain features for the three action class granularities (FC1 in Fig. 3). Note that different from the previous side output works (Shen et al. 2017; Xie and Tu 2015), our approach utilizes an FC layer in the side output flow to obtain features for describing actions.

4.2 Adaptive class group forming

The *adaptive class group forming* module is a key part of the coarse-to-fine network, which aims to form suitable action class groups to guide the feature extraction process in the *multi-granularity feature extraction* module. In this paper, we introduce an additional smaller ConvNet (i.e., CNN_M_2048 (Chatfield et al. 2014)) to form action class groups in different granularities.

Specifically, we first use the CNN_M_2048 ConvNet to predict the action class label of an input frame/optical flow stack, and then use the top 5, top 3, and top 1 action classes in the predicted result to form the action class groups in the three action class granularities, respectively.

Three important issues need to be mentioned about the *adaptive class group forming* module: (1) The *adaptive class group forming* module is only applied in the training stage which helps to construct a reliable multi-granularity feature extracion network. During the testing stage, the feature extraction module will directly output features without
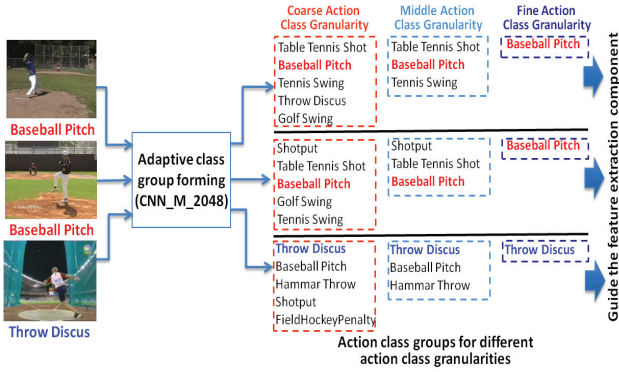
Figure 4: Examples of the adaptively formed action class groups for different inputs (best viewed in color).

the guidance of action class groups. (2) The CNN_M_2048 ConvNet is pre-trained on the same dataset and is fixed during the training process. We fix the CNN_M_2048 ConvNet in training in order to create stable action class groups. (3) When forming action class groups, if the groundtruth label of an input frame/optical flow stack is not listed in the top ranked action group in CNN_M_2048's prediction result, we will mandatorily include it into the action class group to avoid the feature extraction module deriving irrelevant features to the input frame/optical flow stack.

After action class groups are constructed in the *adaptive class group forming* module, they are used to guide the feature extraction process by a cross-entropy loss (De Boer et al. 2005), which forces the feature extraction module to create shared features that best describe the constructed action class groups in multiple granularities:

$$\mathcal{L}_v(\mathbf{W}) = -\frac{1}{N} \sum_{k=1}^{3} \sum_{n \in \mathbf{G}_k} \alpha_k \log \hat{p}(n|\mathbf{W}, k) \qquad (1)$$

where $\mathbf{W}$ is the parameter set for the *multi-granularity feature extraction* module. N is the total number of action classes. $\mathbf{G}_k$ is the constructed action class group for the kth action class granularity and α_k is the weight measuring the relative importance of the kth action class granularity. $\hat{p}(n|\mathbf{W}, k)$ is the probability for the nth action class predicted by the features from kth action class granularity. Note that in order to create action prediction results $\hat{p}(n|\mathbf{W}, k)$, two additional fully connected layers are added to the feature output layer of the *multi-granularity feature extraction* module in the training stage (FC2 & FC3 in Fig. 3).

We argue that by introducing the CNN_M_2048 ConvNet to form action class groups, we can have three advantages:

1. Since the CNN_M_2048 ConvNet is pre-trained on the same dataset, it has the capability to properly parse the action class contents of an input frame/optical flow stack. Thus, it is able to create informative action class groups which are relatively more similar for same-class inputs and less similar for different-class inputs (cf. the action class groups for the two "Baseball Pitching" inputs and the "Throw Discus" input in Fig. 4). Therefore, when

using these action class groups to guide the feature extraction process, we are able to obtain more distinguishable features. Note that the CNN_M_2048 ConvNet does not need to be perfectly trained. From our experiments, CNN_M_2048 roughly trained from partial data is already able to create good results (cf. Sec. 6.1).

