Hidden Two-Stream Convolutional Networks for Action Recognition

Yi Zhu[1], Zhenzhong Lan[2], Shawn Newsam[1], and Alexander Hauptmann[2]

[1] University of California at Merced, Merced CA 95343, USA
{yzhu25,snewsam}@ucmerced.edu
[2] Carnegie Mellon University, Pittsburgh PA 15213, USA
{lanzhzh,alex}@cs.cmu.edu

Abstract. Analyzing videos of human actions involves understanding the temporal relationships among video frames. State-of-the-art action recognition approaches rely on traditional optical flow estimation methods to pre-compute motion information for CNNs. Such a two-stage approach is computationally expensive, storage demanding, and not end-to-end trainable. In this paper, we present a novel CNN architecture that implicitly captures motion information between adjacent frames. We name our approach hidden two-stream CNNs because it only takes raw video frames as input and directly predicts action classes without explicitly computing optical flow. Our end-to-end approach is 10x faster than its two-stage baseline. Experimental results on four challenging action recognition datasets: UCF101, HMDB51, THUMOS14 and ActivityNet v1.2 show that our approach significantly outperforms the previous best real-time approaches.

Keywords: Action recognition · Optical flow · Unsupervised learning.

1 Introduction

The field of human action recognition has advanced rapidly over the past few years. We have moved from manually designed features [3, 23] to learned convolutional neural network (CNN) features [11, 21]; from encoding appearance information to encoding motion information [19]; and from learning local features to learning global video features [13, 25]. The performance has continued to soar higher as we incorporate more of the steps into an end-to-end learning framework. Nevertheless, current state-of-the-art CNN structures still have difficulty in capturing motion information directly from video frames. Instead, traditional local optical flow estimation methods are used to pre-compute motion information for the CNNs [19]. This two-stage pipeline, first compute optical flow and then learn the mapping from optical flow to action labels, is sub-optimal for the following reasons:

– The pre-computation of optical flow is time consuming and storage demanding compared to the CNN step.

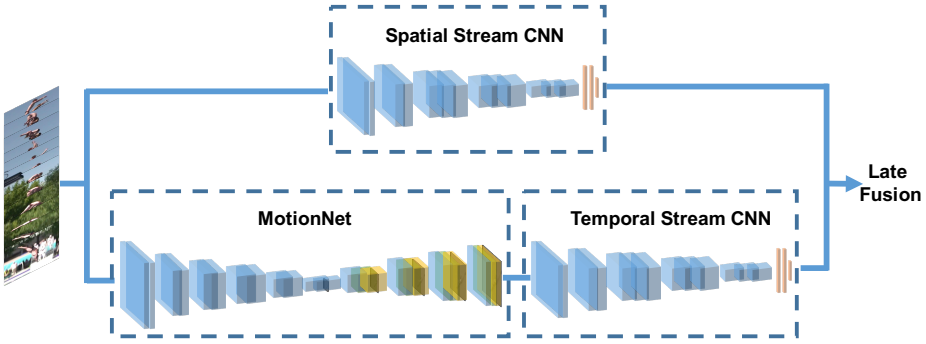

Fig. 1: Illustration of proposed hidden two-stream networks. MotionNet takes consecutive video frames as input and estimates motion. Then the temporal stream CNN learns to project the motion information to action labels. Late fusion is performed through the weighted averaging of the prediction scores of the temporal and spatial streams. Both streams are end-to-end trainable.

– Traditional optical flow estimation is completely independent of the final tasks like action recognition and is therefore potentially sub-optimal.

To solve the above problems, researchers have proposed various methods other than optical flow to capture motion information in videos. For example, new representations like motion vectors [27,33] and RGB image difference [25] or architectures like recurrent neural networks (RNN) [16] and 3D CNNs [17,21,22, 28]. However, most of these are not as effective as optical flow for human action recognition[3]. Therefore, in this paper, we aim to address the above mentioned problems in a more direct way. We adopt the end-to-end CNN approach to learn optical flow so that we can avoid costly computation and storage and obtain task-specific motion representations. However, we face many challenges to learn such a motion estimation model:

– We need to train the models without supervision. The ground truth flow required for supervised training is usually not available except for limited synthetic data [34, 36, 37].
– We need to train our optical flow estimation models from scratch. The models (filters) learned for optical flow estimation tasks are very different from models (filters) learned for other vision tasks [6, 14, 29].
– We cannot simply use the traditional optical flow estimation loss functions. We are concerned chiefly with how to learn an optimal motion representation for video action recognition.

