End-to-End Learning of Motion Representation for Video Understanding

Lijie Fan*[2], Wenbing Huang*[1], Chuang Gan[3], Stefano Ermon[4], Boqing Gong[1], Junzhou Huang[1]
[1] Tencent AI Lab, [2] Tsinghua University, Beijing, China
[3] MIT-Watson Lab, [4] Department of Computer Science, Stanford University
flj14@mails.tsinghua.edu.cn, hwenbing@126.com, ganchuang1990@gmail.com
ermon@cs.stanford.edu, boqinggo@outlook.com, jzhuang@uta.edu

Abstract

Despite the recent success of end-to-end learned representations, hand-crafted optical flow features are still widely used in video analysis tasks. To fill this gap, we propose TVNet, a novel end-to-end trainable neural network, to learn optical-flow-like features from data. TVNet subsumes a specific optical flow solver, the TV-L1 method, and is initialized by unfolding its optimization iterations as neural layers. TVNet can therefore be used directly without any extra learning. Moreover, it can be naturally concatenated with other task-specific networks to formulate an end-to-end architecture, thus making our method more efficient than current multi-stage approaches by avoiding the need to pre-compute and store features on disk. Finally, the parameters of the TVNet can be further fine-tuned by end-to-end training. This enables TVNet to learn richer and task-specific patterns beyond exact optical flow. Extensive experiments on two action recognition benchmarks verify the effectiveness of the proposed approach. Our TVNet achieves better accuracies than all compared methods, while being competitive with the fastest counterpart in terms of features extraction time.

1. Introduction

Deep learning and especially Convolutional Neural Networks (CNNs) have revolutionized image-based tasks, *e.g.*, image classification [16] and objective detection [32]. However, the progress on video analysis is still far from satisfactory, reflecting the difficulty associated with learning representations for spatiotemporal data. We believe that the major obstacle is that the distinctive motion cues in videos demand some new network designs, which are yet to be found and tested.

While there have been some attempts [36] to learn features by convolution operations over both spatial and temporal dimensions, optical flow is still widely and effectively used for video analysis [28, 9, 10, 40, 15, 29]. The optical flow, as the name implies, captures the displacements of pixels between two consecutive frames [42]. Thus, applying the optical flow to the video understanding tasks enables one to model the motion cues explicitly, conveniently, but inefficiently. It is often computationally expensive to estimate the optical flows. A currently successful example of applying optical flow to video understanding is the two-stream model [33], where a CNN is trained on the optical flow data to learn action patterns. Various extensions of the two-stream model have been proposed and achieved state-of-the-art results on serval tasks including action recognition [28, 9, 10, 40] and action detection [15, 29].

Figure 1. Visualization results of optical-flow-like motion features by TV1 [42], TVNet (without training) and TVNet (with training).

Despite the remarkable performance, current optical-flow-based approaches have notable drawbacks:

- **Training is a two-stage pipeline.** In the first stage, the optical flow for every two consecutive frames is extracted via the optimization-based method (*e.g.* TV-L1 [42]). In the second stage, a CNN is trained on the extracted flow data. These two stages are separated and the information (*e.g.* gradients) from the second stage cannot be used to adjust the process of the first stage.

- **Optical flow extraction is expensive in space and time.** The extracted optical flow has to be written to

*indicates equal contributions. This work was conducted when Lijie Fan was served as a research intern in Tencent AI Lab.

the disk for both the training and testing. For the UCF-101 dataset [34] which contains about 10 thousands videos, extracting optical flows for all data via the TV-L1 method takes one GPU-day, and storing them costs more than one TeraByte of storage for the original fields as floats (often a linear JPEG normalization is required to save storage cost [33]).

To tackle the above mentioned problems, we propose a novel neural network design for learning optical-flow like features in an end-to-end manner. This network, named TVNet, is obtained by imitating and unfolding the iterative optimization process of TV-L1 [42]. In particular, we formulate the iterations in the TV-L1 method as customized layers of a neural network. As a result, our TVNet is well-grounded and can be directly used without additional training by any groundtruth optical flows.

Furthermore, our TVNet is end-to-end trainable, and can therefore be naturally connected with a tasks-specific network (*e.g*. action classification network) to form a "deeper" end-to-end trainable architecture. As a result, it is not necessary to pre-compute or store the optical-flow features anymore.

Finally, by performing the end-to-end learning, it is possible to fine-tune the weights of the TVNet that is initialized as a standard optical flow feature extractor. This allows us to discover richer and task-specific features (compared to the original optical flow) and thus to deliver better performance.

To verify the effectiveness of the proposed architecture, we perform experimental comparisons between the proposed TVNet and several competing methods on two action recognition benchmarks (HMDB51 [24] and UCF101 [34]).

To sum up, this paper makes the following contributions:

- We develop a novel neural network to learn motions from videos by unfolding the iterations of the TV-L1 method to customized neural layers. The network, dubbed TVNet, is well-initialized and end-to-end trainable.
- Despite being initialized as a specific TV-L1 architecture, the proposed TVNet can be further fine-tuned to learn richer and more task-oriented features than the standard optical flow.
- Our TVNet achieves better accuracies than other action representation condunterparts (*e.g*., TV-L1 [42], FlowNet2.0 [18]) and 3D Convnets [36] on the two action recognition benchmarks, *i.e*.,72.6% on HMDB51 and 95.4% on UCF101.

2. Related Work

Video understanding, such as action recognition and action similarity detection, has attracted a lot of research attention in the past decades. Different from static image understanding, video understanding requires more reliable motion features to reflect the dynamic changes occurring in videos. Laptev *et al*. [25] proposed a spatio-temporal interest points (STIPs) method by extending Harris corner detectors to 3-dimensional space to capture motion. Similarly, the 3D extensions of SIFT and HOG have also been investigated [7] and [22], respectively. Wang *et al*. [37] proposed improved Dense Trajectories (iDT), where the descriptors were obtained by tracking densely sampled points and describing the volume around the tracklets by histograms of optical flow (HOF) and motion boundary histograms (MBH). Despite its stat-of-the-art performances, IDT is computationally expensive and becomes intractable on large-scale video dataset.