2. Since the ground-truth action class of an input frame/optical flow stack is included in each of its action class groups (cf. the red and blue bold action classes in Fig. 4), features guided by these action class groups are able to capture the characteristics of the input's true action class in different aspects. Therefore, by integrating features from multiple action class granularities (cf. Sec. 4.3), the feature representation is properly strengthened, which has stronger capability to predict the correct action class for the input sample.

3. Moreover, by introducing CNN_M_2048 ConvNet into our coarse-to-fine network, we are also taking the advantage of properly combining two ConvNets (i.e., CNN_M_2048 and VGG-16) to boost action recognition performances. As will be shown in the experimental results, our approach provides a more proper way to combine ConvNets, which has obviously better performance than only using a single ConvNet or combining ConvNets in simpler ways (Xiao et al. 2015; Wang et al. 2016a).

4.3 Coarse-to-fine integration

After obtaining features from multiple action class granularities, we further utilize a *coarse-to-fine integration* module to progressively integrates features from different action class granularities and outputs a precise feature representation. In this paper, we utilize a Long Short Term Memory (LSTM) network to perform coarse-to-fine integration due to its effectiveness in fusing sequential inputs (Donahue et al. 2015; Graves, Mohamed, and Hinton 2013).

Specifically, we utilize an LSTM model with three units, where each unit takes features $\mathbf{x}_t$ from one action class granularity and creates hidden state outputs $\mathbf{h}_t$ to influence the next unit (cf. Fig. 3). The hidden state output from the last unit will be the final integrated feature for the input frame/optical flow stack. The entire process is described by:

$$\begin{aligned} \mathbf{h}_1 &= F_{\mathbf{\Phi}_1}(\mathbf{x}_1, 0) \\ \mathbf{h}_2 &= F_{\mathbf{\Phi}_2}(\mathbf{x}_2, \mathbf{h}_1) \\ \mathbf{h}_3 &= F_{\mathbf{\Phi}_3}(\mathbf{x}_3, \mathbf{h}_2) \end{aligned} \qquad (2)$$

where $\mathbf{x}_t$ and $\mathbf{h}_t$ ($t = 1, 2, 3$) are the input features and hidden state results for tth LSTM unit. $\mathbf{\Phi}_t = \{\mathbf{M}_t, \mathbf{b}_t\}$ is the parameter set for tth unit and $F_{\mathbf{\Phi}_t}$ is the operation of tth unit to create hidden state outputs (Donahue et al. 2015).

In the training stage, we utilize the following loss function to train LSTM model to create the desired results.

$$\begin{aligned} \mathcal{L}_l(\mathbf{\Phi}_1, \mathbf{\Phi}_2, \mathbf{\Phi}_3) = -\frac{\beta}{N}(&\log \hat{p}(n_g|\mathbf{\Phi}_1) \\ + \log \hat{p}(n_g|\mathbf{\Phi}_1, \mathbf{\Phi}_2) &+ \log \hat{p}(n_g|\mathbf{\Phi}_1, \mathbf{\Phi}_2, \mathbf{\Phi}_3)) \end{aligned} \qquad (3)$$

where $\mathbf{\Phi}_1, \mathbf{\Phi}_2, \mathbf{\Phi}_3$ are the parameter sets for the three units in LSTM. β is the weight measuring the relative importance
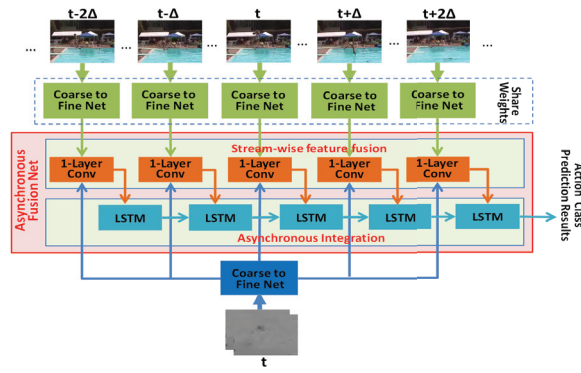
Figure 5: Structure of asynchronous fusion network and its relation with coarse-to-fine networks.