To address these challenges, we first train a CNN with the goal of generating optical flow from a set of consecutive frames. Through a set of specially designed operators and unsupervised loss functions, our new training step can generate

[3] Detailed comparisons can be found in the supplementary material.

optical flow that is similar to that generated by one of the best traditional methods [32]. As illustrated in the bottom of Figure 1, we call this network MotionNet. Given the MotionNet, we concatenate it with a temporal stream CNN that maps the estimated optical flow to the target action labels. We then fine-tune this stacked temporal stream CNN in an end-to-end manner with the goal of predicting action classes for the input frames. We call our new approach hidden two-stream networks as it implicitly generates motion information for action recognition. Our contributions include:

- Our method is both computationally and storage efficient. It is around 10x faster than its two-stage baseline, and we do not need to store the pre-computed optical flow.
- Our method outperforms previous real-time approaches on four challenging action recognition datasets by a large margin.
- The proposed MotionNet is flexible in that it can be directly concatenated with other video action recognition frameworks [1,16,22,35] to improve their efficiency.
- We demonstrate the generalizability of our end-to-end learned optical flow by showing promising results on four optical flow benchmarks without fine-tuning.

2 Related Work

Significant advances in understanding human activities in video have been made over the past few years. Initially, traditional handcrafted features such as Improved Dense Trajectories (IDT) [23] dominated the field of video analysis for several years. Despite their superior performance, IDT and its improvements are computationally formidable for real applications. CNNs [11, 21], which are often several orders of magnitude faster than IDTs, performed much worse than IDTs in the beginning. This inferior performance is mostly because CNNs have difficulty in capturing motion information among frames. Later on, two-stream CNNs [19] addressed this problem by pre-computing the optical flow using traditional optical flow estimation methods [32] and training a separate CNN to encode the pre-computed optical flow. This additional stream (a.k.a., the temporal stream) significantly improved the accuracy of CNNs and finally allowed them to outperform IDTs on several benchmarks. These accuracy improvements indicate the importance of temporal motion information for action recognition as well as the inability of existing CNNs to capture such information.

However, compared to the CNN, the optical flow calculation is computationally expensive. It is the major speed bottleneck of the current two-stream approaches. As an alternative, Zhang *et al.* [33] proposed to use motion vectors to replace the more precise optical flow. This simple improvement brought more than 20x speedup compared to the traditional two-stream approaches. However, this speed improvement came with an equally significant accuracy drop. The encoded motion vectors lack fine structures, and contain noisy and inaccurate motion patterns, leading to much worse accuracy compared to the more precise

optical flow [32]. These weaknesses are fundamental and can not be improved. Another more promising approach is to learn to predict optical flow using supervised CNNs, which is closer to our approach. Ng. *et al.* [15] used optical flow calculated by traditional methods as supervision to train a network to predict optical flow. This method avoids the pre-computation of optical flow at inference time and greatly speeds up the process. However, the quality of the optical flow calculated by this approach is limited by the quality of the traditional flow estimation, which again limits its potential on action recognition. Ilg *et al.* [8] use a network trained on synthetic data where ground truth flow exists. The ability of synthetic data to represent the complexity of real data is very limited. Ilg *et al.* [8] actually show that there is a domain gap between real data and synthetic data. To address this gap, they simply grow the synthetic data to narrow the gap. The problem with this solution is that it may not work for other datasets and it is not feasible to do this for all datasets. Our work addresses the optical flow estimation problem in a much more fundamental and promising way. We predict optical flow on-the-fly using CNNs, thus addressing the computation and storage problems. And we perform unsupervised pre-training on real data, thus addressing the domain gap problem.

Besides the computational problem, traditional optical flow estimation is completely independent of the high-level final tasks like action recognition and is therefore potentially sub-optimal. However, our approach is end-to-end optimized. It is important to distinguish between these two ways of introducing motion information to the encoding CNNs. Although optical flow is currently being used to represent the motion information in the videos, we do not know whether it is an optimal representation. There might be an underlying motion representation that is better than optical flow. In fact, a recent work [30] demonstrated that fixed flow estimation is not as good as task-oriented flow for general computer vision tasks. Hence, we believe that our end-to-end learning framework will help us extract better motion representations than traditional optical flow for action recognition. However, for notational convenience, we still refer our learned motion representation as optical flow.

3 Hidden Two-Stream Networks

In this section, we describe our proposed hidden two-stream networks in detail. We first introduce our unsupervised network for optical flow estimation along with employed good practices in Section 3.1. We name it MotionNet. In Section 3.2, we stack the temporal stream network upon MotionNet to allow end-to-end training. Finally, we introduce the hidden two-stream CNNs in Section 3.3 which combines our stacked temporal stream with a spatial stream.

3.1 Unsupervised Optical Flow Learning

We treat optical flow estimation as an image reconstruction problem [31]. Given a frame pair, we hope to generate the optical flow that allows us to reconstruct

one frame from the other. Formally, taking a pair of adjacent frames I_1 and I_2 as input, our CNN generates a flow field V. Then using the predicted flow field V and I_2, we get the reconstructed frame I_1' using backward warping, i.e., $I_1' = \mathcal{T}[I_2, V]$, where $\mathcal{T}$ is the inverse warping function. Our goal is to minimize the photometric error between I_1 and I_1'. The intuition is that if the estimated flow and the next frame can be used to reconstruct the current frame, then the network should have learned useful representations of the underlying motions.

MotionNet Our MotionNet is a fully convolutional network, consisting of a contracting part and an expanding part. The contracting part is a stack of convolutional layers and the expanding part is a chain of combined convolutional and deconvolutional layers. The details of our network can be seen in the supplementary material. We describe the challenges and proposed good practices to learn better motion representation for action recognition below.

