Ordered Pooling of Optical Flow Sequences for Action Recognition

Jue Wang[1,3] Anoop Cherian[2,3] Fatih Porikli[1,2,3]
[1]Data61/CSIRO, [2] Australian Center for Robotic Vision
[3] Australian National University, Canberra, Australia
jue.wang@anu.edu.au anoop.cherian@anu.edu.au fatih.porikli@anu.edu.au

Abstract

Training of Convolutional Neural Networks (CNNs) on long video sequences is computationally expensive due to the substantial memory requirements and the massive number of parameters that deep architectures demand. Early fusion of video frames is thus a standard technique, in which several consecutive frames are first agglomerated into a compact representation, and then fed into the CNN as an input sample. For this purpose, a summarization approach that represents a set of consecutive RGB frames by a single dynamic image to capture pixel dynamics is proposed recently. In this paper, we introduce a novel ordered representation of consecutive optical flow frames as an alternative and argue that this representation captures the action dynamics more effectively than RGB frames. We provide intuitions on why such a representation is better for action recognition. We validate our claims on standard benchmark datasets and demonstrate that using summaries of flow images lead to significant improvements over RGB frames while achieving accuracy comparable to the state-of-the-art on UCF101 and HMDB datasets.

1. Introduction

Automatically recognizing human actions in videos is a challenging task since actions in real-world are often very complex, may involve hard to detect objects and tools, may have different temporal speeds, can be contaminated by action clutter, or the same action can vary significantly from one actor to another, among several other factors. Nevertheless, developing efficient solutions for this problem could facilitate and nurture many applications, including visual surveillance, augmented reality, video summarization, and human-robot interaction. The recent resurgence of deep learning has demonstrated a significant promise in ameliorating the difficulties in recognizing actions. However, such solutions are still far from being practically useful, and thus this problem continues to be a popular research topic in computer vision [1, 7, 21, 23, 28, 29].

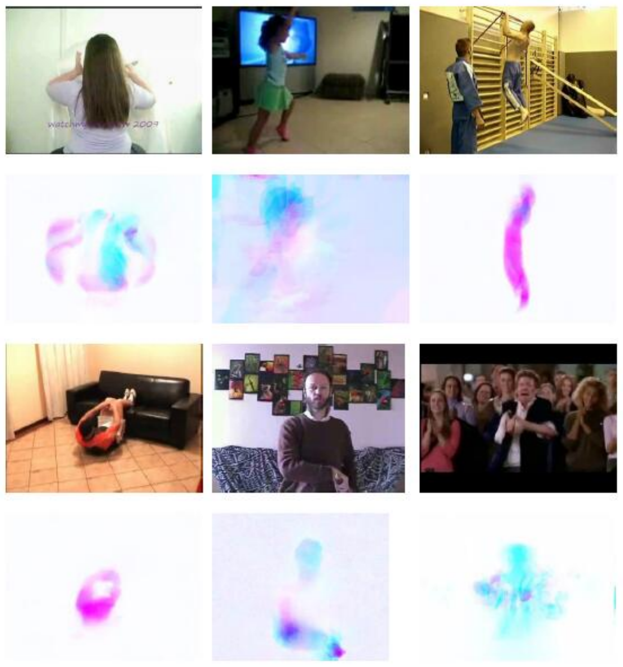
Figure 1. Examples of dynamic flow image and related RGB video frames. From left to right and top to bottom, the action classes are: "brush hair", "cart wheel", "pull up", "sit up", "draw sword" and "clip". Each dynamic flow image is a two channel image that compactly summarizes a sequence of optical flow frames from a video sequence.

Usually, deep architectures for video based action recognition take as input short video clips consisting of one to a few tens of frames. Using longer subsequences would require deeper networks or involve a huge number of parameters, which might not fit in the GPU memory, or may be problematic to train due to computational complexity. This restriction and thus clipping of the temporal receptive fields of the videos to short durations prohibit CNNs from learning long-term temporal evolution of the actions, which is very important in recognition especially when the actions are complex.

One standard way to tackle this difficulty in capturing long-term dependencies is to use temporal pooling that can be applied either when providing the input to the CNN or after extracting features from intermediate CNN layers [4].

In this paper, we explore a recently introduced early fusion scheme based on a Ranking SVM [10] formulation where several consecutive RGB frames in the video are fused to generate one “dynamic image” by minimizing a quadratic objective with temporal order constraints. The main idea of this scheme is that the parameters learned after solving this formulation capture the temporal ordering of the pixels from frame-to-frame, thus summarizing the underlying action dynamics.

One drawback of the rank pooling approach is that the formulation does not directly capture the motion associated with the action – it only captures how the pixel RGB values change from frame-to-frame. Typically, the video pixels can be noisy. Given that the pooling constraints in this scheme only look at increasing pixel intensities from frame-to-frame, it can fit to noise pixels that adhere to this order, however, are unrelated to the action. To mitigate these issues, in this paper, we look at the rank pooling formulation in the context of optical flow images instead of RGB frames.