of the LSTM model. n_g is the ground-truth action class label for an input sample. N is the total number of action classes. $\hat{p}(n_g|\boldsymbol{\Phi}_1..\boldsymbol{\Phi}_t)$ is the predicted probability for the ground-truth class from the tth unit. Similar to Eq. 1, in order to create action prediction probability $\hat{p}(n_g|\boldsymbol{\Phi}_1..\boldsymbol{\Phi}_t)$, an additional fully connected layer is added to the output of each LSTM unit in the training stage (cf. FC1′ in Fig. 3).

4.4 Loss function for coarse-to-fine network

The loss function for the coarse-to-fine network is shown by:

$$\mathcal{L}_{\mathcal{C}}(\boldsymbol{\Psi}_{\mathcal{C}}) = \mathcal{L}_v(\mathbf{W}) + \mathcal{L}_l(\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, \boldsymbol{\Phi}_3) \quad (4)$$

where $\mathcal{L}_v(\mathbf{W})$ and $\mathcal{L}_l(\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, \boldsymbol{\Phi}_3)$ are the losses for the *multi-granularity feature extraction* and *coarse-to-fine integration* modules, respectively. $\boldsymbol{\Psi}_{\mathcal{C}} = \{\mathbf{W}, \boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, \boldsymbol{\Phi}_3\}$ is the parameter set for the entire coarse-to-fine network.

Note that the coarse-to-fine network can be jointly trained with the asynchronous fusion network in our approach. Therefore, Eq. 4 can be further combined with the loss of the asynchronous fusion network to construct a final loss function for the entire approach, as will be discussed in Sec. 5.

5 Asynchronous Fusion Network

The structure of the asynchronous fusion network is shown in Fig. 5. Basically, the asynchronous fusion network aims to fuse an input feature at time t in one stream with multiple input features around t in another stream, so as to leverage the stream-wise complementary information at different time points. It mainly includes two modules: First, the *stream-wise feature fusion* module is used to fuse two input features from different streams. Second, the *asynchronous integration* module is used to integrate the fused outputs over different time and create an action class prediction result for the period of the input features.

5.1 Stream-wise feature fusion

Since inputs from different information streams have different characteristics, simply concatenating them may create less effective fusion results. Therefore, we utilize a Conv-Net to fuse features from different streams due to its effectiveness in fusing multi-stream inputs (Feichtenhofer,

Pinz, and Zisserman 2016). Since input features are only 1-dimensional vectors, we simply view them as two 1-dimensional feature maps and apply a single layer ConvNet with 1×1 kernel to create the fused output.

Note that: (1) In our asynchronous fusion network, an input feature in one stream is fused with 5 input features from another stream. Therefore, five 1-layer ConvNets are used to fuse stream-wise features (cf. Fig. 5). (2) Moreover, the five input features to be fused also have Δ ($\Delta = 5$) time intervals to each other. This enables us to capture the longer-term asynchronous patterns between streams.

5.2 Asynchronous integration

After obtaining stream-wise fusion results with different time intervals, the *asynchronous integration* module will sequentially integrate them and create an action prediction result for the period of the input features. In this paper, we utilize a five-unit LSTM to perform integration (cf. Fig. 5) since it has good capability in integrating sequential inputs (Donahue et al. 2015).