First, we design a network that focuses on small displacement motion. For real data such as YouTube videos, we often encounter the problem that foreground motion (human actions of interest) is small, but the background motion (camera motion) is dominant. Thus, we adopt 3×3 kernels throughout the network to detect local, small motions. Besides, we keep the high frequency image details for later stages. Our first two convolutional layers do not use striding. We use strided convolution instead of pooling for image downsampling because pooling is shown to be harmful for dense per-pixel prediction tasks.

Second, our MotionNet computes multiple losses at multiple scales. Due to the skip connections between the contracting and expanding parts, the intermediate losses can regularize each other and guide earlier layers to converge faster to the final objective. We explore three loss functions that help us to generate better optical flow. These loss functions are as follows.

– A standard pixelwise reconstruction error function, which is calculated as:

$$L_{\text{pixel}} = \frac{1}{hw}\sum_{i}^{h}\sum_{j}^{w}\rho(I_1(i,j) - I_2(i + V_{i,j}^x,\ \ j + V_{i,j}^y)). \tag{1}$$

The V^x and V^y are the estimated optical flow in the horizontal and vertical directions. The inverse warping $\mathcal{T}$ is performed using a spatial transformer module [9]. Here we use a robust convex error function, the generalized Charbonnier penalty $\rho(x) = (x^2 + \epsilon^2)^\alpha$, to reduce the influence of outliers. h and w denote the height and width of images I_1 and I_2.

– A smoothness loss that addresses the aperture problem that causes ambiguity in estimating motions in non-textured regions. It is calculated as:

$$L_{\text{smooth}} = \rho(\nabla V_x^x) + \rho(\nabla V_y^x) + \rho(\nabla V_x^y) + \rho(\nabla V_y^y). \tag{2}$$

∇V_x^x and ∇V_y^x are the gradients of the estimated flow field V^x in each direction. Similarly, ∇V_x^y and ∇V_y^y are the gradients of V^y. The generalized Charbonnier penalty $\rho(x)$ is the same as in the pixelwise loss.

– A structural similarity (SSIM) loss function [26] that helps us to learn the structure of the frames. SSIM is a perceptual quality measure. Given two $K \times K$ image patches I_{p1} and I_{p2}, it is calculated as

$$\text{SSIM}(I_{p1}, I_{p2}) = \frac{(2\mu_{p1}\mu_{p2} + c_1)(2\sigma_{p1p2} + c_2)}{(\mu_{p1}^2 + \mu_{p2}^2 + c_1)(\sigma_{p1}^2 + \sigma_{p2}^2 + c_2)}. \tag{3}$$

Here, μ_{p1} and μ_{p2} are the mean of image patches I_{p1} and I_{p2}, σ_{p1} and σ_{p2} are the variance of image patches I_{p1} and I_{p2}, and σ_{p1p2} is the covariance of these two image patches. c_1 and c_2 are two constants to stabilize division by a small denominator. In our experiments, K is set to 8 and c_1 and c_2 are 0.0001 and 0.001, respectively.
In order to compare the similarity between two images I_1 and I_1', we adopt a sliding window approach to partition the images into local patches. The stride for the sliding window is set to 8 in both the horizontal and vertical directions. Hence, our SSIM loss function is defined as:

$$L_{\text{ssim}} = \frac{1}{N}\sum_{n}^{N}(1 - \text{SSIM}(I_{1n}, I_{1n}')). \tag{4}$$

where N is the number of patches we can extract from an image given the sliding stride of 8, n is the patch index. I_{1n} and I_{1n}' are two corresponding patches from original image I_1 and the reconstructed image I_1'. Our experiments show that this simple strategy significantly improves the quality of our estimated flows. It forces our MotionNet to produce flow fields with clear motion boundaries.

Hence, the loss at each scale s is a weighted sum of the pixelwise reconstruction loss, the piecewise smoothness loss, and the region-based SSIM loss,

$$L_s = \lambda_1 \cdot L_{\text{pixel}} + \lambda_2 \cdot L_{\text{smooth}} + \lambda_3 \cdot L_{\text{ssim}} \tag{5}$$

where λ_1, λ_2, and λ_3 weight the relative importance of the different metrics during training. Since we have predictions at five scales (flow2 to flow6) due to five expansions in the decoder, the overall loss of MotionNet is a weighted sum of loss L_s:

$$L_{\text{all}} = \sum_{s=1}^{5} \delta_s L_s \tag{6}$$

where the δ_s are set to balance the losses at each scale and are numerically of the same order. We describe how we determine the values of these weights in the supplementary materials.

Third, unsupervised learning of optical flow introduces artifacts in homogeneous regions because the brightness assumption is violated. We insert additional convolutional layers between deconvolutional layers in the expanding part to yield smoother motion estimation. We also explored other techniques in the literature, like adding flow confidence and multiplying by the original color images [8] during expanding. However, we did not observe any improvements.