It is clear that optical flow can easily circumvent the above problems associated with sequences of RGB frames. The flow by itself captures the objects in motion in the videos, and thereby capturing the action dynamics directly, while thresholding the flow vectors helps to avoid noise. Thus, we posit that summarizing sequences of optical flow can be significantly more beneficial than using RGB frames for action recognition. By solving the rank-SVM formulation (Section 3), we generate flow images that summarize the action dynamics, dubbed *dynamic flow images*. These images are then used as input to a standard action recognition CNN architecture to train for the actions in the sequences. In Figure 1, we show a few sample dynamic flow images generated using the proposed technique for the respective RGB frames.

A natural question here can be regarding the intuitive benefit of using such a flow summarization scheme, given that standard action recognition CNN frameworks already use a stack of flow frames. Note that such flow stacks usually use only a few frames (usually 10), while using the dynamic flow images, we summarize several tens of flow frames, thereby capturing long-term temporal context.

To validate our claims, we provide extensive experimental comparisons (Section 4) on two standard benchmark action recognition datasets, namely (i) the HMDB-51 dataset, and (ii) the UCF-101 dataset. Our results show that using dynamic flow images lead to significant improvements in action recognition performance. We find that this leads to 4% improvement on HMDB-51[16] and 6% on the UCF-101[25] dataset in comparison to using dynamic RGB images without combining any other methods.

Before moving on to explaining our scheme in detail, we summarize below our important contributions.

1. We provide an efficient and powerful video representation, *dynamic flow image*, generated based on the Ranking SVM formulation. Our representation aggregates local action dynamics (as captured by optical flow) over subseqeunces while preserving the temporal evolution of these dynamics.

2. We provide an action recognition framework based on the popular two-stream network [23], that also combines dynamic flow images, RGB frames (for action context), and hand-crafted trajectory features.

3. We provide extensive experimental comparisons demonstrating the effectiveness of our scheme on two challenging benchmark datasets.

2. Related Work

Extracting discriminative features from videos is the first and most important step in action recognition. Such features can be roughly divided into two groups: handcrafted local features and deep learned features. Of course, there are also many interesting methods that are not limited to these, such as using genetic programming to learn spatial-temporal features automatically [19], using human shape evolution of skeletons [2], and so on. However, we will limit our survey to the techniques most related to the approach presented in this paper.

Handcrafted Local Features. Using local features to generate representations for action recognition is popular because they make the recognition robust to noise, background motion, or illumination changes. One noteworthy such representation is described in [28], where dense trajectories, that capture the dynamics of actions, is used to define regions of interest in the video. Local features (such as HOG, HOF, MBH, etc.), that directly relate to the action can then be extracted from these regions to train classifiers. There have been extensions of this approach to also include semantics of the action, such as using Fisher vectors [22, 31], using bag of visual words model [20], combining with depth data [18], or using human pose [14]. However, the recent trend is towards automatically learning useful features in a data-driven way via convolutional neural networks [4, 7, 29], and we follow this trend in this paper.

Deep Learned Features. Deep learning methods have been extensively used for computer vision tasks since the work of Krizhevsky et al. [15] for object recognition. There have been extensions of this model for the problem of action recognition recently. In [23], a two-stream CNN model is proposed for this task with very promising results. The two-stream model is enhanced using a VGG network in [7], and intermediate fusion of the convolutional layers incorporated. A trajectory-pooled CNN setup is described in [29] that fuses CNN features along action trajectories. Early and late fusion of CNN feature maps is proposed in [13, 32]. Almost all these methods combine the feature maps without accounting for the temporal evolution of these features; as a result, might lose capturing temporal action dynamics effectively.

To account for this shortcoming, Fernando et al. [10] proposed a pooling formulation on video features that accounts for the temporal order of frames. This formulation is extended for RGB frames in [4, 8]. We base this paper on this extension, however deviate from their approach in that we use optical flow frames instead of RGB frames for capturing the action dynamics. To the best of our knowledge, we are not aware of any other work that has analyzed the advantages of using optical flow images in the rank-SVM formulation for action recognition. We showcase extensive experiments to substantiate the benefits against using RGB frames.

We also note that there have been several other deep learning models devised for action modeling such as using recurrent neural networks [3, 6] and long-short term memory networks [5, 17, 32]. While, we acknowledge that using recurrent architectures may benefit action recognition, usually datasets for this task are often small or are noisy, that training such models will be challenging. Thus, we focus on advancing action representations in this paper for effective CNN training.

3. Proposed Method

In this section, we introduce our dynamic flow images; we describe the necessary formulations to generate these images, followed by an exposition to a multi-stream CNN framework for action recognition. Note that our dynamic flow formulation is inspired by the recently introduced dynamic image framework [4], however, our contribution is to show the shortcomings of that formulation and propose remedies.

