Recognize Actions by Disentangling Components of Dynamics

Yue Zhao[1], Yuanjun Xiong[1,2], and Dahua Lin[1]

[1]CUHK - SenseTime Joint Lab, The Chinese University of Hong Kong [2]Amazon Rekognition
{zy317,dhlin}@ie.cuhk.edu.hk {yuanjx}@amazon.com

Abstract

Despite the remarkable progress in action recognition over the past several years, existing methods remain limited in efficiency and effectiveness. The methods treating appearance and motion as separate streams are usually subject to the cost of optical flow computation, while those relying on 3D convolution on the original video frames often yield inferior performance in practice. In this paper, we propose a new ConvNet architecture for video representation learning, which can derive disentangled components of dynamics purely from raw video frames, without the need of optical flow estimation. Particularly, the learned representation comprises three components for representing static appearance, apparent motion, and appearance changes. We introduce 3D pooling, cost volume processing, and warped feature differences, respectively for extracting the three components above. These modules are incorporated as three branches in our unified network, which share the underlying features and are learned jointly in an end-to-end manner. On two large datasets, UCF101 [22] and Kinetics [16], our method obtained competitive performances with high efficiency, using only the RGB frame sequence as input.

1. Introduction

In recent years, action recognition in videos has received increasing attention from the vision community [25, 21, 23, 3], due to its great potential value in real-world applications. A distinctive aspect of the video-based action recognition task is that the *dynamics*, *i.e*. the temporal change in the visual content, plays a crucial role. Whether the dynamics can be effectively represented and utilized, to a large extent, determines the performance of an action recognition method. A key goal of this work is to explore an *efficient* and *effective* way to capture the dynamics in videos.

Since the introduction of deep networks to this area, two different categories of methods have emerged for video modeling. The first category of methods rely on a combination of multiple input modalities, *e.g*. appearance and motion, to represent videos. The *two-stream convnet* [21], a seminal work of this category, combines RGB images and optical flows for video-based recognition. A key advantage of this category lies in its strong performance. Many state-of-the-art frameworks [9, 29, 35, 38] can be considered as variants of this paradigm. Yet, the methods in this category are subject to a significant limitation, that is, the reliance on optical flows to represent motion, which are often expen-

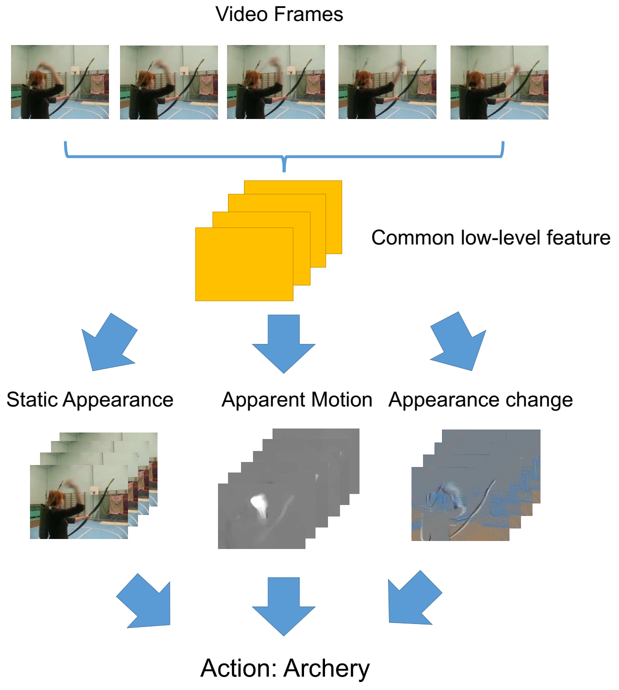

Figure 1. Illustration of disentangling components of dynamics. From the low-level visual feature maps, we derive three components, *i.e*., static appearance (left), apparent motion (center), and appearance changes (right) and combine the high-level feature representation extracted thereon for action recognition.

sive to compute. The models for estimating optical flows are often learned separately, driven by an objective unrelated to recognition. Such issues motivate us to question the necessity of explicitly computing optical flows for action recognition. The second category of methods [13, 23, 3, 18] attempt to derive a unified representation directly from the sequence of frames through 3D convolution. While this approach seems to be more elegant, it currently yields inferior performance when using only original video frames.

The discussion above suggests that the separation of appearance, motion, and even other components entangled in a video may be the key to achieve high performance. However, the conventional way, which relies on an optical flow estimator to separate the motion channel, is limited in both *efficiency* (computing optical flows is costly) and *effectiveness* (optical flows are not tailored to discrimination). In this work, we aim to move beyond such limitations and develop an approach that is free from optical flow computation, while still enjoying the strong performance of two-stream style modeling.