In Section 5.1, we conduct an ablation study to demonstrate the contributions of each of these strategies. Though our network structure is similar to a concurrent work [8], MotionNet is fundamentally different from FlowNet2. First, we perform unsupervised learning while [8] performs supervised learning for optical flow prediction. Unsupervised learning allows us to avoid the domain gap between synthetic data and real data. Unsupervised learning also allows us to train the model for target tasks like action recognition in an end-to-end fashion even if the datasets of target applications do not have ground truth optical flow. Second, our network architecture is carefully designed to balance efficiency and accuracy. For example, MotionNet only has one network, while FlowNet2 has 5 similar sub-networks. The model footprints of MotionNet and FlowNet2 [8] are 170M and 654M, and the prediction speeds are 370fps and 25fps, respectively. We also present an architecture search in the supplementary materials to obtain deep insights in terms of the model trade-off between accuracy and efficiency.

3.2 Projecting Motion Features to Actions

Given that MotionNet and the temporal stream are both CNNs, we would like to combine these two modules into one stage and perform end-to-end training. There are multiple ways to design such a combination to project motion features to action labels. Here, we explore two ways, stacking and branching.

Stacking is the most straightforward approach and just places MotionNet in front of the temporal stream, treating MotionNet as an off-the-shelf flow estimator. Branching is more elegant in terms of architecture design. It uses a single network for both motion feature extraction and action classification. The convolutional features are shared between the two tasks. Due to space limitations, we show in the supplementary materials that stacking is more effective than branching. It achieves better action recognition performance while remaining complementary to the spatial stream. From now on, we choose stacking to project the motion features to action labels.

For stacking, we first need to normalize the estimated flows before feeding them to the encoding CNN. More specifically, as suggested in [19], we first clip the motions that are larger than 20 pixels to 20 pixels. Then we normalize and quantize the clipped flows to have a range between $0 \sim 255$. We find such a normalization is important for good temporal stream performance and design a new normalization layer for it.

Second, we need to determine how to fine tune the network, including which loss to use during the fine tuning. We explored different settings. (a) Fixing MotionNet, which means that we do not use the action loss to fine-tune the optical flow estimator. (b) Both MotionNet and the temporal stream CNN are fine-tuned, but only the action categorical loss function is computed. No unsupervised objective (5) is involved. (c) Both MotionNet and the temporal stream CNN are fine-tuned, and all the loss functions are computed. Since motion is largely related to action, we hope to learn better motion estimators by this multi-task way of learning. As will be demonstrated later in Section 4.2, model (c) achieves the best action recognition performance.

Third, we need to capture relatively long-term motion dependencies. We accomplish this by inputting a stack of multiple consecutive flow fields. Simonyan and Zisserman [19] found that a stack of 10 flow fields achieves a much higher accuracy than only using a single flow field. To make fair comparison, we also fix the length of our input to be 11 frames to allow us to generate 10 optical flows.

3.3 Hidden Two-Stream Networks

We also show the results of combining our stacked temporal stream with a spatial stream. These results are important as they are strong indicators of whether our stacked temporal stream indeed learns complementary motion information or just appearance information.

Following the testing scheme of [19, 24], we evenly sample 25 frames/clips for each video. For each frame/clip, we perform 10x data augmentation by cropping the 4 corners and 1 center, flipping them horizontally and averaging the prediction scores (before softmax operation) over all crops of the samples. In the end, we fuse the two streams' scores with a spatial to temporal stream ratio of 1:1.5.

4 Experiments

4.1 Evaluation Datasets

We perform experiments on four widely used action recognition benchmarks, UCF101 [20], HMDB51 [12], THUMOS14 [5] and ActivityNet [7]. UCF101 is composed of realistic action videos from YouTube. It contains $13,320$ video clips distributed among 101 action classes. HMDB51 includes $6,766$ video clips of 51 actions extracted from a wide range of sources, such as online videos and movies. Both UCF101 and HMDB51 have a standard three-split evaluation protocol and we report the average recognition accuracies over the three splits. THUMOS14 and ActivityNet are large-scale video datasets for action recognition and detection, which contain long untrimmed videos. THUMOS14 has 101 action classes. It includes a training set, validation set, test set and background set. We don't use the background set in our experiments. We use 13,320 training and 1,010 validation videos for training and report the performance on 1,574 test videos. For ActivityNet, we use its 1.2 version which has 100 action classes. Following the standard evaluation split, 4,819 training and 2,383 validation videos are used for training and 2,480 videos for testing.

4.2 Results

In this section, we evaluate our proposed framework on the first split of UCF101. We report the accuracy as well as the processing speed of the inference step in frames per second. The results are shown in Table 1. The implementation details are in the supplementary materials.

Top section of Table 1: Here we compare the performance of two-stage approaches. By two-stage, we mean optical flow is pre-computed, cached, and then

Table 1: Comparison of accuracy and efficiency. Top section: Two-stage temporal stream approaches. Middle Section: End-to-end temporal stream approaches. Bottom Section: Two-stream approaches.