3.1. Formulation

Let us assume that we are provided with a sequence of n consecutive optical flow images $\mathcal{F} = [f_1, f_2, \cdots, f_n]$, where each $f_i \in \mathbb{R}^{d_1 \times d_2 \times 2}$, where d_1, d_2 are the height and width of the image. We assume a two-channel flow image corresponding to the horizontal and vertical components of the flow vector, which we denote by f_i^u and f_i^v respectively. Our goal is to generate a single "dynamic flow" image $F \in \mathbb{R}^{d_1 \times d_2 \times 2}$ that captures the temporal order of the flow images in $\mathcal{F}$. We use the following formulation to obtain this representation.

$$\min_{F \in \mathbb{R}^{d_1 \times d_2 \times 2}, \xi \geq 0} \quad \|F\|^2 + C \sum_{i<j} \xi_{ij} \tag{1}$$
$$\text{s. t. } \langle F, f_i \rangle \leq \langle F, f_j \rangle + 1 - \xi_{ij}, \quad \forall i < j,$$

where $\langle ., . \rangle$ represents an inner product between the original flow vectors and the dynamic flow image to be found. As one may recall, this formulation is similar to the Ranking-SVM formulation [12], where the inner product captures the ranking order, which in our case corresponds to the temporal order of the frames in the sequence. While, ideally, we enforce that the projection (via the inner-product) of the flow frames to the dynamic flow image is lower bounded by one, we relax this constraint using the variables ξ as is usually done in a max-margin framework. While, the above formulation is a direct adaptation of the Ranking SVM formulation, there are a few subtleties that need to be taken care when using this formulation for optical flow images. We will discuss these issues next.

It is often observed that a direct use of raw optical flow in (1) is often inadequate. This is because computing optical flow is a computationally expensive and difficult task, and the flow solvers are often prone to local minima, leading to inaccurate flow estimates. In order to circumvent such issues, we instead apply the algorithm on flow images computed over running averages; such a scheme averages the noise in the flow estimates. For the flow set $\mathcal{F}$, we represent the resulting smoothed flow image for frame f_t as:

$$\hat{f}_t = \frac{1}{t} \sum_{i=1}^{t} f_i. \tag{2}$$

We compute the dynamic flow on such smoothed flow images. Note that since flow is signed, such averaging will cancel white noise.

Another issue specific to optical flow images is that the two flow channels u and v are coupled. They are no more the color channels as in an RGB image, instead they represent velocity vectors, which together capture the motion vector at a pixel. However, our ranking formulation assumes indepedence in the channels. One way to avoid this difficulty is to decorrelate these channels via diagonalization; i.e., use the singular vectors associated with a short set of flow images as representatives of the original flow images. However, our experiments showed that this did not lead to significant improvements. Thus, we choose to ignore this coupling in the current paper, and assume each channel is independent when creating the dynamic flow images. Our experiments (in Section 4) show that this assumption is not detrimental.

Incorporating the above assumptions into the objective, and assuming $F_u, F_v \in \mathbb{R}^{d_1 \times d_2}$ are the two dynamic flow

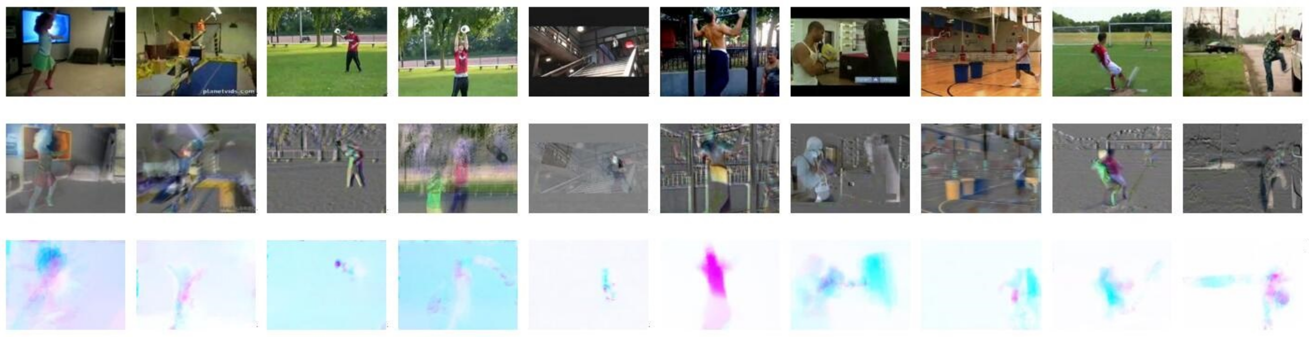

Figure 2. Qualitative comparisons between dynamic images [4] (second row) and our proposed dynamic flow image (third row) representation. An RGB frame from the respective video is also shown (first row). As can be seen from the visualizations, dynamic flow focuses on actions while dynamic images can be contaminated by background pixels.

channels corresponding to the horizontal and vertical flow directions, we can rewrite (1) as:

$$\min_{F_u, F_v \in \mathbb{R}^{d_1 \times d_2}, \xi \geq 0} \|F_u\|^2 + \|F_v\|^2 + C \sum_{i<j} \xi_{ij}$$

subject to

$$\langle F_u, \hat{f}_i^u \rangle + \langle F_v, \hat{f}_i^v \rangle \leq \langle F_u, \hat{f}_j^u \rangle + \langle F_v, \hat{f}_j^v \rangle + 1 - \xi_{ij}, \forall i < j. \quad (3)$$

We solve the above optimization problem using the libSVM package as described in [4]. Once the two flow images F_u and F_v are created, they are stacked to generate a two channel dynamic flow image, which is then fed to a multi-stream CNN as described for learning actions (as described in the next section). In Figure 2, we show several examples of the dynamic flow images, and their respective RGB frames, and dynamic RGB images. As is clear from these visualizations, the dynamic flow images summarizes the actual action dynamics in the sequences, while the dynamic RGB images include averaged background pixels and thus are may include dynamics unrelated to recognizing actions.