Towards this goal, we develop a new ConvNet architecture to learn video representation for action recognition. This model can derive disentangled components *purely* from video frames *without using optical flows*. Specifically, we consider a video as a combination of short-term dynamics and long-term temporal structures. We focus on modeling the former, while using TSN [30], an efficient framework for long-term video modeling, to take care of the latter. As shown in Figure 1, we consider three components for short-term representation, *static appearance*, *apparent motion*, and *appearance changes*. Given a video, our network first computes common low-level features from the sequence of frames through several layers of convolution. Thereon, it disentangles the aforementioned components via three branches, which respectively use 3D pooling to extract static appearance, cost volume processing to capture apparent motion, and a novel scheme to express appearance changes via warped feature differences. These modules are integrated into a unified network to make the final predictions and are learned jointly in an end-to-end fashion.

Overall, the key contribution of this work is a unified network architecture for learning video representation. This architecture is distinguished in three aspects: 1) It can obtain *disentangled components* directly from raw video frames to effectively represent short-term dynamics. This procedure is free from optical flow computation or other external modules. 2) It is both *efficient* – the computation of all components can share underlying features, and *effective* – separated modeling of different components results in high performance. 3) On two public benchmarks for action recognition, UCF101 [22] and Kinetics [16], it obtains improved performance compared to state-of-the-art methods, using only RGB sequences as input, while having high runtime efficiency.

2. Related Work

We now briefly review previous efforts in understanding the video contents. Specifically, we will discuss 1) works on using convolutional networks for action recognition; 2) the two-stream ConvNet method and its variants; 3) other efforts in harnessing motion information for action recognition.

Convolutional networks for action recognition. Deep convolutional neural networks based models have seen wide applications in action recognition [15, 21, 23] and have gradually surpassed the performance of traditional methods [25, 26, 17]. Beyond simply classifying videos by aggregating frame-based prediction [15], a set of methods have been proposed to use 3D convolutions and directly model the dynamics in video frames [13, 23]. While appearing elegant, one possible limitation of these methods is that they mix the modeling of visual appearance and content changes in the 3D convolution operations. Thus their performance was inferior to traditional methods. Two-stream ConvNet based methods were introduced later to mitigate this issue by separating the modeling of appearance and temporal changes [21, 33, 9, 30]. These methods rely on dense optical flow estimation provided by external tools to represent short-term motion information. They achieved much better performance and successfully surpassed the traditional methods in terms of recognition performance. Recently, a hybrid of 3D convolution and two-stream ConvNet was proposed in [24, 3] and achieved competitive results on large-scale datasets [16]. But these methods are still limited by the heavy computation of optical flow. Another line of research investigate how to utilize long term temporal information to help action recognition [30, 5, 7] and temporal action detection [37, 28]. They have shown that end-to-end learning of action recognition models with the help of long term information can benefit the recognition performance.

Motion representation. Motion information is the key factor that differentiates video-based action recognition from image-based object recognition. Many works have been dedicated to exploring different design of motion representations for action recognition [25, 21, 32, 36]. Conventional approaches [6, 25, 26] used hand-crafted features of trajectories which are based on optical flow. More recent convolutional networks treat a stack of optical flow as input directly. The most widely used optical flow method for action recognition is TV-L1 [34]. It is computed by minimizing an energy function defined over the whole displacement field, which prohibits it from end-to-end training. In [31], Xu *et. al.* proposed to construct a four-dimensional cost volume to get a semi-dense correspondence map. The final result was obtained by refinement and interpolation. Most of the state-of-the-art optical flow estimation approaches [1, 19, 31]

