Designing Motion Representation in Videos

A THESIS SUBMITTED TO
THE FACULTY OF ENGINEERING AND INFORMATION TECHNOLOGIES
OF THE UNIVERSITY OF SYDNEY
IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF PHILOSOPHY

Shuyang Sun

Supervisor: Dr. Wanli Ouyang

School of Electrical and Information Engineering
Faculty of Engineering and Information Technologies
The University of Sydney

Oct 2018

Authorship Attribution Statement

The work presented in this thesis is published as [34] in the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

Designing Motion Representation in Videos

Shuyang Sun (Email: shuyang.sun@sydney.edu.au)

Supervisor: Dr. Wanli Ouyang

School of Electrical and Information Engineering

Faculty of Engineering and Information Technologies

The University of Sydney

Copyright in Relation to This Thesis

Statement of Original Authorship

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes. I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Signature:

Shuyang Sun

To those whom I love and those whom love me.

Abstract

Motion representation plays a vital role in the vision-based human action recognition in videos. Generally, the information of a video could be divided into spatial information and temporal information. While the spatial information could be easily described by the RGB images, the design of the motion representation is yet a challenging problem. Current available motion representations are either time-consuming or unrobust. A number of video-related tasks in the field of computer vision, e.g., the action recognition, video detection etc., have strict requirements for both the accuracy and efficiency. However, current state-of-the-art motion representations could only be able to handle one of the problems. The informative modalities could only be responsible for the performance rather than the efficiency, while on the other hand, the fast approach to extract motion representations could only achieve moderate accuracy for tasks defined in the field of computer vision. In order to solve this challenging problem, we design the feature according to two principles. First, to guarantee the robustness, the temporal information should be highly related to the informative modalities, e.g., the optical flow. Second, only basic operations could be applied to make the computational cost affordable when extracting the temporal information. Based on these principles, in this study, we introduce a novel compact motion representation for video action recognition, named Optical Flow guided Feature (OFF), which enables the network to distill temporal information through a fast and robust approach. The OFF is derived from the definition of optical flow and is orthogonal to the optical flow. The derivation also provides theoretical support for using the difference between two frames. By directly calculating pixel-wise spatio-temporal gradients of the deep feature maps, the OFF could be embedded in any existing CNN based video action recognition framework with only a slight additional cost. It enables the CNN to extract spatio-temporal information, especially the temporal information between frames simultaneously. This simple but powerful idea is validated by experimental results. The network with OFF fed only by RGB inputs achieves a competitive accuracy of 93.3% on UCF-101, which is comparable with the result obtained by

two streams (RGB and optical flow), but is 15 times faster in speed. Experimental results also show that OFF is complementary to other motion modalities such as optical flow. When the proposed method is plugged into the state-of-the-art video action recognition framework, it has 96.0% and 74.2% accuracy on UCF-101 and HMDB-51 respectively. The code for this project is available at this link for full re-production.

Keywords

Action Recognition, Video Classification, Motion Representation, Deep Learning, Computer Vision

Acknowledgments

This research would not have been possible without the kind and selfless assistance of many people. As a freshman of computer vision, it is the Dr. Zhanghui Kuang and Dr. Zhanpeng Zhang at SenseTime Research that introduced the fantastic and incredible world of computer vision to me. It was their patience and insistence that helped me get through the tough but memorable gap year after the graduation of my undergraduate study. As an absolute newbie of this field, they taught me how to use the fundamental deep learning framework from scratch, and also any detail of the basic knowledge in the field of computer vision. It was a wonderful time in SenseTime that took me into the fantastic world of artificial intelligence. I would like to appreciate all the friends including Wei Zhang, Shi Qiu, Litong Feng, Sirui Xie, Chunxiao Liu, Xiaoyu Yue, Chengxi Yang, Xingyu Zeng, Shuai Yi, etc., for their kind assistance and thoughtful discussion. Hope our paths will cross again one day, and hope we can meet even better of us at that time.

As for becoming a researcher, my supervisor Wanli Ouyang plays a key role in cultivating me into a junior researcher. As a start-up group here in USYD, we had nothing but the time and freedom. Struggling with the huge amount of ants and the intolerable non-air conditioning summer, we did have a tough undertaking as every detail of the projects for me was a brand-new topic. It was his encouragement and the daily or even hourly discussion that makes the ideas finally came into reality. As the first graduate of SIGMA Lab, I sincerely wish that our group could one day become the paradise for conducting research in computer vision.

Finally, I would like to thank my beloved parents and my girl friend, Yi Zhou, who always unconditionally offer me the love and encouragement. It is their trust in me that helps me come through the troubles and difficulties.