3.2. Three-Stream Prediction Framework

Next, we propose a three-stream CNN setup for action recognition. This architecture is an extension of the popular two-stream model [23] that takes as input individual RGB frames in one stream and a small stack of optical flow frames (about 10 flow images) in the other. One shortcoming of this model is that it cannot see long-range action evolution, for which we propose to use our dynamic flow images (that summarizes about 25 flow frames in one dynamic flow image). As is clear, each such stream looks at complementary action cues. Our overall framework is illustrated in Figure 3.

To be precise, for the dynamic flow stream, for each video sequence, we generate multiple dynamic flow images. In order to achieve this, we first split the input flow video into several sub-sequences each of length w and generated at a temporal stride s. For each sub-sequence, we construct a dynamic flow image using the optical flow images in this window. We associate the same ground truth action label for all the sub-sequences, thus effectively increasing the number of training videos by a factor of $\frac{n}{s}$, where n is the average number of frames in the sequences. Note that we use a separate CNN stream on dynamic flow images. Given that action recognition datasets are usually tiny, in comparison to image datasets (such as ImageNet), increasing the training set is usually necessary for the effective training of the network.

3.3. Practical Extensions

We use the TVL1 optical flow [33] algorithm to generate the flow images using its OpenCV implementation. For every flow image, we subtract the median flow, thus removing camera motion if any (assuming flow from the action dynamics occupies a small fraction with respect to the background). The resulting flow images are then thresholded in the range of $[-20, 20]$ pixels and setting every other flow vector to zero. This step thus removes unwanted flow vectors, that might correspond to noise in the images. Next, we scale the flow to the discrete range of $[0, 255]$, and convert each flow channel as a gray-scale image. A set of such transformed flow images are then used as input to the rank-SVM formulation in (2), thereby generating one dynamic flow image per sub-sequence. Using libSVM package, it takes about 0.25 seconds to generate one dynamic flow image on a single core CPU combining 25 flow frames each of size $224 \times 224 \times 2$.

4. Experiments

In this section, we first describe the datasets used in our experiments, followed by an exposition to the implementation details of our framework and comparisons to prior works.

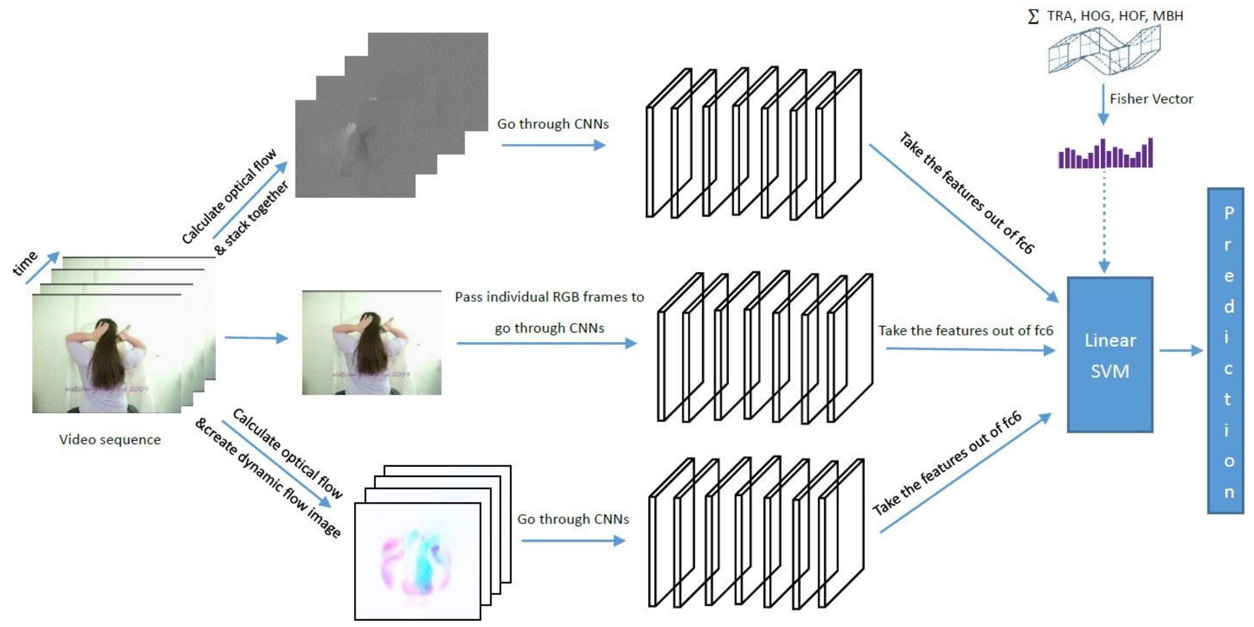
∑ TRA, HOG, HOF, MBH
Fisher Vector
Calculate optical flow & stack together
time
Go through CNNs
Take the features out of fc6
Pass individual RGB frames to go through CNNs
Take the features out of fc6
Linear SVM
Prediction
Video sequence
Calculate optical flow &create dynamic flow image
Go through CNNs
Take the features out of fc6

Figure 3. Architecture of our dynamic-flow CNN based classification setup. Our three-stream CNN consists of an optical flow stream taking stacks of flow frames, an RGB stream taking single RGB frames, and our dynamic flow stream taking single images, each image summarizing the action dynamics over 25 flow frames.