Motivated by the promising results of deep networks on image understanding tasks, there have also been a number of attempts to develop deep architectures to learn motion features for video understanding [20, 21, 28, 13, 17, 43, 14, 12]. The leading approaches fall into two broad categories. The first one is to learn appearance and motion jointly by extending 2D convolutional layers to 3D counterparts [36, 20], including recently proposed I3D [6] and P3D [30]. However, modeling motion information through 3D convolutional filters is computationally expensive, and large-scale training videos are needed for desired performance [6]. The other category of work is based on two-stream networks [33, 28, 40, 9, 10, 26]. This line of approaches trains two networks, one using the appearance (*i.e*., RGB) data and the other one using hand-crafted motion features such as optical flow to represent motion patterns. In contrast, in our method, the motion descriptor is learned with a trainable neural network rather than hand-crafted. As a consequence, our optical-flow-like motion features can be jointly learned and fine-tuned using a task-specific network. Additionally, we do not need to store and read the optical flow from disk, leading to significant computational gains.

A recent research topic is to estimate optical flow by CNNs [8, 35, 31, 18, 26, 4]. These approaches cast the optical flow estimation as an optimization problem with respect to the CNN parameters. A natural idea is to combine the flow CNN with the task-specific network to formulate an end-to-end model (see for example in [45]). Nevertheless, an obvious issue of applying the flow nets is that they require thousands of hundreds of groundtrue flow images to train the parameters of the flow network to produce meaningful optical flows (see [8]). For real applications, it is costly to obtain the labeled flow data. In contrast, our network is well initialized as a particular TV-L1 method and is able to achieve desired performance even in its initial form (without fine-tuning).

Recently, Ng *et al*. [27] proposed to train a single stream convolutional neural network to jointly estimate optical flow and recognize actions, which is most relevant to our work. To capture the motion feature, they formulated FlowNet [11] to learn the optical flow from synthetic ground truth data.

Though the results are promising, the approach still lags behind the state of the arts in terms of accuracy compared to traditional approaches. This is due to the well known gap between synthetic and real videos. Contrastly, our network is formulated by unfolding the TV-L1 method that has been applied successfully to action recognition and we do not rely on the groundtruth of optical flow for training. Thus, our network combines the strengths of both TV-L1 and deep learning.

3. Notations and background

3.1. Notations

A video sequence can be written as a function of three arguments, $\boldsymbol{I}_t(x, y)$, where x, y index the spatial dimensions t is for the time dimension. Denote by Ω all the coordinates of the pixels in a frame. The function value $\boldsymbol{I}_t(x, y)$ corresponds to the pixel brightness at position $\boldsymbol{x} = (x, y) \in \Omega$ in the t-th video frame. A point $\boldsymbol{x}$ may move from time to time across the video frames, and the optical flow is to track such displacement between adjacent frames. We denote by $\boldsymbol{u}^t(\boldsymbol{x}) = (u_1^t(\boldsymbol{x}), u_2^t(\boldsymbol{x}))$ the displacement of the point $\boldsymbol{x}$ from time t to the next frame $t+1$. We omit the superscript t and/or argument $\boldsymbol{x}$ from $\boldsymbol{u}^t(\boldsymbol{x})$ when no ambiguity is caused.

3.2. The TV-L1 method

Among the existing approaches to estimating optical flows, the TV-L1 method [42] is especially appealing for its good balance between efficiency and accuracy. We review it in detail in this subsection to make the paper self-contained. The design of our TV-Net (cf. Section 4) is directly motivated by the optimization procedure of TV-L1.

The main formulation of TV-L1 is as follows,

$$\min_{\boldsymbol{u}(\boldsymbol{x}), \boldsymbol{x}\in\Omega} \sum_{\boldsymbol{x}\in\Omega} (|\nabla \boldsymbol{u}_1(\boldsymbol{x})| + |\nabla \boldsymbol{u}_2(\boldsymbol{x})|) + \lambda |\rho(\boldsymbol{u}(\boldsymbol{x}))|, \quad (1)$$

where the first term $|\nabla \boldsymbol{u}_1| + |\nabla \boldsymbol{u}_2|$ accounts for the *smoothness condition*, while the second term $\rho(\boldsymbol{u})$ corresponds to the famous *brightness constancy assumption* [42]. In particular, the brightness of a point $\boldsymbol{x}$ is assumed to remain the same after it shifts to a slightly different location in the next frame, i.e., $\boldsymbol{I}_0(\boldsymbol{x} + \boldsymbol{u}) \approx \boldsymbol{I}_1(\boldsymbol{x})$. Accordingly, $\rho(\boldsymbol{u}) = \boldsymbol{I}_1(\boldsymbol{x} + \boldsymbol{u}) - \boldsymbol{I}_0(\boldsymbol{x})$ is defined in order to penalize the brightness difference in the second term. Since the function $\boldsymbol{I}_1(\boldsymbol{x} + \boldsymbol{u})$ is highly non-linear with respect to $\boldsymbol{u}$, Zach et al. [42] approximate the brightness difference $\rho(\boldsymbol{u})$ by the Taylor expansion at an initial displacement $\boldsymbol{u}^0$, leading to $\rho(\boldsymbol{u}) \approx \nabla \boldsymbol{I}_1(\boldsymbol{x} + \boldsymbol{u}^0)(\boldsymbol{u} - \boldsymbol{u}^0) + \boldsymbol{I}_1(\boldsymbol{x} + \boldsymbol{u}^0) - \boldsymbol{I}_0(\boldsymbol{x})$.

The above gives a first-order approximation to the original problem and linearizes it to an easier form. Furthermore, the authors introduce an auxiliary variable $\boldsymbol{v}$ to enable a convex relaxation of the original problem

$$\min_{\{\boldsymbol{u},\boldsymbol{v}\}} \sum_{\boldsymbol{x}\in\Omega} (|\nabla \boldsymbol{u}_1| + |\nabla \boldsymbol{u}_2|) + \frac{1}{2\theta}|\boldsymbol{u} - \boldsymbol{v}|^2 + \lambda |\rho(\boldsymbol{v})|, \quad (2)$$

Algorithm 1 The TV-L1 method for optical flow extraction.