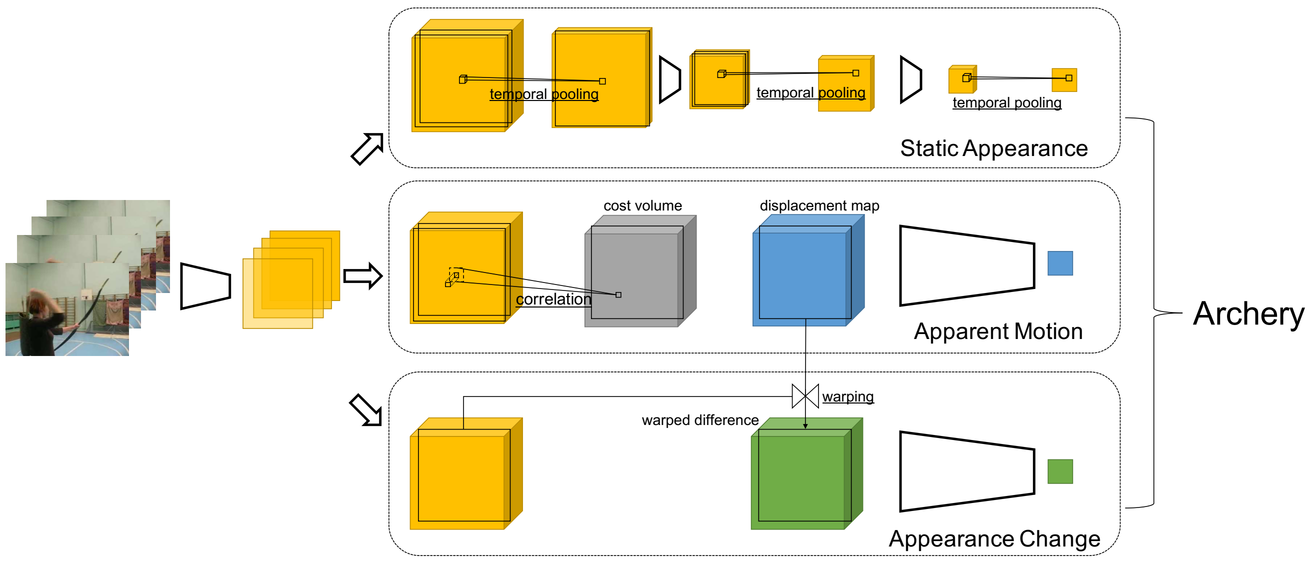

Figure 2. Overview of the proposed network architecture. Given a consecutive frame sequence, the model first produces some low-level feature maps which are then fed into three branches, namely static appearance (upper), apparent motion (middle), and appearance change (bottom). These branches compute their corresponding high-level feature and make predictions respectively. At the end, these predictions are combined to be the final prediction.

are prohibitive for action recognition applications due to the high computational cost. FlowNet [8, 11] is a family of CNN-based optical flow prediction methods for general purpose. It introduced a correlation layer to predict optical flow from inter-frame correlation. However, a larger network (FlowNet 2.0) cascaded with several subnetworks is required for good estimation, which at the same time brings much computation. Zhu *et. al.* proposed a fully convolutional network called *MotionNet* in [38] to produce optical flow by next frame prediction. The learned optical flow was then plugged into the two-stream network.

Due to the expensive computation, optical flow is usually calculated in advance and then stored in hard drives, which increases the burden of storage. Motivated by this, efforts have been made to find good alternatives. Zhang *et. al.* proposed to use motion vector [35], which can be obtained directly from compressed videos without extra calculation. However, motion vector is noisy and lacks fine structure. Wang *et. al.* found in [30] that the difference between adjacent RGB frames, namely RGB-diff, can be used instead to substitute optical flow at the cost of performance degradation. In [4, 2], alternative motion representations have been introduced by collapsing the information from multiple frames into a single image.

3. Method

Our primary goal in this work is to develop an *efficient* and *effective* representation of short-term dynamics in videos. This representation can work with various high-level models for capturing long-term temporal structures, *e.g.* TSN [30] and LSTM [7], to completely model a video. Towards this goal, we take a systematic perspective and consider the dynamics as a compound of three key components: 1) *static appearance* – the overall scene appearance, which is usually stable over time, 2) *apparent motion* – the changes due to the movement of objects or the camera, and 3) *appearance changes* – the inherent changes in the appearance caused by other factors.

3.1. Overall Architecture

We design a *unified* convolutional network that can disentangle the three components outlined above *purely* based on an input sequence of frames. Figure 2 shows the overall architecture of our design. Specifically, given a short video clip in the form of a frame sequence, the network first produces low-level feature maps with 64 channels for individual frames via several convolution layers. The low-level feature maps mainly capture the primitive visual elements and thus can be shared by all components as the common foundation for further processing. Subsequently, these features are fed to three branches, each of which is devised to characterize a component. These branches will compute the component-specific high-level features and then make classification predictions respectively. At the end, the component-specific predictions would be combined into the final prediction by average.

The detailed design of the architecture follows BN-Inception [12]. Technical details of the architecture are provided in the supplemental materials. In what follows, we

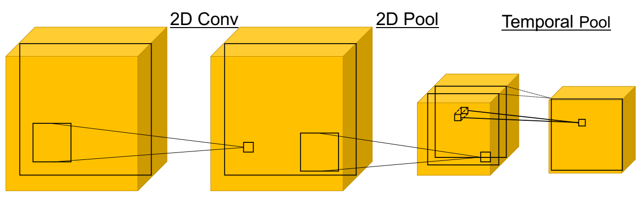

Figure 3. An illustration of the static appearance branch with 2D convolution, spatial 2D pooling and temporal 1D pooling.

will introduce the three component-specific branches with details, as well as how the components are fused.

3.2. The Static Appearance Branch

The first branch is to derive a representation of the *static appearance* of the observed scene. In mainstream architectures, *e.g.* two-stream ConvNet [21] and its variants [7, 27, 30], the appearance features are typically computed per frame and thus are sensitive to nuances like motion blur and sudden camera movement. This issue can be effectively mitigated by selecting the highest responses across neighboring frames.

Following this idea, we incorporate both convolution layers and temporal pooling layers in this branch. The former is for extracting visual patterns (like in a typical CNN), while the latter is for *stabilizing* the features across neighboring frames. Figure 3 shows the structure of this branch. Given a sequence of frame-wise low-level features, this branch gradually distill the appearance feature by *iteratively* applying 2D convolution, spatial 2D pooling, and temporal 1D pooling. Here, the combination of a spatial pooling layer followed by temporal pooling layer essentially constitutes a *3D pooling* operation that pools activations over spatio-temporal neighborhoods. Along the iterations, both the spatial and temporal resolutions of the feature maps are gradually reduced, while the number of channels is increased. Over multiple iterations, the input feature maps with 64 channels will be turned into a *single* 1024-dimensional feature vector, which serves as the representation of the static appearance component.