Chapter 1

Introduction

Video action recognition has received longstanding attentions in the community of computer vision for decades. It aims at automatically recognizing human action from video sequences. Since CNNs have achieved great successes in image classification and other related tasks [20, 30, 35, 15, 50, 52, 24], lots of CNN based methods have been proposed by considering video action recognition as a classification task [5, 44, 23, 51, 11, 10, 9, 42, 43, 32, 29]. Compared to the image classification methods, temporal information is the key ingredient of video action recognition. Therefore, the design of the motion representations comes to be an important topic in video related tasks.

Optical flow is found to be a useful motion representation in video action recognition, including the Two-Stream-based [29, 44] and 3D convolution-based methods [5]. However, extracting dense optical flows is still inefficient. It costs over 90% of the whole run-time in a two-stream based pipeline both at training and testing phases. Moreover, 3D convolutions on RGB input can also capture temporal information, but the RGB-based 3D CNN still does not perform on par with its two-stream version. Other motion descriptors, e.g., 3DHOG [19], improved Dense Trajectory [40], and motion vector [51], are either inefficient or not so effective as optical flow.

How to design/use motion representation that is both fast and robust? To this end, the required computation should be economical and the representation should be sufficiently guided by the motion information. Taking the above requirements into consideration, we propose the Optical Flow guided Feature (OFF), which is fast to compute and can comprehensively represent motion dynamics in a video clip.

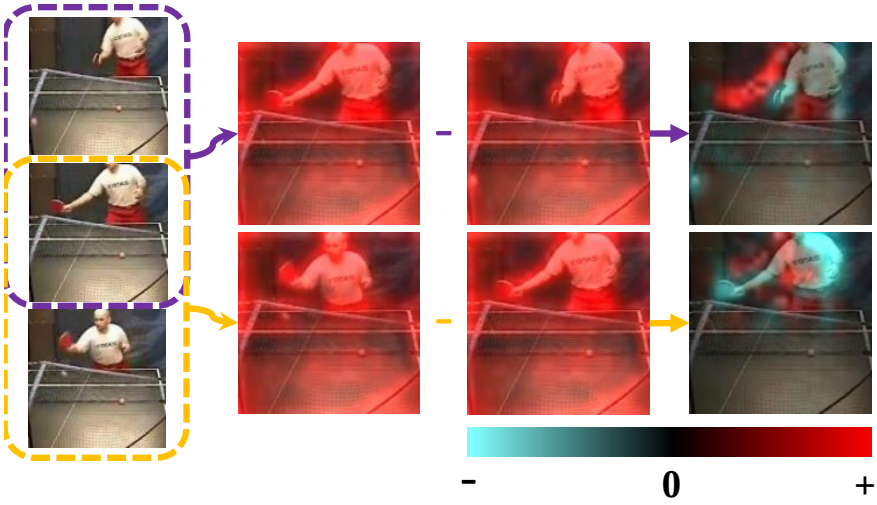

Figure 1.1: **The Optical Flow guided Feature (OFF).** Left column: input frames. Middle two columns: standard deep features before applying OFF onto two frames. Right column: temporal difference in OFF. The colors red and cyan are used respectively for positive and negative values. The feature difference between two frames is valid and comprehensive in representing motion information. Best seen in color and zoomed in.

In this paper, we define a new feature representation from the orthogonal space of optical flow on the feature level [16]. Such definition brings the guidance from optical flow here to the representation, therefore, we name it as the Optical Flow guided Feature (OFF). The feature consists of spatial gradients of feature maps in horizontal and vertical directions, and temporal gradients obtained from the difference between feature maps from different frames. Since all the operations in OFF are differentiable, the whole process is end-to-end trainable when OFF is plugged into one CNN architecture. Actually the OFF unit only consists of pixel-wise operators on CNN features. These operators are fast to apply, and enable the network with RGB input to capture spatial and temporal information simultaneously.

One vital component in OFF is the difference between features from different images/segments. As shown in Fig. 1.1, the difference between the features from two images provides representative motion information that can be conveniently employed by CNNs. The negative values in the difference image depict the locations where the body parts/objects disappear, while the positive values represent where they emerge. This pattern of disappearing at one location and emerging at another location can be easily treated as a specific motion pattern and captured by later CNN layers. The temporal difference could be further combined with the spatial gradients such that the constituted OFF is guided by the optical flow on feature level according to our derivation in later section. Moreover, calculation of the motion dynamics at the feature level is faster and also

more robust because 1) it enables the spatial and temporal networks with the capability of weight sharing and 2) deeply learned features convey more semantic and discriminative representations with reliable elimination of local and background noises in the raw frames.

Our work has two main contributions.