Hyper-parameters: $\lambda, \theta, \tau, \epsilon, N_{warps}, N_{iters}$
Input: $\boldsymbol{I}_0, \boldsymbol{I}_1, \boldsymbol{u}^0$
$\boldsymbol{p}_1 = [\boldsymbol{p}_{11}, \boldsymbol{p}_{12}] = [\boldsymbol{0}, \boldsymbol{0}]$;
$\boldsymbol{p}_2 = [\boldsymbol{p}_{21}, \boldsymbol{p}_{22}] = [\boldsymbol{0}, \boldsymbol{0}]$;
for $w = 1$ **to** N_{warps} **do**
  Warp $\boldsymbol{I}_1(\boldsymbol{x} + \boldsymbol{u}^0), \nabla \boldsymbol{I}_1(\boldsymbol{x} + \boldsymbol{u}^0)$ by interpolation;
  $\rho(\boldsymbol{u}) = \nabla \boldsymbol{I}_1(\boldsymbol{x} + \boldsymbol{u}^0)(\boldsymbol{u} - \boldsymbol{u}^0) + \boldsymbol{I}_1(\boldsymbol{x} + \boldsymbol{u}^0) - \boldsymbol{I}_0(\boldsymbol{x})$,
  $n = 0$;
  while $n < N_{iters}$ and $stopping\_criterion > \epsilon$ **do**

$$\boldsymbol{v} = \begin{cases} \lambda\theta \nabla \boldsymbol{I}_1 & \rho(\boldsymbol{u}) < -\lambda\theta |\nabla \boldsymbol{I}_1|^2, \\ -\lambda\theta \nabla \boldsymbol{I}_1 & \rho(\boldsymbol{u}) > \lambda\theta |\nabla \boldsymbol{I}_1|^2, \\ -\rho(\boldsymbol{u}) \frac{\nabla \boldsymbol{I}_1}{|\nabla \boldsymbol{I}_1|^2} & \text{otherwise}, \end{cases}$$

  where $\nabla \boldsymbol{I}_1$ represents $\nabla \boldsymbol{I}_1(\boldsymbol{x} + \boldsymbol{u}^0)$ for short;
  $\boldsymbol{u}_d = \boldsymbol{v} + \theta \text{div}(\boldsymbol{p}_d), d = 1, 2$;
  $\boldsymbol{p}_d = \frac{\boldsymbol{p}_d + \tau/\theta \nabla \boldsymbol{u}_d}{1 + \tau/\theta |\nabla \boldsymbol{u}_d|}, d = 1, 2$;
  $n = n + 1$;
  end while
end for

in which a very small θ can force $\boldsymbol{u}$ and $\boldsymbol{v}$ to be equal at the minimum. This objective is minimized by alternatively updating $\boldsymbol{u}$ and $\boldsymbol{v}$. The details of the optimization process are presented in Algorithm 1, where the variables $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are the dual flow vector fields.

Understanding Algorithm 1. The core computation challenge in the algorithm is on the pixel-wise computations of the *gradients* (*i.e.*, $\nabla \boldsymbol{I}_1$ and $\nabla \boldsymbol{u}_d$), *divergence* (*i.e.*, $div(\boldsymbol{p})$), and *warping* (*i.e.*, $\boldsymbol{I}_1$ and $\nabla \boldsymbol{I}_1$). The details of the numerical estimations are provided as below.

- **Gradient-1.** The gradient of the image $\boldsymbol{I}_1$ is computed by central difference:

$$\frac{\partial \boldsymbol{I}_1(i,j)}{\partial x} = \begin{cases} \frac{\boldsymbol{I}_1(i+1,j) - \boldsymbol{I}_1(i-1,j)}{2} & 1 < i < W, \\ 0 & \text{otherwise}. \end{cases} \quad (3)$$

  We can similarly compute $\frac{\partial}{\partial y}\boldsymbol{I}_1(i,j)$ along the j index.

- **Gradient-2.** The gradient of each component of the flow $\boldsymbol{u}$ is computed via the forward difference:

$$\frac{\partial \boldsymbol{u}_d(i,j)}{\partial x} = \begin{cases} \boldsymbol{u}_d(i+1,j) - \boldsymbol{u}_d(i,j) & 1 \le i < W, \\ 0 & \text{otherwise}, \end{cases} \quad (4)$$

  where $d \in \{1, 2\}$. Also, $\frac{\partial}{\partial y}\boldsymbol{u}_d(i,j)$ can be similarly computed by taking the difference on the j index.

- **Divergence.** The divergence of the dual variables $\boldsymbol{p}$ is computed via the backward difference:

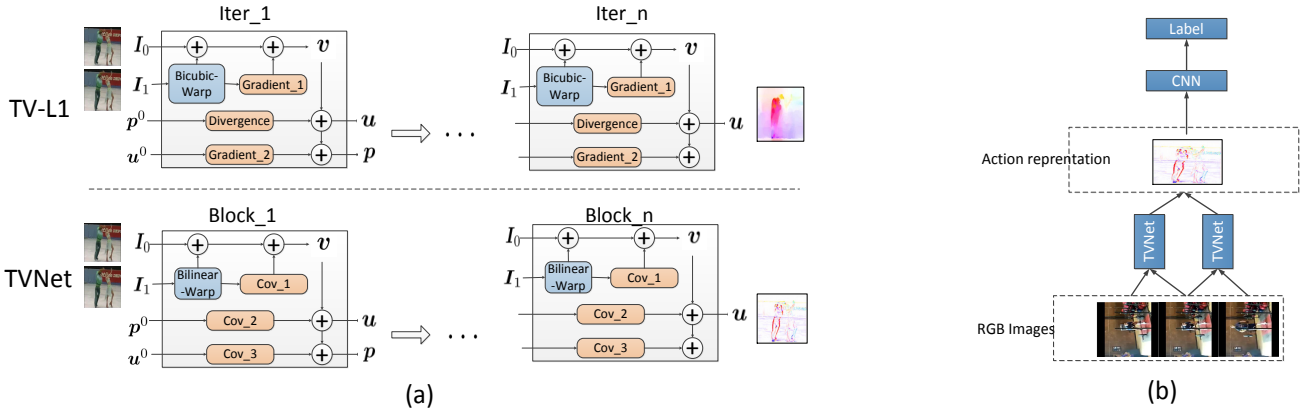

Figure 2. (a) Illustration of the process for unfolding TV-L1 to TVNet. For TV-L1, we illustrate each iteration of Algorithm 1. We reformulate the bicubic warping, gradient and divergence computations in TV-L1 to bilinear warping and convolution operations in TVNet. (b) The end-to-end model for action recognition.