Note that the *3D convolution* introduced in previous work [23] also allows the features to be fused both spatially and temporally. However, for capturing static appearance, our design of *2D convolution* followed by *3D pooling* is more suitable for two reasons: 1) The purpose of this component is to capture the features that are *stable* over time but not to account for the dynamics. Our design offers enough expressive power to capture appearance patterns, but in a much lower cost. 2) 3D convolution layers, due to their significantly larger parameters, often take more samples and iterations to train. In our design, 3D pooling is parameter-free, while the 2D convolution layers can be readily transferred from the CNNs pretrained on image data.

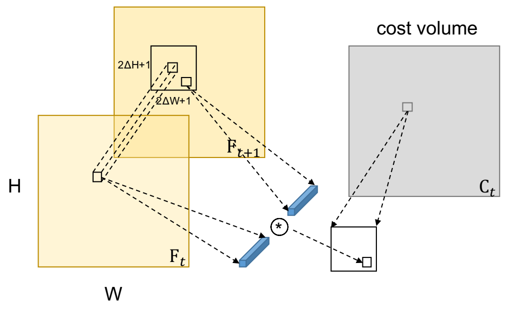

Figure 4. An illustration of cost volume construction. Note that the cost volume $\mathbf{C}_t$ is a four-dimension tensor.

3.3. The Apparent Motion Branch

Generally, *apparent motion* refers to the spatial displacement of feature points in video frames. Previous works on action recognition [21, 27, 30] have repeatedly shown the significance of the motion information. In those works, the apparent motion is usually represented by dense optical flow fields. However, as mentioned, the computation of optical flows is expensive and often requires an external module.

In designing the branch for apparent motion, we explore an alternative way, that is, to directly base the motion representation on *cost volume*. Note that cost volumes have been widely used in low-level vision tasks, *e.g.* optical flow estimation [31], as an intermediate facility. To our best knowledge, this is the first time that cost volumes are *directly* used for motion representation in an action recognition model.

Cost volume construction. We construct cost volumes on top of the low-level feature maps between consecutive frames. Given a pair of feature maps $\mathbf{F}_t$ and $\mathbf{F}_{t+1}$, we can construct a cost volume $\mathbf{C}_t \in \mathbb{R}^{H \times W \times (2\Delta H+1) \times (2\Delta W+1)}$ by matching each point with its neighbors in a window of the size $(2\Delta H + 1) \times (2\Delta W + 1)$. Elements in the cost volume $\mathbf{C}_t$ are the matching similarities. Particularly, in our construction, $\mathbf{C}_t(i, j, \delta i, \delta j)$ is the cosine similarity between $\mathbf{f}_t(i, j)$ and $\mathbf{f}_{t+1}(i + \delta i, j + \delta j)$, which are respectively the low-level feature vectors at location (i, j) and $(i + \delta i, j + \delta j)$.

FlowNet [8] also adopts a similar construction for the correlation layer. Our formulation, however, differs essentially in two ways: 1) The elements of the correlation maps in FlowNet are *unnormalized* dot products between feature vectors instead of cosine similarity. 2) The correlation maps in FlowNet are combined with appearance feature maps when fed to the next layer as input. Such differences suggest that the construction in FlowNet preserves more appearance information, while our construction focuses on motion information, which will be further strengthened in the next step, namely *cost volume processing*.

Cost volume processing. On top of a cost volume $\mathbf{C}_t$, we will further derive a lower dimensional representation to capture the motion information more concisely. A natural way is to compute a point-wise motion field by associating the feature points in one frame to those in the next, using a winner-take-all (WTA) assignment as in [31]. However, we argue that this way does not necessarily result in a sufficient characterization of the motion. On a low-resolution feature map (instead of a high-resolution image), a considerable portion of the displacements are sub-pixel. Hence, hard correspondence between feature points may lead to erroneous estimates. Moreover, as efficiency is a key goal in our design, it is undesirable to introduce heavy post-processing stages to refine the estimates. We will show in an ablative study that the representation derived from hard assignment yields inferior performance (see Sec. 4).