4.1. Datasets

We use two standard datasets for the task: (i) the HMDB51 dataset [16] and (ii) the UCF101 dataset [25]. For both the datasets, the standard evaluation protocol is the average classification accuracy over three splits.

HMDB51 dataset [16] consists of 6766 video clips distributed in 51 action categories. The videos are downloaded from Youtube and are generally of low resolution.

UCF101 dataset [25] is a relatively larger dataset, containing 13,320 videos spanning over 101 actions. The videos are mostly from sports activities and contain significant camera motions and people appearance variations.

4.2. Implementation Details

Training CNNs. In the experiments to follow, we use the two successful CNN architectures, namely Alexnet [15] and VGG-16 [24]. We use the Caffe toolbox [11] for the implementation. As the number of training videos is substantially limited to train a standard deep network from scratch, we decided to fine-tune the networks from models pre-trained for image recognition tasks (on the ImageNet dataset). On the training subsets of our dataset, we fine-tune all the layers of the respective networks with an initial learning rate of 10^{-4} for VGG-16 and 10^{-3} for Alex net with a drop-out of 0.85 for 'fc7' and 0.9 for 'fc6' as recommended in [24, 7]. The drop-out is subsequently increased when the validation loss begins to increase. The network is trained using SGD with a momentum of 0.9 and weight decay of 0.0005. We use a mini-batch size of 64 for HMDB51 and 128 for UCF101.

Fusion Strategy. As alluded to earlier, we use a three-stream network with RGB, stacked-optical flow, and dynamic flow images. The three networks are trained separately. During testing, output of the fc6-layer of each stream is extracted (see Figure 3). These intermediate features are then concatenated into a single vector, which is then fused via a linear SVM. We also experimented with features from the fc7 layer, however we found them to perform slightly inferior compared to fc6. As is typically done, we also extract dense trajectory features from the videos (such as HOG, HOF, and MBH) [27], which are encoded using Fisher vectors, and are concatenated with the CNN features before passing to the linear SVM.

4.3. Experimental Results

We organize our experiments into various categories, namely (i) to decide the hyper-parameters of our formulation (e.g., the window size to generate dynamic flow images), (ii) benefits of using dynamic flow images against dynamic RGB images and other data modalities, (iii) influence of the CNN architecture, (iv) complementarity of the dynamic flow images to hand-crafted features, and (v) comparisons against the state of the art. Below, we detail each of these experiments.

Table 1. Influence of using different number of consecutive flow frames (window size) to construct one dynamic flow image. This experiment uses the split 1 of HMDB51 dataset with the VGG-16 CNN model.

Dynamic Flow window size	Accuracy
15	48.23%
25	**48.75%**
30	46.60%

Table 2. Evaluation on HMDB51 split 1 using VGG-16 model.

Method	Accuracy
Static RGB[7]	47.06%
Stacked Optical Flow [7]	55.23%
Dynamic Image [4]	44.74%
Dynamic Flow	48.75%
Dynamic Image+RGB	47.96%
Dynamic Flow+RGB	**58.30%**
Dynamic Image+Dynamic Flow+RGB	54.86%
(S)Optical Flow+RGB[7]	58.17%
Dynamic Image+RGB+(S)Optical Flow	55.40%
Dynamic Flow+RGB+(S)Optical Flow	**61.70%**

Table 3. Evaluation on UCF101 using AlexNet CNN model.

Method	Accuracy
On split 1	
Static RGB[29]	71.20%
Stacked Optical Flow[29]	80.10%
Dynamic Flow	75.36%
Dynamic Flow + RGB	**84.93%**
(S)Optical Flow + RGB[29]	84.70%
Dynamic Flow + RGB + (S)Optical Flow	**88.63%**
Over three splits[1]	
Static RGB	70.10%
Dynamic Flow	76.19%
Dynamic Image[4]	70.90%
Dynamic Image + RGB[4]	76.90%
Dynamic Flow + RGB	**84.93%**

Hyper-parameters: We tested the performance of dynamic flow images using different temporal window sizes, i.e., the number of consecutive flow frames used for generating one dynamic flow image. The results of this experiment on the HMDB51 dataset split 1 is provided in Table 1. As is clear, increasing window sizes is not beneficial as it may lead to more action clutter. Motivated by this experiment, we use a window size of 25 at a temporal stride of 5 in all the experiments in the sequel.

Benefits over Dynamic Images (DI): As discussed in Section 3, dynamic flow (DF) captures the action dynamics directly in comparison to the dynamic RGB images [4]. To demonstrate this, we show experiments using the VGG-16 network in Table 2 on the HMDB51 dataset and using the Alexnet network on the UCF101 dataset in Table 3[2]. We also show comparisons to RGB, stack of flow, and various combinations of them. As is clear from the two tables, DF + RGB is about 9-10% better than DI + RGB consistently on both the datasets and both CNN architectures. Surprisingly, combining DI with DF + RGB leads to a reduction in performance of about 4% (Table 2). This, we suspect, is because DI includes pixel dynamics from the scene background that may be unrelated to the actions and may confuse the subsequent classifier; such noise is avoided by computing optical flow. We investigate this further, in Table 4, where we provide the per-class accuracy on HMDB51 for DI+RGB, DF+RGB, and DI+DF+RGB. We find that 29 of the 51 actions in this dataset lose in performance when using DI+DF+RGB as against DF+RGB, which implies that DF+RGB is indeed a better representation for actions.