$$
\begin{aligned}
&\operatorname{div}(\boldsymbol{p}_d)(i,j) \\
&= \begin{cases} \boldsymbol{p}_{d1}(i,j) - \boldsymbol{p}_{d1}(i-1,j) & 1 < i < W, \\ \boldsymbol{p}_{d1}(i,j) & i = 1, \\ -\boldsymbol{p}_{d1}(i-1,j) & i = W. \end{cases} \\
&+ \begin{cases} \boldsymbol{p}_{d2}(i,j) - \boldsymbol{p}_{d2}(i,j-1) & 1 < j < H, \\ \boldsymbol{p}_{d2}(i,j) & j = 1, \\ -\boldsymbol{p}_{d2}(i,j-1) & j = H. \end{cases} \qquad (5)
\end{aligned}
$$

Another pixel-wise estimation is the brightness $\boldsymbol{I}_1(\boldsymbol{x}+\boldsymbol{u}^0)$. It is often obtained by warping the frame $\boldsymbol{I}_1$ along the initial flow field $\boldsymbol{u}^0$ using the bicubic interpolation.

Multi-scale TV-L1. Since the Taylor expansion is applied to linearize the brightness difference, the initial flow field $\boldsymbol{u}^0$ should be close to the real field $\boldsymbol{u}$ to ensure a small approximation error. To achieve this, the approximation field $\boldsymbol{u}^0$ is derived by a multi-scale scheme in a coarse-to-fine manner. To be specific, at the coarsest level, $\boldsymbol{u}^0$ is initialized as the zero vectors and the corresponding output of Algorithm 1 is applied as the initialization of the next level[1].

4. TVNets

This section presents the main contribution of this paper, *i.e.*, the formulation of TVNet. The central idea is to imitate the iterative process in TV-L1 and meanwhile unfold the iterations into a layer-to-layer transformations, in the same spirit as the neural networks.

4.1. Network design

We now revisit Algorithm 1 and convert its key components to a neural network. First, the iterations in Algorithm 1 can be unfolded as a fixed-size feed-forward network if we fix the number of the iterations within the while-loop to be N_{iters} (see Figure 2). Second, each iteration (*i.e.* layer) is continuous and is almost everywhere smooth with respect to the input variables. Such property ensures that the gradients can be back-propagated through each layer, giving rise to an end-to-end trainable system.

Converting Algorithm 1 into a neural network involves efficiency and numerical stability considerations. To this end, we modify Algorithm 1 by replacing the computations of the gradients and divergence Eq. (3)-(5) with specific convolutions, performing warping with bilinear interpolation, and stabilizing the division calculations with a small threshold. We provide the details below.

Convolutional computation. The most tedious part in Algorithm 1 is the pixel-wise computation of Eq. (3)-(5). We propose to perform all these calculations with specific convolutional layers. We define the following kernels,

$$\boldsymbol{w}_c = [0.5, 0, -0.5], \boldsymbol{w}_f = \boldsymbol{w}_b = [-1, 1]. \qquad (6)$$

Thus, for the pixels in the valid area ($1 < i < W$), Eq. (3)-(4) can be equivalently written as

$$\frac{\partial}{\partial x}\boldsymbol{I}_1 = \boldsymbol{I}_1 * \boldsymbol{w}_c, \qquad (7)$$

$$\frac{\partial}{\partial x}\boldsymbol{u}_d = \boldsymbol{u}_d * \boldsymbol{w}_f, \qquad (8)$$

where $*$ defines the convolution operation. Eq. (6) only describes the kernels along the x axis. We transpose them to obtain the kernels along the y axis.

The divergence in Eq.(5) is computed by a backward difference, but the convolution is computed in a forward direction. To rewrite Eq.(5) in convolution form, we need to first shift the pixels of $\boldsymbol{p}_{d1}$ right (and shift $\boldsymbol{p}_{d2}$ down) by one pixel and pad the first column of $\boldsymbol{p}_{d1}$ (and the first row of $\boldsymbol{p}_{d2}$) with zeros, leading to $\hat{\boldsymbol{p}_{d1}}$ (and $\hat{\boldsymbol{p}_{d2}}$). Then, Eq.(5) can be transformed to

$$\operatorname{div}(\boldsymbol{p}_d) = \hat{\boldsymbol{p}_{d1}} * \boldsymbol{w}_b + \hat{\boldsymbol{p}_{d2}} * \boldsymbol{w}_b^{\mathrm{T}}, \qquad (9)$$

where $\boldsymbol{w}_b^{\mathrm{T}}$ denotes the transposition of $\boldsymbol{w}_b$. We then refine the boundary points for the outputs of Eq. (7)-(9) to meet the boundary condition in Eq. (3)-(5).

[1]Figure 1 in the supplementary material demonstrates the framework with three-scale optimization.

Bilinear-interpolation-based warping. The original TV-L1 method uses bicubic interpolation for the warping process. Here, for efficiency reasons, we adopt bilinear interpolation instead. Note that the bilinear interpolation has been applied successfully in previous works such as the spatial transformer network [19] and the optical flow extraction method [11]. We denote by $\boldsymbol{I}_1^w = \boldsymbol{I}_1(\boldsymbol{x} + \boldsymbol{u}^0)$ the warping. Then, we compute

$$\boldsymbol{I}_1^w(i,j) = \sum_n^H \sum_m^W \boldsymbol{I}_1(m,n) \max(0, 1 - |i + u_1 - m|) \max(0, 1 - |j + u_2 - n|), \quad (10)$$

where u_1 and u_2 are respectively the horizontal and vertical flow values of $\boldsymbol{u}^0$ at position (i, j). We follow the details in [19] and derive the partial gradients for Eq. (10) with respect to $\boldsymbol{u}^0$ as the bilinear interpolation is continuous and piecewise smooth.