5.3 Loss function for the asynchronous fusion network & the entire framework

The entire asynchronous fusion network can be trained by:

$$\mathcal{L}_{\mathcal{A}}(\boldsymbol{\Psi}_{\mathcal{A}}) = -\frac{\gamma}{N} \sum_{t=1}^{T} \log \hat{p}(n_g|\boldsymbol{\Phi}_1, .., \boldsymbol{\Phi}_t, \mathbf{K}_1, .., \mathbf{K}_t) \quad (5)$$

where N is the total number of action classes. n_g is the ground-truth class label of input video. $T = 5$ is the total number of LSTM units and 1-layer ConvNets. $\boldsymbol{\Phi}_t$ and $\mathbf{K}_t$ are the parameter sets for the tth LSTM unit and tth 1-layer ConvNet, respectively. $\boldsymbol{\Psi}_{\mathcal{A}} = \{\boldsymbol{\Phi}_1, ..., \boldsymbol{\Phi}_T, \mathbf{K}_1, ..., \mathbf{K}_T\}$ and γ are the parameter set and weight for the entire asynchronous fusion network, respectively. $\hat{p}(n_g|\boldsymbol{\Phi}_1, ..., \boldsymbol{\Phi}_t, \mathbf{K}_1, ..., \mathbf{K}_t)$ is the predicted probability for the ground-truth class from the tth LSTM unit.

Moreover, the asynchronous fusion network can be jointly trained with the coarse-to-fine network by combining their loss functions. Therefore, the overall framework of our approach can be trained by:

$$(\boldsymbol{\Psi}_{\mathcal{C},s_1}, \boldsymbol{\Psi}_{\mathcal{C},s_2}, \boldsymbol{\Psi}_{\mathcal{A}})^* =$$
$$\arg\min \left(\sum_{t=1}^{T} \mathcal{L}_{\mathcal{C}}^t(\boldsymbol{\Psi}_{\mathcal{C},s_1}) + \mathcal{L}_{\mathcal{C}}(\boldsymbol{\Psi}_{\mathcal{C},s_2}) + \mathcal{L}_{\mathcal{A}}(\boldsymbol{\Psi}_{\mathcal{A}}) \right) \quad (6)$$

where $\boldsymbol{\Psi}_{\mathcal{C},s_1}$, $\boldsymbol{\Psi}_{\mathcal{C},s_2}$ are the parameter sets of the coarse-to-fine networks for the first and second information streams. $\boldsymbol{\Psi}_{\mathcal{A}}$ is the parameter set of the asynchronous fusion network. $\mathcal{L}_{\mathcal{C}}(\cdot)$ and $\mathcal{L}_{\mathcal{A}}(\cdot)$ are the loss functions of the coarse-to-fine and asynchronous fusion networks (cf. Eqs. 4 and 5). $T = 5$ is the total number of inputs in the first stream (cf. Fig. 5). Note that since the five coarse-to-fine networks in the first stream share weights, we use the same parameter set $\boldsymbol{\Psi}_{\mathcal{C},s_1}$ to calculate the loss of each input $\mathcal{L}_{\mathcal{C}}^t(\boldsymbol{\Psi}_{\mathcal{C},s_1})$, $t = 1, ..., 5$.

Besides, it should also be noted that our approach actually requires to construct two independent models, where

one model fuses an appearance-stream input with multiple motion-stream inputs, and another model fuses a motion-stream input with multiple appearance-stream inputs. The action prediction results from both models and at different time periods are then combined to decide the final label of an input video (cf. Fig. 2). In this paper, we follow the mainstream two-stream methods (Wang et al. 2016b) to combine action prediction results, which adds the action prediction results from different models & periods and selects the class with the largest overall prediction score as the final result.

6 Experimental Results

6.1 Datasets & experimental settings

Datasets. We perform experiments on two benchmark datasets: UCF101 (Soomro, Zamir, and Shah 2012) and HMDB51 (Kuehne et al. 2011). UCF101 dataset is a commonly used dataset for action recognition. It contains $13,320$ video clips in 101 action classes. HMDB51 dataset is a large collection of realistic videos, which contains $6,766$ video clips in 51 action classes.