Instead, we propose a method based on soft assignment. Specifically, we introduce a displacement map $\mathbf{V}_t \in \mathbb{R}^{H \times W \times 2}$ to capture the motion from time t to $t+1$. To simplify the notation, we omit the time index t in the following discussion. In this map, each location (i,j) is associated with a 2-dimensional vector $\mathbf{v}_{i,j} = (v^y_{i,j}, v^x_{i,j})$ that represents the displacement at that location. Particularly, $v^y_{i,j}$ and $v^x_{i,j}$ respectively represent the displacement along vertical and horizontal directions. We compute the displacement map $\mathbf{V}_t$ as follows:

$$v^y_{i,j} = \sum_{\delta i = -\Delta H}^{\Delta H} \rho^y_{(i,j)}(\delta i) \cdot \delta i,$$
$$v^x_{i,j} = \sum_{\delta j = -\Delta W}^{\Delta W} \rho^x_{(i,j)}(\delta j) \cdot \delta j. \tag{1}$$

Here, the coefficients $\rho^y_{(i,j)}$ and $\rho^x_{(i,j)}$ are determined based on the cost volume as

$$\rho^y_{(i,j)}(\delta i) = \frac{\sum_{\delta j'} \exp(c_{i,j}(\delta i, \delta j')/\tau)}{\sum_{\delta i'} \sum_{\delta j'} \exp(c_{i,j}(\delta i', \delta j')/\tau)},$$
$$\rho^x_{(i,j)}(\delta j) = \frac{\sum_{\delta i'} \exp(c_{i,j}(\delta i', \delta j)/\tau)}{\sum_{\delta i'} \sum_{\delta j'} \exp(c_{i,j}(\delta i', \delta j')/\tau)}. \tag{2}$$

Here, $c_{i,j}(\delta i, \delta j) = \mathbf{C}_t(i, j, \delta i, \delta j)$. This computation can be understood as computing the *expected displacement* at each point based on a distribution over correspondences. The temperature coefficient τ controls the concentration of the distribution. When $\tau \to 0$, the probability mass will be concentrated on the point with highest similarity, and this scheme reduces to winner-take-all assignment. The computation of $\rho^y_{(i,j)}$ and $\rho^x_{(i,j)}$ can be accomplished by softmax transforms along the vertical and horizontal dimensions. With these assignment weights, the displacement map $\mathbf{V}_t$ can be obtained by 2D convolution.

After obtaining the displacement maps, we feed the maps as input feature maps to the subsequent convolution layers in this branch for high-level feature extraction. The result is a 1024-dimensional feature vector representing the apparent motion information of the input frame sequence. The parameters in this branch are end-to-end trained together with other branches using video action labels.

3.4. The Appearance Change Branch

Not all changes observed in a video can be explained by apparent motion. Other factors such as the inherent changes of an object's appearance or the variation in illumination may also cause changes in video frames. In previous work, such changes are often simply captured by the differences between consecutive frames, *e.g.* RGB-Diff [30]. This way, however, will mix the appearance changes with apparent motion, thus going against our desire to disentangle them into different components. In this work, we take a different approach. Given a pair of feature maps from consecutive frames $\mathbf{F}_t$ and $\mathbf{F}_{t+1}$, we first warp $\mathbf{F}_t$ according to $\mathbf{V}_t$, the estimated motion field from the apparent motion branch, resulting in a warped feature map as $\mathbf{F}'_{t+1} = \mathcal{W}(\mathbf{F}_t, \mathbf{V}_t)$. The warping is done with bilinear interpolation. The features at those locations whose source points are outside the image domain are set to zeros. Finally, we compute the difference between the warped feature map $\mathbf{F}'_{t+1}$ and $\mathbf{F}_{t+1}$ as $\mathbf{F}_{t+1} - \mathbf{F}'_{t+1}$. We call the resultant difference map as *warped differences*, which are used as the representation of the appearance changes.

The warped differences are fed into subsequent layers in the similar manner of the displacement maps in the apparent motion branch. We again achieve a 1024-dimensional feature vector characterizing the appearance change information to be fused into the final representation.

4. Experimental Results

To evaluate the proposed network architecture, we conduct action recognition experiments on two benchmark datasets, with in-depth study of the components in the architecture to verify our design principles. We also provide visualization of the learned motion representation as a critical part of the final representation and discuss its relationship with conventional optical flows.

4.1. Experimental Settings

Dataset. We conduct experiments on two action recognition benchmark datasets: UCF101 [22] and Kinetics [16]. The UCF101 dataset contains 13,320 videos divided into 101 action classes. The videos are collected from the Internet. We follow the evaluation scheme of the THUMOS13 challenge [10] and adopt the three training/testing splits for evaluation. The proportions of videos in the training and testing sets are around 75% and 25%, respectively. For this dataset, the top-1 action recognition accuracy is used as the evaluation metric to compare different approaches. The

Kinetics human action dataset [16] contains over 300,000 videos from 400 human action classes. The videos are collected with textual queries on the YouTube website. Image based CNN models are firstly used to roughly filter the negative samples. For each video a 10-second clip is extracted and assigned a manually annotated category label. Due to the inaccessibility of some videos on YouTube, our version of the Kinetics dataset contains 240436, 19796 and 38685 clips in the training, validation, and testing subset, respectively. On this dataset, we measure the action recognition performance with respect to top-1 and top-5 recognition accuracy.