Method	Accuracy (%)	fps
TV-L1 [32]	85.65	14.75
FlowNet [4]	55.27	52.08
FlowNet2 [8]	79.64	8.05
NextFlow [18]	72.2	42.02
Enhanced Motion Vectors [33]	79.3	390.7
MotionNet (2 frames)	84.09	48.54
ActionFlowNet (2 frames) [15]	70.0	200.0
ActionFlowNet (16 frames) [15]	83.9	–
Stacked Temporal Stream CNN (a)	83.76	169.49
Stacked Temporal Stream CNN (b)	84.04	169.49
Stacked Temporal Stream CNN (c)	84.88	169.49
Two-Stream CNNs [19]	88.0	14.3
Very Deep Two-Stream CNNs [24]	**90.9**	**12.8**
Hidden Two-Stream CNNs (a)	87.50	120.48
Hidden Two-Stream CNNs (b)	87.99	120.48
Hidden Two-Stream CNNs (c)	**89.82**	**120.48**

fed to a CNN classifier to project flow to action labels. For fair comparison, our MotionNet here is pre-trained on UCF101, but not fine-tuned using the action classification loss. It only takes frame pairs as input and outputs one flow estimate. The results show that our MotionNet achieves a good balance between accuracy and speed in this setting.

In terms of accuracy, our unsupervised MotionNet is competitive to TV-L1 while performing much better (4% $\sim$ 12% absolute improvement) than other methods of generating flows, including supervised training using synthetic data (FlowNet [4] and FlowNet2 [8]), and directly getting flows from compressed videos (Enhanced Motion Vectors [33]). These improvements are very significant in datasets like UCF101. In terms of speed, we are also among the best of the CNN based methods and much faster than TV-L1, which is one of the fastest traditional methods.

Middle section of Table 1: Here we examine the performance of end-to-end CNN based approaches. None of these approaches store intermediate flow information and thus run much faster than the two-stage approaches. If we compare the average running time of these approaches to the two-stage ones, we can see that the time spent on writing and reading intermediate results is almost 3x as much as the time spent on all other steps. Therefore, from an efficiency perspective, it is important to do end-to-end training and predict optical flow on-the-fly.

ActionFlowNet [15] is what we denote as a branched temporal stream. It is a multi-task learning model to jointly estimate optical flow and recognize actions.

The convolutional features are shared which leads to faster speeds. However, even the 16 frames ActionFlowNet performs 1% worse than our stacked temporal stream. Besides, ActionFlowNet uses optical flow from traditional methods as labels to perform supervised training. This indicates that during the training phase, it still needs to cache flow estimates which is computation and storage demanding for large-scale video datasets. Also the algorithm will mimic the failure cases of the classical approaches.

If we compare the way we fine-tune our stacked temporal stream CNNs, we can see that model (c) where we include all the loss functions to do end-to-end training, is better than the other models including fixing MotionNet weights (model (a)) and only using the action classification loss function (model (b)). These results show that both end-to-end fine-tuning and fine-tuning with unsupervised loss functions are important for stacked temporal stream CNN training.
Bottom section of Table 1: Here we compare the performance of two-stream networks by fusing the prediction scores from the temporal stream CNN with the prediction scores from the spatial stream CNN. These comparisons are mainly used to show that stacked temporal stream CNNs indeed learn motion information that is complementary to what is learned in appearance streams.

The accuracy of the single stream spatial CNN is 80.97%. We observe from Table 1 that significant improvements are achieved by fusing a stacked temporal stream CNN with a spatial stream CNN to create a hidden two-stream CNN. These results show that our stacked temporal stream CNN is able to learn motion information directly from the frames and achieves much better accuracy than spatial stream CNN alone. This observation is true even in the case where we only use the action loss for fine-tuning the whole network (model (b)). This result is significant because it indicates that our unsupervised pre-training indeed finds a better path for CNNs to learn to recognize actions and this path will not be forgotten in the fine-tuning process. If we compare the hidden two-stream CNNs to the stacked temporal stream CNNs, we can see that the gap between model (c) and model (a)/(b) widens. The reason may be because, without the regularization of the unsupervised loss, the networks start to learn appearance information. Hence they become less complementary to the spatial CNNs.

Finally, we can see that our models achieve very similar accuracy to the original two-stream CNNs. Among the two representative works we show, Two-Stream CNNs [19] is the earliest two-stream work and Very Deep Two-Stream CNNs [24] is the one we improve upon. Therefore, Very Deep Two-Stream CNNs [24] is the most comparable work. We can see that our approach is about 1% worse than Very Deep Two-Stream CNNs [24] in terms of accuracy but about 10x faster in terms of speed.

5 Discussion

5.1 Ablation Studies for MotionNet

Because of our specially designed loss functions and operators, our proposed MotionNet can produce high quality motion estimates, which allows us to achieve

Table 2: Ablation study of good practices employed in MotionNet.