Experimental settings. We implement our approach on Caffe (Jia et al. 2014). The batch size and momentum are set to be 16 and 0.9, respectively. The weight parameters for different granularities in the coarse-to-fine network ($\alpha_1, \alpha_2, \alpha_3$ in Eq. 1) are set to be $0.1, 0.1, 1$ respectively. Besides, the weight parameters for the LSTM models in the coarse-to-fine and asynchronous fusion networks (β and γ in Eqs. 3 and 5) are set as 2 to let the networks focus more on the reliability on their final outputs.

We use the same method as (Wang et al. 2016b; Wang, Qiao, and Tang 2015; Wang, Farhadi, and Gupta 2016) to construct optical flow stacks and perform data augmentation. Moreover, the VGG-16 models in both appearance and motion streams are initialized with a pre-trained model from ImageNet (Deng et al. 2009). When training the entire framework, we set the initial learning rate as 10^{-2} and is decreased to its $1/10$ for every 20K iterations. The maximum iteration is 100K. Besides, the CNN_M_2048 ConvNet used to construct action class groups are trained with $1/8$ of the training data and 10K iterations.

During evaluation, we sample 12 periods from each video, where each period includes 5 frames and 5 corresponding optical flow stacks with temporal distance $\Delta = 5$ (cf. Fig. 5). The prediction results from these periods are combined to obtain the final result.

6.2 Results for the coarse-to-fine network

In order to evaluate the effectiveness of our coarse-to-fine network, we compare six methods: (1) The standard two-stream approach (Simonyan and Zisserman 2014) (*Two-stream baseline*). (2) Combining 2 two-stream networks (VGG-16 and CNN_M_2048) by fusing their fully connected layers for recognition (Xiao et al. 2015; Wang et al. 2016a) (*Direct combine ConvNets*). (3) Delete the *adaptive class group forming* module and only use the loss function in Eq. 3 to train the coarse-to-fine network (*CO2FI-no class grouping*). (4) Delete the coarsest action class granularity and only

Table 1: Results of coarse-to-fine network (split1)

	Methods	Appearance	Motion	2-stream
UCF101	Two-stream baseline	79.2%	84.8%	89.8%
	Direct combine ConvNets	80.1%	85.4%	90.6%
	CO2FI-no class grouping	79.1%	85.2%	90.0%
	CO2FI-two granularities	81.0%	86.9%	91.7%
	CO2FI-no coarseness	79.5%	85.4%	90.4%
	CO2FI-complete	**81.7%**	**87.9%**	**92.8%**
HMDB51	Two-stream baseline	48.1%	55.4%	58.4%
	CO2FI-complete	**55.5%**	**63.0%**	**67.9%**

use the two finer action class granularities in our coarse-to-fine network for recognition (*CO2FI-two granularities*). (5) Use three action class granularities in the coarse-to-fine network, but each granularity only contains a single ground-truth action class (*CO2FI-no coarseness*). (6) The complete version of our coarse-to-fine network (*CO2FI-complete*).

Table 1 compares the action recognition results on split 1 of UCF101 and HMDB51 datasets, where the mean classification accuracy for appearance stream, motion stream, and two-streams are listed. Note that in order to delete the effect of the asynchronous fusion network in this experiment, we directly add a softmax layer after the coarse-to-fine network to obtain recognition results. From Table 1, we can observe:

(1) The performance of the *CO2FI-no class grouping* method is similar to *two-stream baseline* and is obviously lower than the complete version of our approach (*CO2FI-complete*). This implies that without the guidance of the *adaptive class group forming* module, the coarse-to-fine network will construct less precise features and bring few improvements. Besides, the *Direct combine ConvNets* method also achieves less obvious improvements. This further indicates that satisfactory results cannot be easily obtained without a proper way to combine ConvNets.