Implementation details. We use stochastic gradient descent algorithm to train our neural network models. The mini-batch size is set to 256 and the momentum is set to 0.9. On UCF101, we use a small initial learning rate of 0.001 for the static appearance branch, and decrease by $\frac{1}{10}$ every 1,500 iterations. The training procedure stops after 4500 iterations. For the motion branch and the appearance change branch, we use a larger learning rate of 0.005, which is decreased by a factor of $\frac{1}{10}$ after 10,000 and 16,000 iterations. The training stops after 18,000 iterations. On Kinetics, a learning rate initialized with 0.01 is decreased by $\frac{1}{10}$ every 40,000 iterations. The whole procedure takes 110,000 iterations. We use gradient clipping of 20 to avoid exploding gradient at early stage. All the experiments are run on the Caffe toolbox [14]. To facilitate convergence, we follow the practices introduced in [30]: (1) cross-modality pre-training: the weights of all branches are initialized with ImageNet [20] pre-trained BNInception [12]; (2) regularization techniques including partial BN (for the experiments in UCF101 only), and an additional dropout layer after the `global_pool` layer. For more details on other design parameters of the network architecture, please refer to the supplementary materials.

4.2. Ablation Study

We justify the performance of our proposed design by conducting an ablation study. Also, we hope this study would provide useful insights for future design of video representations. Particularly, we first evaluate the recognition performance when separately using each branch for action recognition. Then, as discussed in Sec. 1, we investigate whether the learning of short term dynamics representation discussed in this work can benefit from jointly learning with long term information. Finally we analyze the choices of fusing representations from branches. All experiments in this ablation study are performed on the split 1 of UCF101 dataset.

Modeling static appearance with 3D pooling. To verify the effect of stabilization for 3D pooling, we first compare the 3D-Pool architecture with original single-frame 2D spatial ConvNet using RGB frames as input. From Table 1, we can see that the insertion of temporal pooling improves the single-frame model by 2%, which is a significant gain for UCF101. By increasing the number of input frames, consistent improvement can be clearly observed in Table 2. In the following experiments, the number is set to be 8 to strike a balance between performance and runtime cost since processing more than 8 frames provides marginal performance gain while linearly increasing the processing time for one input.

Method	# frames	Accuracy
2DConv+2DPool	1	84.5%
3DConv [3]	64	84.5%
2DConv+3DPool (late pool)	8	85.9%
2DConv+3DPool (gradual pool)	8	86.5%

Table 1. Comparison of the network architectures for the static appearance branch on the UCF101 split 1. In the lower half, we compare two schemes for adding 3D pooling into the network.

# of frames	1	2	4	8	16
Accuracy	84.5%	85.4%	85.6%	86.8%	87.4%

Table 2. Recognition accuracy with different frame numbers on the UCF101 split 1.

When only trained on the UCF101 dataset, our model also achieves a better result than the I3D model with RGB input [3]. This observation demonstrates that with parameter-free 3D pooling operation, our method can be more data efficient on smaller scale dataset like UCF101.

To assess the effect of inserting 3D pooling operations after different layers, we experiment with two types of pooling strategies: (1) Late 3DPool, which only has one global temporal pooling after the network produces its final features. In other words, the temporal dimension is preserved and only fused in the highest level. (2) Gradual 3DPool, which is to do temporal pooling after `pool2`, `inception3c`, and `inception4e`. The results are shown in the lower half of Table 1. Gradual 3DPool performs better than Late 3DPool. This suggests that introducing 3D pooling operations with small window sizes but at multiple positions can help better preserve the appearance information and produce better feature representations.

Modeling apparent motion with cost volume processing. Here we explore the design choices of cost volume construction. We leverage the low-level feature map after the `conv1` layer in the base network, and vary the stride of convolution. We also experiment with different assignment methods. The performance of these settings is illustrated in Table 3. First we find the displacement map produced by the soft assignment method is better than the hard assignment, denoted as "WTA". Also the higher resolution of the input feature map leads to better quality of the motion representation.

Method	Accuracy	FPS
TV-L1 [34]	92.0%	15
FlowNet [8]	86.8%	52
FlowNet 2.0 [11]	90.4%	8
MotionNet [38]	87.5%	120
Hidden Two-Stream [38]	89.8%	120
Cost volume + WTA	86.4%	-
Disp. Map (112 × 112)	88.2%	190
Disp. Map (224 × 224)	89.6%	53

Table 3. Comparison of accuracy and efficiency between different motion representations on UCF101 split 1.

Warping method	Input	Accuracy
No warping	RGB	84.0%
No warping	Feature	84.8%
TV-L1 warping	RGB	85.1%
Disp. Map warping	RGB	80.0%

Table 4. Comparison between different warping strategies and warped inputs.

Branch	w/o TSN	w/ TSN
Static Appearance	86.5%	87.3%
Displacement Map	80.8%	83.4%
Full model	89.5%	91.2%

Table 5. Comparison between settings with and without long term information in the learning. The experiments are conducted on UCF101 split 1.

Static App.	Motion		App. change		Accuracy
	TV-L1	Disp. Map	RGB-Diff	Warped-Diff	
✓	✓				94.0%
✓	✓		✓		93.8%
✓	✓			✓	94.1%
✓		✓			91.2%
✓		✓	✓		91.5%
✓		✓		✓	91.3%

Table 6. The comparison of results by fusion from different branches. The results are reported on UCF101 split 1.