Method	Small Disp	SSIM	CDC	Smoothness	MultiScale	Accuracy (%)
MotionNet	×	×	×	×	×	77.79
MotionNet	✓	✓	✓	✓	×	80.63
MotionNet	✓	✓	✓	×	✓	80.14
MotionNet	✓	✓	×	✓	✓	81.25
MotionNet	✓	×	✓	✓	✓	81.58
MotionNet	×	✓	✓	✓	✓	82.22
MotionNet	✓	✓	✓	✓	✓	**82.71**

promising action recognition accuracy. Here, we run an ablation study to understand the contributions of these components. The results are shown in Table 2. *Small Disp* indicates using a network that focuses on small displacements. *CDC* means adding an extra convolution between deconvolutions in the expanding part of MotionNet. *MultiScale* indicates computing losses at multiple scales.

First, we examine the importance of using a network structure that focuses on small displacement motions. We keep the other aspects of the implementation the same, but use a larger kernel size and stride in the beginning of the network. The accuracy drops from 82.71% to 82.22%. This drop shows that using smaller kernels with a deeper network indeed helps to detect small motions.

Second, we examine the importance of adding the SSIM loss. Without SSIM, the action recognition accuracy drops to 81.58%. This more than 1% performance drop shows that it is important to focus on discovering the structure of frame pairs.

Third, we examine the effect of removing convolutions between the deconvolutions in the expanding part of MotionNet. This strategy is designed to smooth the motion estimation. As can be seen in Table 2, removing extra convolutions brings a significant performance drop from 82.71% to 81.25%.

Fourth, we examine the advantage of incorporating the smoothness objective. Without the smoothness loss, we obtain a much worse result of 80.14%. This result shows that our real-world data is very noisy. Adding smoothness regularization helps to generate smoother flow fields by suppressing noise. This suppression is important for the following temporal stream CNNs to learn better motion representations for action recognition.

Fifth, we examine the necessity of computing losses at multiple scales during deconvolution. Without the multi-scale scheme, the action recognition accuracy drops to 80.63%. The performance drop shows that it is important to regularize the output at each scale in order to produce the best flow estimation in the end. Otherwise, we found that the intermediate representations during deconvolution may drift to fit the action recognition task, and not predict optical flow.

Finally, we explore a model that does not employ any of these practices. As expected, the performance is the worst, which is 4.94% lower than our full MotionNet.

Table 3: Evaluation of optical flow and action classification. For flow evaluation, lower error is better. For action recognition, higher accuracy is better.

Method	Sintel	KITTI2012	KITTI2015	Middlebury	UCF101
FlowNet2	**6.02**	**1.8**	**11.48**	0.52	81.97
TV-L1	10.46	14.6	47.64	**0.45**	**85.65**
MotionNet	11.93	7.5	30.65	0.91	84.88

5.2 Learned Optical Flow

In this section, we systematically investigate the effects of different motion estimation models for action recognition, as well as their flow estimation quality. We also show some visual examples to discover possible directions for future improvement. Here, we compare three optical flow models: TV-L1, MotionNet and FlowNet2. To quantitatively evaluate the quality of learned flow, we test the three models on four well received benchmarks, MPI-Sintel, KITTI 2012, KITTI 2015 and Middlebury. For action recognition accuracy, we report their performance on UCF101 split1. The results can be seen in Table 3. We use EPE (endpoint error) to evaluate MPI-Sintel, KITTI 2012 and Middlebury with lower being better. We use Fl (percentage of optical flow outliers) to evaulate KITTI 2015 with lower being better. We use classification accuracy to evaluate UCF101 with higher being better.

For flow quality, FlowNet2 generally performs better, except on Middlebury because it mostly contains small displacements. Our MotionNet has similar performance to TV-L1 on Sintel and Middlebury, and outperforms TV-L1 on KITTI 2012 and KITTI 2015. The result is encouraging because the KITTI benchmark contains real data (not synthetic), which indicates that the flow estimation from our MotionNet is robust and generalizable. In addition, although FlowNet2 ranks higher on optical flow benchmarks, it performs the worst on action recognition tasks. This interesting observation means that lower EPE does not always lead to higher action recognition accuracy. This is because EPE is a very simple metric based on L2 distance, which does not consider motion boundary preservation or background motion removal. This is crucial, however, for recognizing complex human actions.

We also show some visual samples in Figure 2 to help understand the effect of the quality of estimated flow fields for action recognition. The color scheme follows the standard flow field color coding in [8]. In general, the estimated flow fields from all three models look reasonable. MotionNet has lots of background noise compared to TV-L1 due to its global learning. This maybe the reason why it performs worse than TV-L1 for action recognition. FlowNet2 has very crisp motion boundaries, fine structures and smoothness in homogeneous regions. It is indeed a good flow estimator in terms of both EPE and visual inspection. However, it achieves much worse results for action recognition, 3.5% lower than TV-L1 and 2.9% lower than our MotionNet. Thus, which motion representation is best for action recognition remains an open question.