Numerical stabilization. We need to take care of the division in Algorithm 1, *i.e.*, $\boldsymbol{v} = -\rho(\boldsymbol{u})\frac{\nabla \boldsymbol{I}_1}{|\nabla \boldsymbol{I}_1|^2}$. The operation is ill-defined when the denominator is equal to zero. To avoid this issue, the original TV-L1 method checks whether the value of $|\nabla \boldsymbol{I}_1|^2$ is bigger than a small constant; if not, the algorithm will set the denominator to be this small constant. Here, we utilize a soft non-zero transformation by rewriting the update of $\boldsymbol{v}$ as $\boldsymbol{v} = -\rho(\boldsymbol{u})\frac{\nabla \boldsymbol{I}_1}{|\nabla \boldsymbol{I}_1|^2 + \varepsilon}$, where a small value $\varepsilon > 0$ is added to the denominator. This transformation is more efficient as we do not need to explicitly check the value of $|\nabla \boldsymbol{I}_1|^2$ at each step.

Another division computation in Algorithm 1 is $\boldsymbol{p}_d = \frac{\boldsymbol{p}_d + \tau/\theta \nabla \boldsymbol{u}_d}{1 + \tau/\theta |\nabla \boldsymbol{u}_d|}$. At first glance, this division is safe since the denominator is guaranteed to be larger than 1. However, as we will see later, its gradients contain division computations where the denominators can be zero. Thus, we apply the soft transformation by adding a small value $\varepsilon > 0$ to the denominator, namely,

$$\boldsymbol{p}_d = \frac{\boldsymbol{p}_d + \tau/\theta \nabla \boldsymbol{u}_d}{1 + \tau/\theta \sqrt{\nabla \boldsymbol{u}_{d1}^2 + \nabla \boldsymbol{u}_{d2}^2 + \varepsilon}}. \quad (11)$$

The gradient of $\boldsymbol{p}_d$ with respect to $\nabla \boldsymbol{u}_{d1}$ is in this form

$$\frac{\partial}{\partial \nabla \boldsymbol{u}_{d1}} \boldsymbol{p}_d = \boldsymbol{a} - \frac{\boldsymbol{b}}{\sqrt{\nabla \boldsymbol{u}_{d1}^2 + \nabla \boldsymbol{u}_{d2}^2 + \varepsilon}}, \quad (12)$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are well-defined variables (the details are provided in the supplementary material). In practice, both $\nabla \boldsymbol{u}_{d1}$ and $\nabla \boldsymbol{u}_{d2}$ are often equal to zero within the still area of the image (*e.g.*, the background). As such, the computation of the gradients would encounter a division by zero if the positive term ε was not added in Eq. (12).

Multi-scale version. The multi-scale TVNet is formulated by directly unfolding the multi-scale version of TV-L1. A higher scale takes as input the up-sampled output of its immediate lower scale. There are multiple warps at each scale and each warp consists of multiple iterations. Hence, the total number of iterations of the multi-scale TVNets are $N_{scales} \times N_{warps} \times N_{iters}$.

4.2. Going beyond TV-L1

In the previous section, we have transformed the TV-L1 algorithm to a feed-forward network. However, such network is parameter-free and not learnable. To formulate a more expressive network, we relax certain variables in TV-L1 to be trainable parameters. Relaxing the variables render TVNet not equivalent to TV-L1 any more. However, it allows the network to learn more complex, task-specific feature extractors by end-to-end training.

The first variable we relax is the initialization optical field $\boldsymbol{u}^0$. In TV-L1, $\boldsymbol{u}^0$ is set to be zero. However, from the optimization prospective, zero initialization is not necessarily the best choice; making $\boldsymbol{u}^0$ trainable will enable us to automatically determine a better initialization for the optimization. We also propose to relax the convolutional filters in Eq. (7)-(9). The original convolutions are used to derive the (numerical) gradients and divergences. Allowing the convolutional filters to be trainable parameters will enable them to discover more complex patterns in a data-driven way. We will demonstrate the benefit of the trainable version compared to the original architecture in our experiments.

4.3. Multi-task Loss

As discussed before, our TVNet can be concatenated to any task-specific networks (*e.g.*, the BN-Inception net for action classification [40]) to perform end-to-end action recogntion without the need of explicitly extracting the optical flow data, as illustrated in Figure 2 (c). Because of the end-to-end structure, the parameters of TVNet can be fine-tuned by back-propagating gradients of the task-specific loss. Additionally, since the original TV-L1 method is developed to minimize the energy function in Eq. (1), we can also use this function as an additional loss function to force it to produce meaningful optical-flow-like features. To this end, we formulate a multi-task loss as

$$L = L_c + \lambda L_f. \quad (13)$$

Here L_c is the action classification loss (*e.g.* the cross entropy), L_f is defined in Eq. (1) where the exact computation other than the Tailor approximation is applied to compute $\boldsymbol{\rho}(\boldsymbol{u}(\boldsymbol{x}))$, and λ is a hyper-parameter to trade-off these two losses. We set $\lambda = 0.1$ in all our experiments and find that it works well across all of them. Note that it is tractable to compute the gradients of L_f as it has been translated to convolutions and the bilinear interpolation (see § 4.1).

5. Experiments

This section performs experimental evaluations to verify the effectiveness of the proposed TVNet. We first carry out a complete comparison between TVNets of various structures with the TV-L1 method regarding the optimization efficiency. Then, we compare the performance of TVNets with state-of-

Table 1. The average EPEs on MiddleBurry. "Training $\boldsymbol{u}^0$" means only $\boldsymbol{u}^0$ is trained; "All Training" means both $\boldsymbol{u}^0$ and the convolution filters are trained. After training, TVNet-50 outperforms TV-L1 significantly although TV-L1 has a much larger number of optimization iterations (*i.e.*, 1250).

Methods	No training	Training $\boldsymbol{u}^0$	All Training
TVNet-10	3.47	2.92	1.24
TVNet-30	3.01	2.04	0.40
TVNet-3-1-10	2.00	0.82	0.52
TVNet-1-3-10	2.81	2.17	0.46
TVNet-50	2.93	1.58	**0.35**
TV-L1-10	3.48	TV-L1-3-1-10	1.79
TV-L1-30	3.02	TV-L1-1-3-10	2.74
TV-L1-50	2.86	TV-L1-5-5-50	0.66

the-art methods on the task of action recognition[2].