Benefits of Three-stream Model: In Tables 2 and 3, we also evaluate our three-stream network against the original two-stream framework [24]. It can be seen that the performance of DF + RGB is similar to Stack(S) of optical flow + RGB (which is the standard two-stream model) on both UCF101 and HMDB51 datasets. However, interestingly, we find that, after combining stacked optical flow with DF+RGB, the accuracy improves by 3% on HMDB51 and 4% on UCF101, which shows that our new representation carries complementary information (such as long-range dynamics) that is absent previously.

Comparisons between CNN architectures: To further validate and understand the behavior of the three-stream model, we repeated the experiment in the last section using an Alexnet architecture. As is shown in Table 5 and 6, the accuracy of VGG-16 is seen to be 2%-7% higher than for Alex net, which is expected. We also find that the performance of DF, DF + RGB and DF + RGB + stacked optical flow show the same trend in both Alex net and VGG-16 networks.

Benefits from Hand-crafted Features: In Table 7 and 8, we evaluate the performance of three-stream network and its combination with IDT-FV [27]. On HMDB51, after applying IDT-FV, the accuracy of each method improves by 2% to 10%. While, using this combination also improves the accuracy for DI + RGB, this improvement is significantly inferior to DF+RGB. On the UCF101 dataset, IDT-FV improves performance by 1– 2%.

Comparisons to the State of the Art: In Table 9, we compare our method against the state-of-the-art results on HMDB51 and UCF101 datasets. For this comparison,

[2]The results of Alexnet on UCF101 was taken directly from [4].

Table 4. Accuracy on each class in HMDB51 split 1 using different methods.

Action	DI+RGB	DF+RGB	DI+DF+RGB
brush_hair	60%	**77%**	**77%**
cartwheel	3%	**23%**	13%
catch	27%	**43%**	**43%**
chew	43%	**60%**	**60%**
clap	38%	**52%**	**52%**
climb_stairs	**60%**	53%	**60%**
climb	67%	**73%**	**73%**
dive	50%	**60%**	53%
draw_sword	30%	**47%**	40%
dribble	73%	**90%**	80%
drink	20%	**43%**	**43%**
eat	33%	**50%**	37%
fall_floor	34%	24%	**38%**
fencing	63%	**77%**	70%
flic_flac	30%	**50%**	**50%**
golf	93%	**97%**	93%
handstand	50%	**70%**	60%
hit	14%	25%	**29%**
hug	53%	**67%**	57%
jump	31%	**52%**	41%
kick_ball	37%	**40%**	33%
kick	7%	**20%**	17%
kiss	83%	83%	**87%**
laugh	50%	**70%**	57%
pick	30%	**40%**	37%
pour	83%	**97%**	93%
pullup	87%	**100%**	**100%**
punch	63%	63%	**70%**
push	70%	**87%**	77%
pushup	57%	60%	**67%**
ride_bike	93%	93%	**97%**
ride_horse	80%	**83%**	77%
run	32%	**50%**	39%
shake_hands	70%	73%	**77%**
shoot_ball	80%	**87%**	83%
shoot_bow	87%	**93%**	87%
shoot_gun	67%	60%	**70%**
sit	37%	**57%**	53%
situp	**87%**	77%	83%
smile	40%	**43%**	40%
smoke	40%	47%	**53%**
somersault	60%	**83%**	70%
stand	20%	**23%**	20%
swing_baseball	13%	**17%**	**17%**
sword_exercise	13%	**30%**	17%
sword	**31%**	17%	21%
talk	57%	**67%**	60%
throw	10%	**33%**	7%
turn	37%	**57%**	47%
walk	40%	43%	**50%**
wave	7%	**20%**	**20%**
Average	48%	**58%**	55%

Table 5. Accuracy comparison between AlexNet and VGG-16 on HMDB51 split 1.

Method	Alex net	VGG-16
Static RGB	40.50%[23]	47.12%
Dynamic Flow	43.69%	48.75%
Dynamic Image	40.88%	44.74%
Dynamic Flow + RGB	**53.75%**	**58.30%**
Dynamic Image + RGB	45.21%	47.96%

Table 6. Accuracy comparison between AlexNet and VGG-16 on UCF101 split 1.

Method	Alex net	VGG-16
Static RGB	71.20%[29]	80.00%
Dynamic Flow	75.36%	78.00%
Dynamic Flow+RGB	**84.25%**	**87.63%**
Dynamic Flow+RGB+(S)Optical flow	**88.65%**	**90.30%**

Table 7. Accuracy comparison on HMDB51 split 1 using VGG-16 after combining with IDT-FV.