(2) Comparing the *CO2FI-no coarseness* method with the *CO2FI-two granularities* method, we can see that less noticeable improvements are obtained if each action class granularity only contains one ground-truth action class (*CO2FI-no coarseness*). Comparatively, when each action class granularity includes more action classes, more obvious improvements are achieved with only two action class granularities (*CO2FI-two granularities*). This indicates that the shared characteristics from multiple action classes are the key parts to improve feature representations, and the improvements are restrained if these shared characteristics cannot be obtained (as in *CO2FI-no coarseness*).

(3) The complete version of our coarse-to-fine network (*CO2FI-complete*), which obtains features by including more action class granularities with different coarseness, has the largest improvement over the baseline. This further demonstrates the effectiveness of our approach.

6.3 Results for the asynchronous fusion network

Table 2 shows the performance of our asynchronous fusion network. In Table 2, the upper part shows the results by applying the fusion network on the baseline two-stream ConvNet (i.e., *Baseline+SYN*, *Baseline+ASYN*), and the lower part shows the results by combining our fusion network with

Table 2: Results of asynchronous fusion network (split 1)

Methods	UCF101	HMDB51
Two-stream baseline	89.8%	58.4%
Baseline+SYN	89.7%	–
Baseline+ASYN ($\Delta = 1$)	90.3%	–
Baseline+ASYN ($\Delta = 5$)	**91.0%**	**60.9%**
CO2FI	92.8%	67.9%
CO2FI+ASYN ($\Delta = 5$)	**93.7%**	**69.5%**

Figure 6: An example of the effect of the asynchronous fusion network: since two streams create high prediction scores for the ground-truth class "Highjump" at different time points, the recognition result will be easily confused if not considering the stream-wise asynchronous pattern.

the coarse-to-fine network (*CO2FI+ASYN*). Moreover, *SYN* refers to the method that fuses two stream-wise features at the same time point. *ASYN ($\Delta = 1$)* and *ASYN ($\Delta = 5$)* mean using our asynchronous fusion network to fuse stream-wise features, where the temporal distances of input features being fused are 1 and 5 (cf. Fig. 5).

From the upper part of Table 2, we can see that simply fusing features at the same time point brings no improvements (*Baseline+SYN*). When we only fuse stream-wise features that are temporally close to each other (*Baseline+ASYN ($\Delta = 1$)*), the improvements are still less obvious since the longer-term asynchronous patterns are not properly captured. Comparatively, when fusing stream-wise features with larger temporal distances (*Baseline+ASYN ($\Delta = 5$)*), we can obtain more noticeable improvements. This demonstrates that the asynchrony between different information streams indeed affects action recognition performances. Moreover, from the lower part of Table 2, we can also observe that when combining our asynchronous fusion network with the coarse-to-fine network, we can obtain further improved recognition performances by leveraging both mutli-granularity features and stream-wise complementary information (*CO2FI+ASYN ($\Delta = 5$)*).

Fig. 6 further shows an example about the effect of the asynchronous fusion network. In Fig. 6, since the two information streams of the "Highjump" video have asynchronous patterns, they create high prediction scores for the ground-truth action class at different time points (e.g., t_3 for appearance stream and t_2 for motion stream in Fig. 6). If we simply sum up the prediction scores over time or only consider the stream-wise correlation at the same time, the final recognition result will be confused with other action classes (cf. *Overall score w/o ASYN*). Comparatively, if we consider the asynchrony between streams and allow stream-wise feature fusion at different time, the complementary information between streams can be more properly used, resulting in a correct result (cf. *Overall score with ASYN* in Fig. 6).

6.4 Comparison with the state-of-the-art

Table 3 compares our approach (*CO2FI+ASYN*) with the state-of-the-art methods. Since many works reported results by performing a late fusion with hand-crafted IDT features (Wang et al. 2013), we also show fusion result of our approach (*CO2FI+ASYN+IDT*). Note that in this experiment, we adopt three training/testing splits on both datasets in order to have a fair comparison with other methods.