The three hyper-parameters, N_{scales}, N_{warps} and N_{iters} determine the structure of the TVNet. For convenience, we denote the TVNet with particular values of the hyper-parameters as TVNet-N_{scales}-N_{warps}-N_{iters}. We denote the architecture as TVNet-N_{iters} for short when both N_{scales} and N_{warps} are fixed to be 1. For the TV-L1 method, the hyper-parameters are fixed as $N_{scales} = N_{warps} = 5$ and $N_{iters} = 50$ in all experiments unless otherwise specified. Our methods are implemented by the Tensorflow platform [1]. Unless otherwise specified, all experiments were performed on 8 Tesla P40 GPUs.

5.1. Comparison with TV-L1

Initialized as a particular TV-L1 method, the parameters of TVNet can be further finetuned as discussed in Section 4.2. Therefore, it is interesting to evaluate how much the training process can improve the final performance. For this purpose, we compare the estimation errors between TVNet and TV-L1 on the optical flow dataset, *i.e.*, the MiddleBurry dataset [2].

Dataset. The MiddleBurry dataset [2] is a widely-used benchmark for evaluating different optical flow extraction methods. Here we only perform evaluation on the training set as we are merely concerned about the training efficiency of TVNets. For the training set, only 8 image pairs are provided with the ground-true optical flow.

Implementation details. The estimation errors are measured via the average End-Point Error (EPE) defined by

$$EPE \doteq \frac{1}{N}\sum_{i=1}^{N}\sqrt{(u_{1,i}-u_{1,i}^{gt})^2+(u_{2,i}-u_{2,i}^{gt})^2}, \quad (14)$$

where $(u_{1,i}, u_{2,i})$ and $(u_{1,i}^{gt}, u_{2,i}^{gt})$ are the predicted and ground-true flow fields, respectively. For the training of TVNets, we adopt the EPE (Eq. (14)) as the loss function, and apply the batch gradient decent method with the learning rate and max-iteration being 0.05 and 3000, respectively. Several structures, *i.e.*, TVNet-10, TVNet-30, TVNet-50, TVNet-3-10, TVNet-1-3-10, and their counterparts of TV-L1 are implemented to compare the difference between different network designs.

[2]We also provide additional experimental evaluations on action similarity labeling in the supplementary material.

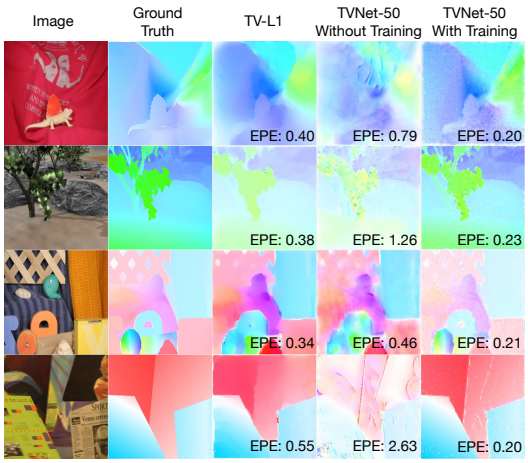

Figure 3. Examples of flow fields from TV-L1 and TVNet-50 estimated on MiddleBurry. With training, TVNet-50 is able to extract finer details than TV-L1 does.

Table 2. The execution speed of different flow extraction methods. Only one gpu is used for the evaluations. As no ground-truth is given on UCF101, we apply the term $\rho(\boldsymbol{u})$ (Eq.(1)) instead of End-Point-Error to measure the optical flow error. TVNet-50 achieves the fastest speed among all the methods. Theoretically, since TVNet-50 has a much smaller number of iterations than TV-L1 (*i.e.* 50 v.s. 1250), the speed of TVNet-50 should be more than 100 times faster than TV-L1. However, due to the different implementations of TV-L1 and TVNet, the real computational reduction of TVNet is not so big. As the TVNet-50 is implemented by Tensorflow, we can easily perform parallel flow extraction with TVNet-50 by enlarging the batch size of input images (*e.g.*, setting batch = 10); as such, the FPS will be further improved to 60.

Methods	FPS	Flow Errors	Trainable	#Parameters
DIS-Fast	9.23	1.29	No	No
Deepflow	0.69	1.33	No	No
Flownet2.0	4.53	1.32	Yes	10^5
TV-L1	6.67	0.86	No	No
TVNet-50	**12**	0.93	Yes	10^2

Table 3. Classification accuracy of various motion descriptors on HMDB51 (split 1) and UCF101 (split 1).The top part shows the results of current best action representation methods; the middle part reports the accuracies of the four baselines; the bottom part presents the performance of our models. TVNet-50 achieves the best results on both datasets.

Methods	HMDB51	UCF101
C3D [36]	-	82.3%
ActionFlowNet [27]	56.4%	83.9%
TV-L1	56.0%	85.1%
DIS-Fast	40.4%	71.2%
Deepflow	50.4%	82.1%
Flownet2.0	52.3%	80.1%
TVNet-50 (no training)	55.6%	83.5%
TVNet-50 (no flow loss)	56.9%	84.5%
TVNet-50	**57.5%**	**85.5%**

Results. We have performed one-to-one comparisons between TVNets and TV-L1 on MiddleBurry in Table 1. Given the same architecture, TVNet without training achieves close performance to TV-L1. This is not surprising since TVNet and TV-L1 are almost the same except the way of interpolation (bilinear vs. bicubic). To further evaluate the effect of training $\boldsymbol{u}^0$, we conduct additional experiments and report the results in Table 1. Clearly, making $\boldsymbol{u}^0$ trainable in TVNets can indeed reduce the End-Point Error. With training both $\boldsymbol{u}^0$ and the convolution filters, all TVNets except TVNet-10 achieve lower errors than TV-L1-5-5-50, even though the number of iterations in TVNets (not more than 50) are much smaller than that of TV-L1-5-5-50 (up to 1250). Figure 3 displays the visualization of the optical flow between TV-L1-5-5-50 and TVNet-50. Another interesting observation is from the comparison between TVNet-30, TVNet-50, TVNet-3-10 and TVNet-1-3-10. It is observed TVNet-30 and TVNet-50 finally outperform TVNet-3-10 and TVNet-1-3-10 after training, implying that the flat structure (*i.e.* $N_{scales} = N_{warps} = 1$) is somehow easier to train. For the remaining experiments below, we will only compare the performance between TVNet-50 and TV-L1-5-5-50, and denote TV-L1-5-5-50 as TV-L1 for similarity.