We also compare with other optical flow based two-stream approaches in terms of both recognition accuracy and running efficiency. The results are summarized in Table 3. Runtime speed is tested on the server using one NVIDIA TITAN X GPU. The results show that our model is much faster than traditional optical flow based methods. It also shows that our approach can achieve competitive or even superior results with optical flow based methods such as FlowNet 2.0 [11] and MotionNet in [38] with a FPS as high as 190.

Modeling appearance changes. We study different settings of warped differences in Table 4. We observe that RGB difference warped by TV-L1 achieves higher results than naive RGB difference. This verifies our hypothesis that motion and appearance change should be decoupled. Then we use the motion field estimated by the cost volume for the guidance of warping. In experiments we find although the cost volume motion field is noisy, it can still help produce reasonable result for the warping. Further in the following discussion of representation fusion, we will show that such representation provides complementary information and contributes to improved recognition accuracy.

Combination with learning long-term dynamics Here we perform a series of experiments to compare the settings with or without long-term modeling techniques such as TSN [30]. The results are summarized in Table 5. We observe that long term information still benefits the learning of short term representations. This also conforms our idea of dividing video contents into short term dynamics and long term temporal structures. In the following experiments the models are all trained with TSN [30].

Fusing representation from branches. Due to the current disentangling design, we are able to investigate the contribution of each branch to the final recognition performance and the choice of different branch combinations. The results are shown in Table 6. We first combine only the static appearance and apparent motion branches. Then we add the warped difference based appearance branch. For comparison, we also experiment with using non-warped RGB difference for input. The results suggest that: 1) The best performance is achieved when combining all three branches, which corroborates our disentangling of the components. 2) The warped difference can also help improve the performance of traditional optical flow based methods, showing the validity of the appearance change component. 3) Currently, although the proposed architecture is much faster than the traditional optical flow based method, there is still a performance gap in terms of accuracy.

4.3. Comparison with the State-of-the-Art

We compare the action recognition performance of our approach with other state-of-the-art methods that take RGB frames as input. The results on UCF101 and Kinetics are shown in Table 7 and Table 8 respectively. First we notice that our model achieves comparable or superior results on these benchmark datasets. When trained and tested on the UCF101 dataset, our model outperforms previous models by a healthy margin. On the Kinetics dataset, our model outperforms the heavy 3D convolution based models while remaining very efficient.

Since it is expected that the 3D convolution based methods benefit more from larger datasets, we also benchmarked our method on UCF-101 using models pre-trained on Kinetics. The improvement demonstrates that our method is also able to scale with more training data.

Method	Pre-train	Accuracy
C3D (1 nets) [23]	Sports-1M	82.3%
C3D (3 nets) [23]	Sports-1M	85.2%
Pseudo-3D ResNet [18]	ImageNet+Sports-1M	88.6%
RGB-I3D [3]	ImageNet	84.5%
RGB+EMV [35]	ImageNet	86.4%
TSN (RGB) [30]	ImageNet	85.7%
TSN (RGB+RGB-Diff) [30]	ImageNet	91.0%
Ours	ImageNet	91.8%
Ours	ImageNet+Kinetics	95.9%

Table 7. Comparison with state-of-the-art on the UCF101 dataset. We compare with other methods taking only RGB sequences as input. The accuracy is averaged over three splits.

Method	Accuracy(Top-1/Top-5)
Two-Stream (RGB) [16]	56.0% / 77.3%
3D-ConvNet [16]	56.1% / 79.5%
RGB-I3D [3]	68.4% / 88.0%
Ours	71.5% / 89.9%

Table 8. Comparison with state-of-the-art using only RGB frames on the Kinetics dataset.

4.4. Runtime Analysis

To quantify the efficiency of our method, we test the runtime speed for each of the proposed components and the overall architecture using an NVIDIA TITAN X GPU. The size of displacement map is 224×224. The runtime profiling is summarized in Table 9. Note that the runtime for the motion and appearance change branches is divided into two parts: 1) the construction of displacement map takes 15.4 ms per frame and is shared between the two branches and 2) the classification phase is individually executed by the following subnetworks. The whole system can achieve a real-time performance at over 40 FPS.

Branch	Static App.	Motion	App. Change	Overall
ms/frame	1.19	(15.4)+3.33	(15.4)+4.23	24.1
FPS	840	53	51	41

Table 9. Runtime analysis for the proposed individual components and the overall architecture. Numbers in brackets indicate the time cost for displacement map processing which is shared between branches.

4.5. Visualization

In this section, we present a qualitative study by visualizing the intermediate results. As shown in Figure 5, the displacement map contains much more noise than the optical flow calculated by TV-L1 [34] because the optimization objective in TV-L1 includes a regularization term which favors smoothness while in the cost-volume formulation we do not enforce such constraint. However, the following classification subnet is expected to filter out the distraction of noise. We also compare the method between the naive RGB difference and the motion-warped RGB difference. Compared with RGB difference, motion warped RGB difference focuses more on the change of apperance, reflected by the more distinctive edges of moving objects. For more results and further discussion, please refer to the supplementary materials.