From Table 3, we can see that our approach has better performances than most of the state-of-the-art methods. Specifically, when comparing with the most recent works using
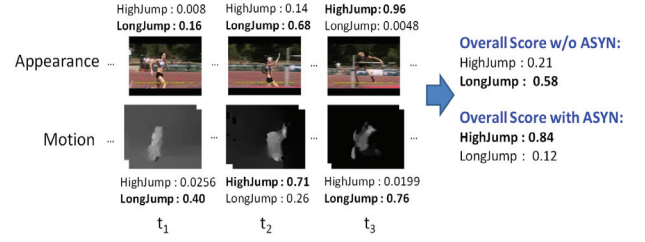
Table 3: Comparison of different methods (3 splits)

Methods	UCF101	HMDB51
C3D (3 nets) [Tran *et al.* 2015]	85.2%	–
AdaScan [Kar *et al.* 2017]	89.4%	54.9%
TDD+FV [Wang *et al.* 2015]	90.3%	63.2%
GRP [Cherian *et al.* 2017]	91.9%	65.4%
Three-stream sDTD [Shi *et al.* 2017]	92.2%	65.2%
Transformations [Wang *et al.* 2016]	92.4%	62.0%
Two-Stream Fusion [Feichtenhofer *et al.* 2016]	92.5%	65.4%
KVMF [Zhu *et al.* 2016]	93.1%	63.3%
ST-ResNet [Feichtenhofer *et al.* 2016]	93.4%	66.4%
L^{2}STM [Sun *et al.* 2017]	93.6%	66.2%
ST-VLMPF [Duta *et al.* 2017]	93.6%	**69.5%**
TSN (2 modelities) [Wang *et al.* 2016b]	94.0%	68.5%
CO2FI + ASYN	**94.3%**	69.0%
Dynamic Image Networks + IDT [Bilen *et al.* 2016]	89.1%	65.2%
AdaScan + IDT [Kar *et al.* 2017]	91.3%	61.0%
TDD + IDT [Wang *et al.* 2015]	91.5%	65.9%
GRP + IDT [Cherian *et al.* 2017]	92.3%	67.0%
ST-ResNet + IDT [Feichtenhofer *et al.* 2016]	94.6%	70.3%
CO2FI + ASYN + IDT	**95.2%**	**72.6%**

ResNet (*ST-ResNet*) or introducing an additional information stream (*ST-VLMPF*), our approach can also obtain similar or better results. This demonstrates the effectiveness of our proposed approach. Note that comparing with *ST-ResNet* and *ST-VLMPF*, we use a relatively short ConvNet (VGG-16) and do not introduce additional information streams. It is expected that the performances of our approach can be further improved if using deeper ConvNets such as ResNet or including more information streams. Moreover, our approach fused with IDT (*CO2FI+ASYN+IDT*) also performs better than other IDT-fused methods. This furhter indicates the robustness of our approach in improving performances.

7 Conclusion

This paper presents a novel framework for action recognition. Our framework consists of two key ingredients: 1) a coarse-to-fine network, which extracts and integrates deep features from multiple action class granularities to obtain a more precise feature representation for actions; 2) an asynchronous fusion network which integrates stream-wise features at different time points for better leveraging the infor-

mation in multiple streams. Experimental results show that our approach achieves the state-of-the-art performance.

References

Bilen, H.; Fernando, B.; Gavves, E.; Vedaldi, A.; and Gould, S. 2016. Dynamic image networks for action recognition. In *CVPR*.

Chatfield, K.; Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2014. Return of the devil in the details: Delving deepinto convolutional nets. In *CoRR abs/1405.3531*.

Cherian, A.; Fernando, B.; Harandi, M.; and Gould, S. 2017. Generalized rank pooling for activity recognition. In *CVPR*.