Method	+IDT-FV	Original
Static RGB	61.89%	47.06%
Dynamic Flow	58.30%	48.75%
Dynamic Image	47.96%	44.74%
Dynamic Image+RGB	65.44%	47.96%
Dynamic Flow+RGB	67.35%	58.30%
Dynamic Flow+RGB+(S)Optical flow	**67.48%**	**61.70%**
Dynamic Image+RGB+(S)Optical flow	64.13%	54.40%

Table 8. Accuracy comparisons on UCF101 split 1 using VGG-16 after combining with IDT-FV.

Method	+IDT-FV	Original
Dynamic Flow+RGB	**89.20%**	87.63%
Dynamic Flow+RGB+(S)Optical flow	**91.10%**	90.30%

we use the VGG-16 model trained on dynamic flow images, combined with static RGB, a stack of 10 optical flow frames, and IDT-FV features. The results are averaged over three splits as is the standard protocol. For both the datasets, we find that our three stream model with dynamic flow images outperforms the best results using dynamic image networks [4]. For example, on HMDB51, our results by replacing the dynamic image with dynamic flow leads to a 2% improvement. We also notice a performance boost against the recent hierarchical variant [8] of the dynamic images (66.9% against 67.35%) that recursively summarizes such images from video sub-sequences.

Compared to other state-of-the-art methods, our results are very competitive. Notably, while the accuracy of two-stream fusion [7] is slightly higher than ours, our CNN architecture is significantly simpler. In Figures 4 and 5, we provide qualitative comparisons between dynamic flow and dynamic images.

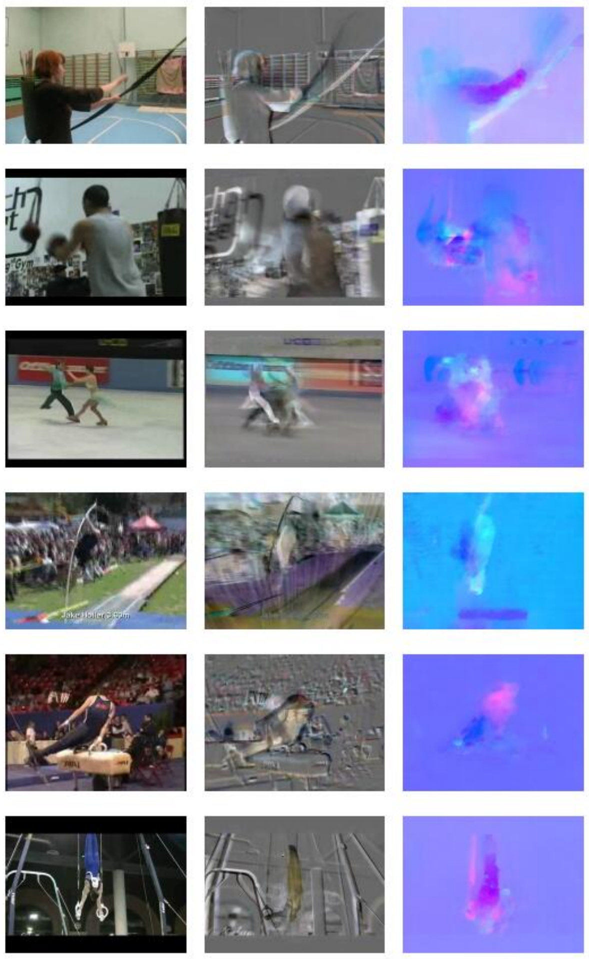
Figure 4. Left to right: Qualitative results of RGB frames, dynamic images, and dynamic flow on UCF101 dataset.

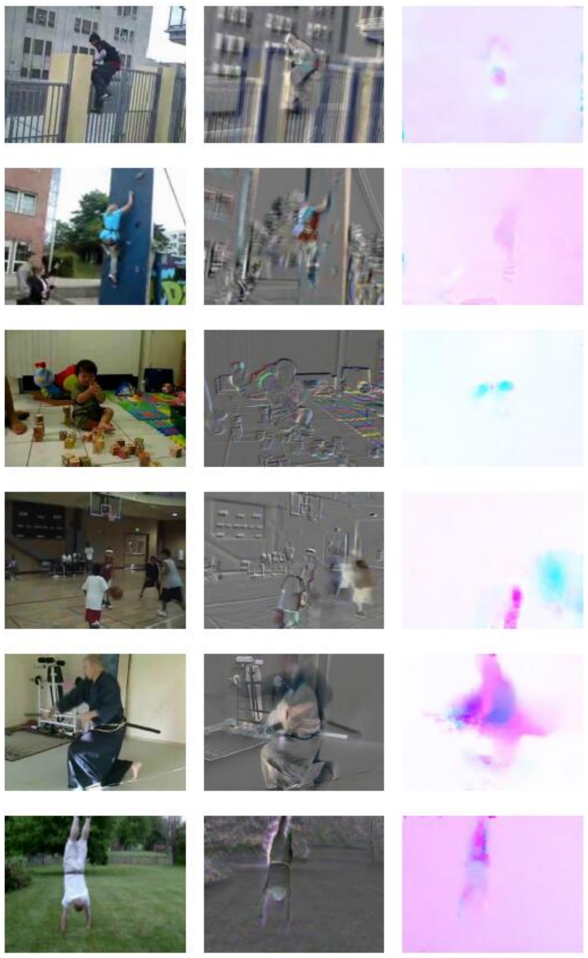
Figure 5. Left to right: Qualitative results of RGB frames, dynamic images, and dynamic flow on HMDB51 dataset.