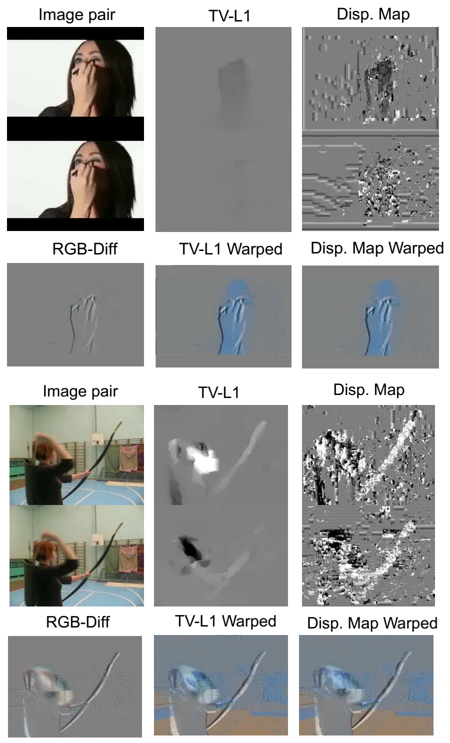

Figure 5. Visualization results of several representations. Upper left: input image pair; Upper middle: optical flow produced by TV-L1; Upper right: displacement map generated by cost volume. Lower left: RGB-Diff; Lower middle: RGB-Diff warped by TV-L1; Lower right: RGB-Diff warped by displacement map.

5. Conclusions

In this work, we propose a new network architecture for deep learning based action recognition. The approach is based on the idea of disentangling different components in the dynamics within an input frame sequence, including static appearance, apparent motion, and appearance changes. We believe this work will provide a new perspective for action recognition by lifting the reliance on optical flow in achieving good performance. The proposed network does not need any optical flow for input or supervision and provides an unified and efficient representation for the dynamics, which is verified by extensive experiments on benchmark datasets.

Acknowledgement This work is partially supported by the Big Data Collaboration Research grant from SenseTime Group (CUHK Agreement No. TS1610626), and the Early Career Scheme (ECS) of Hong Kong (No. 24204215).

References

[1] C. Bailer, B. Taetz, and D. Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 4015–4023, 2015. 2

[2] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3034–3042, 2016. 3

[3] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1, 2, 6, 8

[4] J. W. Davis. Hierarchical motion history images for recognizing human motion. In *IEEE Workshop on Detection and Recognition of Events in Video*, 2001. 3

[5] A. Diba, V. Sharma, and L. Van Gool. Deep temporal linear encoding networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[6] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *The 2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance.*, pages 65–72. IEEE, 2005. 2

[7] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 3, 4

[8] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015. 3, 4, 7

[9] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, 2016. 1, 2

[10] H. Idrees, A. R. Zamir, Y.-G. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah. The thumos challenge on action recognition for videos in the wild. *Computer Vision and Image Understanding*, 155:1–23, 2017. 5

[11] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 3, 7

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, 2015. 3, 6

[13] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013. 2

[14] 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 *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014. 6

[15] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014. 2

[16] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 1, 2, 5, 6, 8

[17] D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with fisher vectors on a compact feature set. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 1817–1824, 2013. 2

[18] Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2, 8

[19] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1164–1172, 2015. 2

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 6

[21] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Neural Information Processing Systems (NIPS)*, pages 568–576, 2014. 1, 2, 4

[22] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. 2012. 1, 2, 5

[23] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015. 1, 2, 4, 8

[24] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 2

[25] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3169–3176. IEEE, 2011. 1, 2

[26] H. Wang and C. Schmid. Action recognition with improved trajectories. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 3551–3558, 2013. 2

[27] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4305–4314, 2015. 4

[28] L. Wang, Y. Xiong, D. Lin, and L. Van Gool. Untrimmednets for weakly supervised action recognition and detection.

In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[29] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015. 1

[30] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision (ECCV)*, pages 20–36. Springer, 2016. 2, 3, 4, 5, 6, 7, 8

[31] J. Xu, R. Ranftl, and V. Koltun. Accurate Optical Flow via Direct Cost Volume Processing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 4, 5

[32] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018. 2

[33] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, 2015. 2

[34] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l 1 optical flow. *Pattern Recognition*, pages 214–223, 2007. 2, 7, 8

[35] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang. Real-time action recognition with enhanced motion vector cnns. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2718–2726, 2016. 1, 3, 8

[36] M. Zhao, T. Li, M. A. Alsheikh, Y. Tian, H. Zhao, A. Torralba, and D. Katabi. Through-wall human pose estimation using radio signals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[37] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin. Temporal action detection with structured segment networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2

[38] Y. Zhu, Z. Lan, S. Newsam, and A. G. Hauptmann. Hidden two-stream convolutional networks for action recognition. *arXiv preprint arXiv:1704.00389*, 2017. 1, 3, 7