De Boer, P.; Kroese, D.; Mannor, S.; and Rubinstein, R. 2005. A tutorial on the cross-entropy method. *Annals of Operations Research* 134(1):19–67.

Deng, J.; Dong, W.; Socher, R.; Li, L.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*.

Donahue, J.; Anne Hendricks, L.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S.; Saenko, K.; and Darrell, T. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*.

Duta, I. C.; Ionescu, B.; Aizawa, K.; and Sebe, N. 2017. Spatio-temporal vector of locally max pooled features for action recognition in videos. In *CVPR*.

Feichtenhofer, C.; Pinz, A.; and Wildes, R. 2016. Spatiotemporal residual networks for video action recognition. In *NIPS*.

Feichtenhofer, C.; Pinz, A.; and Zisserman, A. 2016. Convolutional two-stream network fusion for video action recognition. In *CVPR*.

Graves, A.; Mohamed, A. R.; and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In *ICASSP*.

Ji, S.; Xu, W.; Yang, M.; and Yu, K. 2013. 3d convolutional neural networks for human action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 35(1):221–231.

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; and Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*.

Kar, A.; Rai, N.; Sikka, K.; and Sharma, G. 2017. Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos. In *CVPR*.

Kataoka, H.; He, Y.; Shirakabe, S.; and Satoh, Y. 2016. Motion representation with acceleration images. In *ECCVW*.

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

Liu, J.; Gao, C.; Meng, D.; and Zuo, W. 2016. Two-stream contextualized cnn for fine-grained image classification. In *AAAI*.

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

Sharma, S.; Kiros, R.; and Salakhutdinov, R. 2015. Action recognition using visual attention. In *CoRR abs/1511.04119*.

Shen, W.; Zhao, K.; Jiang, Y.; Wang, Y.; Bai, X.; and Yuille, A. 2017. Deepskeleton: Learning multi-task scale-associated deep side outputs for object skeleton extraction in natural images. *IEEE Trans. Image Processing*.

Shi, Y.; Tian, Y.; Wang, Y.; and Huang, T. 2017. Sequential deep trajectory descriptor for action recognition with three-stream CNN. *IEEE Trans. Multimedia*.

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

Song, S.; Lan, C.; Xing, J.; Zeng, W.; and Liu, J. 2017. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *AAAI*.

Soomro, K.; Zamir, A. R.; and Shah, M. 2012. UCF101: a dataset of 101 human actions classes from videos in the wild. In *CoRR abs/1212.0402*.

Sun, L.; Jia, K.; Chen, K.; Yeung, D. Y.; Shi, B. E.; and Savarese, S. 2017. Lattice long short-term memory for human action recognition. In *ICCV*.

Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; and Paluri, M. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*.

Wang, H.; Klaser, A.; Schmid, C.; and Liu, C. 2013. Dense trajectories and motion boundary descriptors for action recognition. *Intl. Journal Comp. Vision* 103(1):60–79.

Wang, J.; Wei, Z.; Zhang, T.; and Zeng, W. 2016a. Deeply fused nets. In *CoRR abs/1605.07716*.

Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; and Van Gool, L. 2016b. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*.

Wang, X.; Farhadi, A.; and Gupta, A. 2016. Actions transformations. In *CVPR*.

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

Wu, Z.; Wang, X.; Jiang, Y.; Ye, H.; and Xue, X. 2015. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *ACM MM*.

Wu, J.; Wang, G.; Yang, W.; and Ji, X. 2016. Action recognition with joint attention on multi-level deep features. In *CoRR abs/1607.02556*.

Xiao, T.; Xu, Y.; Yang, K.; Zhang, J.; Peng, Y.; and Zhang, Z. 2015. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *CVPR*.

Xie, S., and Tu, Z. 2015. Holistically-nested edge detection. In *ICCV*.

Zhu, W.; Hu, J.; Sun, G.; Cao, X.; and Qiao, Y. 2016. A key volume mining deep framework for action recognition. In *CVPR*.