Table 9. Classification accuracy against the state of the art on HMDB51 and UCF101 datasets averaged over three splits.

Method	HMDB51	UCF101
Two-stream [23]	59.40%	88.00%
Two-stream SR-CNNs [30]	–	92.60%
Very Deep Two-stream Fusion [7]	**69.20%**	**93.50%**
LSTM-MSD [17]	63.57%	90.80%
IDT-FV [28]	57.20%	86.00%
IDT-HFV [20]	61.10%	87.90%
TDD+IDT-FV [29]	65.90%	91.50%
C3D + IDT-FV [26]	–	90.40%
Dynamic Image + RGB + IDT-FV [4]	65.20%	89.10%
Hierarchical Rank Pooling [9]	66.90%	91.40%
Ours		
Dyn. Flow + RGB + IDT-FV	67.19%	89.41%
Dyn. Flow+RGB+(S)Op.Flow+IDT-FV	67.35%	91.32%

5. Conclusions

In this paper, we presented a novel video representation – dynamic flow– that summarizes a set of consecutive optical flow frames in a video sequence as a single two-channel image. We showed that our representation can compactly capture the long-range dynamics of actions. Considering this representation as an additional CNN input cue, we proposed a novel three-stream CNN architecture that incorporates single RGB frames (for action context), stack of flow images (for local action dynamics), and our novel dynamic flow stream (for long range action evolution). Experiments were provided on standard benchmark datasets (HMDB51 and UCF101) and clearly demonstrate that our method is promising in comparison to the state of the art. More importantly, our experimental results reveal that our representation captures the action dynamics more robustly than the recent dynamic image algorithm and provides complimentary information (long-range) compared to the traditional stack of flow frames.

Acknowledgements: AC is supported by the Australian Research Council (ARC) through the Centre of Excellence for Robotic Vision (CE140100016).

References

[1] J. K. Aggarwal and M. S. Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3):16, 2011. 1

[2] B. B. Amor, J. Su, and A. Srivastava. Action recognition using rate-invariant analysis of skeletal shape trajectories. *PAMI*, 38(1):1–13, 2016. 2

[3] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Sequential deep learning for human action recognition. In *Human Behavior Understanding*, pages 29–39. 2011. 3

[4] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *CVPR*, 2016. 2, 3, 4, 6, 7, 8

[5] 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. In *CVPR*, 2015. 3

[6] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, 2015. 3

[7] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 1, 2, 3, 5, 6, 7, 8

[8] B. Fernando, P. Anderson, M. Hutter, and S. Gould. Discriminative hierarchical rank pooling for activity recognition. In *CVPR*, 2016. 3, 7

[9] B. Fernando, P. Anderson, M. Hutter, and S. Gould. Discriminative hierarchical rank pooling for activity recognition. In *CVPR*, 2016. 8

[10] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *CVPR*, 2015. 2, 3

[11] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Intl. Conf. on Multimedia*. ACM, 2014. 5

[12] T. Joachims. Optimizing search engines using clickthrough data. In *SIGKDD*. ACM, 2002. 3

[13] 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. 3

[14] P. Koniusz, A. Cherian, and F. Porikli. Tensor representations via kernel linearization for action recognition from 3D skeletons. *ECCV*, 2016. 2

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 2, 5

[16] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 2, 5

[17] Q. Li, Z. Qiu, T. Yao, T. Mei, Y. Rui, and J. Luo. Action recognition by learning deep multi-granular spatio-temporal video representation. In *ICMR*, 2016. 3, 8

[18] A. A. Liu, Y. T. Su, W. Z. Nie, and M. Kankanhalli. Hierarchical clustering multi-task learning for joint human action grouping and recognition. *PAMI*, 39(1):102–114, 2017. 2

[19] L. Liu, L. Shao, X. Li, and K. Lu. Learning spatio-temporal representations for action recognition: a genetic programming approach. *IEEE Transactions on Cybernetics*, 46(1):158–170, 2016. 2

[20] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *Computer Vision and Image Understanding*, 2016. 2, 8

[21] R. Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010. 1

[22] S. Sadanand and J. J. Corso. Action bank: A high-level representation of activity in video. In *CVPR*, 2012. 2

[23] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 1, 2, 4, 7, 8

[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 5, 6

[25] K. Soomro, A. Roshan Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. In *CRCV-TR-12-01*, 2012. 2, 5

[26] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, 2015. 8

[27] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *International journal of computer vision*, 103(1):60–79, 2013. 5, 6

[28] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 1, 2, 8

[29] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015. 1, 2, 3, 6, 7, 8

[30] Y. Wang, J. Song, L. Wang, L. Van Gool, and O. Hilliges. Two-stream SR-CNNs for action recognition in videos. In *BMVC*, 2016. 8

[31] J. Wu, Y. Zhang, and W. Lin. Good practices for learning to recognize actions using FV and VLAD. *IEEE Transactions on Cybernetics*, 2015. 2

[32] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 3

[33] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. In *Joint Pattern Recognition Symposium*, 2007. 4