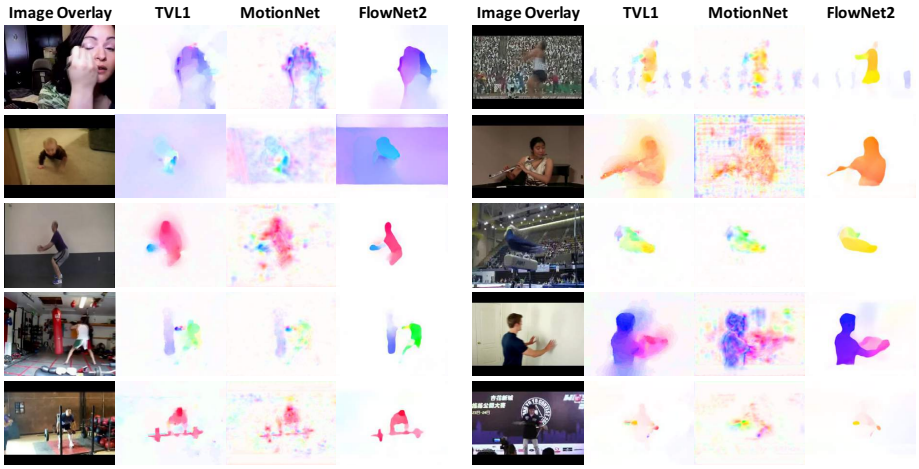

Fig. 2: Visual comparisons of estimated flow field from TV-L1, MotionNet and FlowNet2. Left: ApplyEyeMakeup, BabyCrawling, BodyWeightSquats, BoxingPunchingBag and CleanAndJerk. Right: Hammering, PlayingFlute, PommelHorse, WallPushups and YoYo. This figure is best viewed in color.

6 Comparison to State-of-the-Art Real-Time Approaches

In this section, we compare our proposed method to recent real-time state-of-the-art approaches as shown in Table 4[4]. Among all real-time methods, our hidden two-stream networks achieves the highest accuracy on the four benchmarks. We also show the flexibility of our MotionNet by concatenating it to temporal streams with different backbone CNN architectures, e.g., VGG16 [24], TSN [25] and I3D [1]. With deeper networks, we can achieve higher recognition accuracy and still be real-time. We are 6.1% better on UCF101, 14.2% better on HMDB51, 8.5% better on THUMOS14 and 7.8% better on ActivityNet than the previous state-of-the-art. This indicates that our stacked end-to-end learning framework can implicitly learn better motion representations than motion vectors [10, 33] and RGB differences [25] with respect to the task of action recognition.

7 Conclusion

We have proposed a new framework called hidden two-stream networks to recognize human actions in video. It addresses the problem of capturing the temporal relationships among video frames which the current CNN architectures have difficulty with. Different from the current common practice of using traditional

[4] In general, the requirement for real-time processing is 25 fps. We also compare to other non real-time approaches in the supplementary materials.

Table 4: Comparison to state-of-the-art real-time approaches on four benchmarks with respect to mean classification accuracy. * indicates results from our implementation.

Method	UCF101(%)	HMDB51(%)	THUMOS14(%)	ActivityNet(%)
MV + FV [10]	78.5	46.7	–	–
EMV [33]	80.2	–	41.6	–
C3D (1 Net) [21]	82.3	49.7*	54.6	74.1
ActionFlowNet [15]	83.9	56.4	51.3*	68.8*
RGB + EMV [33]	86.4	–	61.5	–
3DNet [2]	90.2	–	–	–
RGB Diff (TSN) [25]	91.0	64.5*	71.9*	83.0*
Ours (VGG16)	90.3	60.5	66.7	77.8
Ours (TSN)	93.2	66.8	74.5	87.9
Ours (I3D)	**97.1**	**78.7**	**80.6**	**91.2**

local optical flow estimation methods to pre-compute the motion information for CNNs, we use an unsupervised pre-training approach. Our MotionNet is computationally efficient and end-to-end trainable. It is flexible and can be directly applied in other frameworks for various video understanding applications. Experimental results on four challenging benchmarks demonstrate the effectiveness of our approach.

In the future, we would like to improve our hidden two-stream networks in the following directions. First, we would like to improve our optical flow prediction based on the observation that the smoothness loss has significant impact on the quality of the motion estimations for action recognition. Second, we would like to incorporate other best practices that improve the overall performance of the networks. For example, joint training of the two streams instead of a simple late fusion. Third, it would be interesting to see how addressing the false label assignment problem can help improve our overall performance. Finally, removing global camera motion and partial occlusion within the CNN framework would be helpful for both optical flow estimation and action recognition.

Acknowledgement We gratefully acknowledge the support of NVIDIA Corporation through the donation of the Titan Xp GPUs used in this work.