5.2. Action recognition

Dataset. Our experiments are conducted on two popular action recognition datasets, namely the UCF101 [34] and the HMDB51 [24] datasets. The UCF101 dataset contains 13320 videos of 101 action classes. The HMDB51 dataset consists of 6766 videos from 51 action categories.

Implementation details. As discussed before, our TVNets can be concatenated by a classification network to formulate an end-to-end model to perform action recognition. We apply the BN-Inception network [40] as the classification model in our experiments due to its effectiveness. The BN-Inception network is pretrained by the cross-modality skill introduced in [39] for initialization.

We sample a stack of 6 consecutive images from each video and extract 5 flow frames for every consecutive pair. The resulting stack of optical flows are fed to the BN-Inception network for prediction. To train the end-to-end model, we set the mini-batch size of the sampled stacks to 128 and the momentum to 0.9. The learning rate was initialized to 0.005. The maximum number of learning iterations for the UCF101 and the HMDB51 datasets was chosen as 18000 and 7000, respectively. We decreased the learning rates by a factor of 10 after the 10000th and 16000th iterations for the UCF101 experiment, and after 4000th and 6000th iterations for the HMDB51 case. We only implement TVNet-50 in this experiment. To prevent overfitting, we also carry out the corner cropping and scale jittering [40]; the learning rate for TVNets is further divided by 255.

For the testing, stacks of flow fields are extracted from the center and four corners of a video. We sample 25 stacks from each location (*i.e.*, center and corners), followed by flipping them horizontally to enlarge the testing samples. All the sampled snippets (250 in total) are fed to BN-Inception [40] and their outputs are averaged for prediction.

Table 4. Mean classification accuracy of the state-of-the-arts on HMDB51 and UCF101.

Method	HMDB51	UCF101
ST-ResNet [9]	66.4%	93.4%
ST-ResNet + IDT [9]	70.3%	94.6%
TSN [40]	68.5%	94.0%
KVMF [44]	63.3%	93.1%
TDD [38]	65.9%	91.5%
C3D (3 nets) [36]	-	90.4%
Two-Stream Fusion [10]	65.4%	92.5%
Two-Stream (VGG16) [3]	58.5%	91.4%
Two-Stream+LSTM [28]	-	88.6%
Two-Stream Model [33]	59.4%	88.0%
Ours	71.0%	94.5%
Ours + IDT	**72.6**%	**95.4**%

Baselines. Beside the TV-L1 method, we carry out other three widely-used flow extraction baselines including DIS-Fast [23], DeepFlow [41] and FlowNet2.0 [18]. For FlowNet2.0, we use the pretrained model by the KITTI dataset. For all baselines, we compute the optical flow beforehand and store the flow fields as JPEG images by linear compression. All methods share the same training setting and classification network for fair comparison.

Computational efficiency comparisons. We have added thorough computational comparison between TVNet, TV-L1, DIS-Fast, Deepflow, and Flownet2.0 in Table 2. To do so, we randomly choose one testing video from the UCF101 dataset, and compute the optical flow for every two consecutive frames. The average running time (excluding I/O times) for TVNet-50, TV-L1, DIS-Fast, DeepFlow and FlowNet2.0 are summarized in Table 2. The results verify the advantages of TVNets regarding high number of Frames-per-Second (FPS), low optical flow error, end-to-end trainable property, and small number of model parameters. Flownet2.0 performs more accurately than TV-L1 on the optical flow datasets (e.g. MiddleBurry) as reported by [18]. However, for the action datasets, TV-L1 and our TVNet obtain lower flow error than Flownet2.0 according to Table 2.

Classification accuracy comparisons. Table 3 presents action recognition accuracies of TVNet-50 compared with the four baselines and current best action representation methods. Clearly, TVNet-50 outperforms all compared methods on both datasets. Compared to TV-L1, the improvement of TVNet-50 on UCF101 is not big; however, our TVNet-50 is computationally advantageous over TV-L1 because it only employs one scale and one warp, while TV-L1 adopts five scales and five warps. Even when we freeze its parameters, TVNet-50 still achieves better results than DIS-Fast, DeepFlow and FlowNet2.0; as our TVNet is initialized as a special TV-L1, the initial structure is sufficient to perform

Figure 4. Illustrations of the motion patterns obtained by TV-L1 and TVNet-50 on the UCF101 dataset. From the first to the last column, we display the image-pair (first image only), the motion features by TV-L1, TVNet-50 without training and with training, respectively. Interestingly, with training, TVNet-50 generates more abstractive motion features than TV-L1 and its non-trained version. These features not only automatically remove the movement of the background (see the "punch" example), but also capture the outline of the moving objects.

promisingly. The TVNet is trained with the multi-task loss given by Eq. (13). To verify the effect of the flow loss term, *i.e.* L_f, we train a new model only with the classification loss. Table 3 shows that such setting decreases the accuracy.

Flownet2.0 can also be jointly finetuned for action classification. This is done in ActionFlowNet [27] but the results, as incorporated in Table 3, are worse than ours. This is probably because TVNet has much fewer parameters than Flownet2.0, making the training more efficient and less prone to overfitting. For the UCF101 dataset, the TVNet outperforms C3D [36] by more than 2%. The C3D method applied 3-dimensional convolutions to learn spatiotemporal features. In contrast to this implicit modeling, in our model, the motion pattern is extracted by TVNet explicitly. We also visualize the outputs by TV-L1 and TVNets in Figure 4.

Comparison with other state-of-the-arts. To compare with state-of-the-art methods, we apply several practical tricks to our TVNet-50, as suggested by previous works [33, 40]. First, we perform the two-stream combination trick [33] by additionally training a spatial network on RGB images. We use the BN-Inception network as the spatial network and apply the same experimental setting as those in [40] for the training. At testing, we combine the predictions of the spatial and temporal networks with a fixed weight (*i.e.*, 1:2). Second, to take the long-term temporal awareness into account, we perform the temporal pooling of 3 sampled segments for each video during training as suggested by [40].