References

1. Carreira, J., Zisserman, A.: Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
2. Diba, A., Pazandeh, A.M., Gool, L.V.: Efficient Two-Stream Motion and Appearance 3D CNNs for Video Classification. In: European Conference on Computer Vision (ECCV) Workshops (2016)

3. Fernando, B., Gavves, E., M., J.O., Ghodrati, A., Tuytelaars, T.: Modeling Video Evolution for Action Recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
4. Fischer, P., Dosovitskiy, A., Ilg, E., Husser, P., Hazrba, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T.: FlowNet: Learning Optical Flow with Convolutional Networks. In: International Conference on Computer Vision (ICCV) (2015)
5. Gorban, A., Idrees, H., Jiang, Y.G., Roshan Zamir, A., Laptev, I., Shah, M., Sukthankar, R.: THUMOS Challenge: Action Recognition with a Large Number of Classes. `http://www.thumos.info/` (2015)
6. Gu, B., Xin, M., Huo, Z., Huang, H.: Asynchronous Doubly Stochastic Sparse Kernel Learning. In: Association for the Advancement of Artificial Intelligence (AAAI) (2018)
7. Heilbron, F.C., Escorcia, V., Ghanem, B., Niebles, J.C.: ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
8. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
9. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial Transformer Network. In: Neural Information Processing Systems (NIPS) (2015)
10. Kantorov, V., Laptev, I.: Efficient Feature Extraction, Encoding and Classification for Action Recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
11. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale Video Classification with Convolutional Neural Networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
12. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: HMDB: A Large Video Database for Human Motion Recognition. In: International Conference on Computer Vision (ICCV) (2011)
13. Lan, Z., Zhu, Y., Hauptmann, A.G., Newsam, S.: Deep Local Video Feature for Action Recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
14. Miao, X., Zhen, X., Liu, X., Deng, C., Athitsos, V., Huang, H.: Direct Shape Regression Networks for End-to-End Face Alignment. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
15. Ng, J.Y.H., Choi, J., Neumann, J., Davis, L.S.: ActionFlowNet: Learning Motion Representation for Action Recognition. In: IEEE Winter Conference on Applications of Computer Vision (WACV) (2018)
16. Ng, J.Y.H., Hausknecht, M., Vijay., S., Vinyals, O., Monga, R., Toderici, G.: Beyond Short Snippets: Deep Networks for Video Classification. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
17. Qiu, Z., Yao, T., Mei, T.: Learning Spatio-Temporal Representation with Pseudo-3D Residual Networks . In: International Conference on Computer Vision (ICCV) (2017)
18. Sedaghat, N.: Next-Flow: Hybrid Multi-Tasking with Next-Frame Prediction to Boost Optical-Flow Estimation in the Wild. arXiv:1612.03777 (2016)
19. Simonyan, K., Zisserman, A.: Two-Stream Convolutional Networks for Action Recognition in Videos. In: Neural Information Processing Systems (NIPS) (2014)
20. Soomro, K., Zamir, A.R., Shah, M.: UCF101: A Dataset of 101 Human Action Classes From Videos in The Wild. In: CRCV-TR-12-01 (2012)

21. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning Spatiotemporal Features with 3D Convolutional Networks. In: International Conference on Computer Vision (ICCV) (2015)
22. Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A Closer Look at Spatiotemporal Convolutions for Action Recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
23. Wang, H., Schmid, C.: Action Recognition with Improved Trajectories. In: International Conference on Computer Vision (ICCV) (2013)
24. Wang, L., Xiong, Y., Wang, Z., Qiao, Y.: Towards Good Practices for Very Deep Two-Stream ConvNets. arXiv:1507.02159 (2015)
25. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Gool, L.V.: Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. In: European Conference on Computer Vision (ECCV) (2016)
26. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image Quality Assessment: From Error Visibility to Structural Similarity. IEEE Transaction on Image Processing (2004)
27. Wu, C.Y., Zaheer, M., Hu, H., Manmatha, R., Smola, A.J., Krhenbhl, P.: Compressed Video Action Recognition. arXiv:1712.00636 (2017)
28. Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K.: Rethinking Spatiotemporal Feature Learning For Video Understanding. arXiv:1712.04851 (2017)
29. Xue, J., Zhang, H., Dana, K.: Deep Texture Manifold for Ground Terrain Recognition. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
30. Xue, T., Chen, B., Wu, J., Wei, D., Freeman, W.T.: Video Enhancement with Task-Oriented Flow. arXiv:1711.09078 (2017)
31. Yu, J.J., Harley, A.W., Derpanis, K.G.: Back to Basics: Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness. In: European Conference on Computer Vision (ECCV) Workshops (2016)
32. Zach, C., Pock, T., Bischof, H.: A Duality Based Approach for Realtime TV-L1 Optical Flow. In: 29th DAGM conference on Pattern Recognition (2014)
33. Zhang, B., Wang, L., Wang, Z., Qiao, Y., Wang, H.: Real-time Action Recognition with Enhanced Motion Vector CNNs. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
34. Zhu, Y., Lan, Z., Newsam, S., Hauptmann, A.G.: Guided Optical Flow Learning. arXiv preprint arXiv:1702.02295 (2017)
35. Zhu, Y., Long, Y., Guan, Y., Newsam, S., Shao, L.: Towards Universal Representation for Unseen Action Recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
36. Zhu, Y., Newsam, S.: DenseNet for Dense Flow. In: IEEE International Conference on Image Processing (ICIP) (2017)
37. Zhu, Y., Newsam, S.: Learning Optical Flow via Dilated Networks and Occlusion Reasoning. In: IEEE International Conference on Image Processing (ICIP) (2018)