Table 4 summarizes the classification accuracy of TVNets compared with the state-of-the-art approaches over all three splits of the UCF101 and the HMDB51 dataset. The improvements achieved by TVNets are quite substantial compared to the original two-stream method [33] (6.5% on UCF101 and 11.6% on HMDB51). Such significant gains are achieved as a result of employing better models (*i.e.*, BN-Inception net) and also considering end-to-end motion mining.

The TSN method [40] is actually a two-stream model with TV-L1 inputs. TSN shares the same classification network and experimental setups as our TVNets. As shown in Table 3, our TVNets outperform TSN on both action datasets (e.g. 71.6% vs. 68.5% on HMDB51), verifying the effectiveness of TVNets for the two-stream models.

Combining CNN models with trajectory-based hand-crafted IDT features [37] can improve the final performances [38, 36, 5, 9]. Hence, we averaged the L2-normalized SVM scores of FV-encoded IDT features (*i.e.*, HOG, HOF and MBH) with the L2-normalized video predictions (before the loss layer) of our methods. Table 4 summarizes the results and indicates that there is still room for improvement. Our 95.4% on the UCF101 and 72.6% on the HMDB51 remarkably outperform all the compared methods.

A recent state-of-the-art result is obtained by I3D [6], achieving 97.9% on UCF101 and 80.2% on HMDB51. However, the I3D method improves the performance by using a large amount of additional training data. It is unfair to compare their results with ours.

6. Conclusion

In this paper, we propose a novel end-to-end motion representation learning framework, named as TVNet. Particularly, we formulate the TV-L1 approach as a neural network, which takes as input stacked frames and outputs optical-flow-like motion features. Experimental results on two video understanding tasks demonstrate its superior performances over the existing motion representation learning approaches. In the future, we will explore more large-scale video understanding tasks to examine the benefits of the end-to-end motion learning method.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 6

[2] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011. 6

[3] N. Ballas, L. Yao, C. Pal, and A. Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015. 7

[4] Y. Bian, C. Gan, X. Liu, F. Li, X. Long, Y. Li, H. Qi, J. Zhou, S. Wen, and Y. Lin. Revisiting the effectiveness of off-the-shelf temporal modeling approaches for large-scale video classification. *arXiv preprint arXiv:1708.03805*, 2017. 2

[5] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3034–3042, 2016. 8

[6] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733. IEEE, 2017. 2, 8

[7] M.-y. Chen and A. Hauptmann. Mosift: Recognizing human actions in surveillance videos. 2009. 2

[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 *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015. 2

[9] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pages 3468–3476, 2016. 1, 2, 7, 8

[10] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, 2016. 1, 2, 7

[11] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *ICCV*, 2015. 2, 5

[12] C. Gan, C. Sun, L. Duan, and B. Gong. Webly-supervised video recognition by mutually voting for relevant web images and web video frames. In *ECCV*, pages 849–866, 2016. 2

[13] C. Gan, N. Wang, Y. Yang, D.-Y. Yeung, and A. G. Hauptmann. Devnet: A deep event network for multimedia event detection and evidence recounting. In *CVPR*, 2015. 2

[14] C. Gan, T. Yao, K. Yang, Y. Yang, and T. Mei. You lead, we exceed: Labor-free video concept learning by jointly exploiting web videos and images. *CVPR*, 2016. 2

[15] G. Gkioxari and J. Malik. Finding action tubes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 759–768, 2015. 1

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[17] S. Herath, M. Harandi, and F. Porikli. Going deeper into action recognition: A survey. *Image and Vision Computing (IVC)*, pages –, 2017. 2

[18] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *arXiv preprint arXiv:1612.01925*, 2016. 2, 7

[19] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015. 5

[20] 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, Jan 2013. 2

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

[22] A. Klaser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, pages 275–1, 2008. 2

[23] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool. Fast optical flow using dense inverse search. In *European Conference on Computer Vision*, pages 471–488. Springer, 2016. 7

[24] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *Proc. Int. Conference on Computer Vision (ICCV)*, pages 2556–2563. IEEE, 2011. 2, 7

[25] I. Laptev. On space-time interest points. *International journal of computer vision*, 64(2-3):107–123, 2005. 2

[26] X. Long, C. Gan, G. de Melo, J. Wu, X. Liu, and S. Wen. Attention clusters: Purely attention based local feature integration for video classification. *CVPR*, 2018. 2

[27] J. Y.-H. Ng, J. Choi, J. Neumann, and L. S. Davis. Actionflownet: Learning motion representation for action recognition. *arXiv preprint arXiv:1612.03052*, 2016. 2, 6, 8

[28] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, June 2015. 1, 2, 7

[29] X. Peng and C. Schmid. Multi-region two-stream r-cnn for action detection. In *European Conference on Computer Vision*, pages 744–759. Springer, 2016. 1

[30] Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 5534–5542. IEEE, 2017. 2

[31] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. *arXiv preprint arXiv:1611.00850*, 2016. 2

[32] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In

Advances in neural information processing systems, pages 91–99, 2015. 1

[33] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pages 568–576, 2014. 1, 2, 7, 8

[34] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 2, 7

[35] D. Teney and M. Hebert. Learning to extract motion from videos in convolutional neural networks. In *Asian Conference on Computer Vision*, pages 412–428. Springer, 2016. 2

[36] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4489–4497, 2015. 1, 2, 6, 7, 8

[37] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3551–3558, 2013. 2, 8

[38] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4305–4314, 2015. 7, 8

[39] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015. 7

[40] 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 *Proc. European Conference on Computer Vision (ECCV)*, pages 20–36. Springer, 2016. 1, 2, 5, 7, 8

[41] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1385–1392, 2013. 7

[42] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l 1 optical flow. *Pattern Recognition*, pages 214–223, 2007. 1, 2, 3

[43] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang. Real-time action recognition with enhanced motion vector cnns. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2718–2726. IEEE, 2016. 2

[44] W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao. A key volume mining deep framework for action recognition. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1991–1999. IEEE, 2016. 7

[45] Y. Zhu, Z. Lan, S. Newsam, and A. G. Hauptmann. Hidden two-stream convolutional networks for action recognition. *arXiv preprint arXiv:1704.00389*, 2017. 2