First, **OFF is a fast and robust motion representation**. OFF is fast to enable over 200 frames per second with *only RGB* as the input and is derived from and guided by the optical flow. Taking only RGB from videos, experimental results show that the CNN with OFF is close in performance when compared with the state-of-the-art optical flow based algorithms. The CNNs with OFF can achieve 93.3% on UCF-101 with *only RGB* as the input, which is currently state-of-the-art among the RGB-based action recognition methods. When plugging OFF in the state-of-the-art action recognition method [44] in a Two-Stream manner (RGB + Optical Flow), the performance of our algorithm could result in 96.0% on UCF-101 and 74.2% on HMDB-51.

Second, **an OFF equipped network can be trained in an end-to-end fashion**. In this way, the spatial and motion representations can be jointly learned through a single network. This property is friendly for video tasks on large-scale datasets, as it may not require the network to pre-compute and store motion modalities for training. Besides, the OFF can be used between images/segments in a video clip both on image level and feature level. With the capacity to extract temporal information on feature level, the network could save its number of parameters and computational cost by sharing network weights from the spatial and temporal two streams. That is, we somehow combine the independent two streams into one, which also enables the network to capture spatial and temporal representations simultaneously.

Moreover, **we provide theoretical support for the use of temporal difference between frames in video based tasks.** Previous works have shown that the temporal difference between frames is useful in video related tasks [44], however, there is no theoretical evidence to help explain why this simple idea works that well. In this work, we will give a theoretical derivation to illustrate the relationship between the temporal difference and optical flow, and suggest a way of using such a modality in video related tasks.

The rest of this paper is organized as follows. Chapter 2 introduces recent methods that are related to our work. Chapter 3 illustrates the definition of OFF and details of our proposed method. Chapter 4 explains our implementation method in CNN. Our experimental results are summarized in Chapter 5, with concluding remarks in conclusion Chapter 6.

Chapter 2

Related Works

Traditional methods extracted hand-craft local visual features such as 3DHOG [19], Motion Boundary Histograms (MBH) [8], improved Dense Trajectory (iDT) [40, 41] and then encoded them into sparse or compact feature vectors which were fed into classifiers [27, 26]. Deeply learned features were then found to perform better than hand-crafted features for action recognition [29, 42].

As a significant breakthrough in action recognition, Two-Stream based frameworks used the deep CNN to learn from the hand-craft motion features like optical flow and iDT [29, 42, 51, 44, 9, 48, 5, 36, 11, 12]. These attempts have achieved remarkable progress in improving the recognition accuracy, but still rely on the pre-computed optical flow or iDT, which constrains the speed of the whole framework.

In order to obtain the motion modality in a fast way, recent works used optical flow only at the training stage [23], or proposed motion vector as the simplified version of optical flow [51]. These attempts have produced degraded optical flow results and still did not perform on par with the approaches using traditional optical flow as the input stream.

Many approaches learn to capture the motion information directly from input frames using 3D CNN [36, 38, 5, 37, 9, 39]. Boosted by the temporal convolution and pooling operations, 3D CNN could distill the temporal information between consecutive frames without segmenting them into short snippets. Compared with the learning of filters to capture motion information, our OFF is a principled representation mathematically derived from the optical flow. 3D CNN, constrained by network design, training sample, and parameter regularization like weight decay, may not be able to learn good motion representation like OFF. Therefore, current state-of-the-art

3D CNN based algorithms still rely on traditional optical flow to help the networks to capture motion patterns. In comparison, our OFF 1) well captures the motion patterns so that RGB stream with OFF performs on par with two stream methods, and 2) is also complementary to other motion representations like optical flow.

Moreover, there are some approaches used for improving the speed of estimating the optical flow [54, 23, 51]. However, the performance of these works is still moderate, and as for another aspect, unfortunately, these representations are still illustrating the frame-level temporal information, which will also lead to the high computational expense when long-term temporal information is required. This phenomenon suggests that the motion representation may need to describe the variance beyond the consecutive frames in order to save the computational resources.

The work [54] has proposed to embed an additional network into the two-stream framework to learn the optical flow through an unsupervised approach, and another study [23] simply guide the spatial network with optical flow serving as the ground truth. The work [51] proposes to transfer the weights from the network fed by optical flow to the network fed by motion vector. However, these attempts are either not efficient enough or not effective enough.

To capture long-term temporal information from videos, one intuitive approach is to introduce the Long Short-Term Memory (LSTM) module as an encoder to encode the relationship between the sequence-illustrating deep features [48, 33, 28]. LSTM can still be applied on the OFF. Therefore, our OFF is complementary to these methods.

Concurrent with our work, another state-of-the-art method applies a strategy called *ranked pool* [13] that generates a fast video-level descriptor, namely, the *dynamic images* [3]. However, the very nature in design and implementation between the dynamic images and ours are different. The dynamic images are designed to summarize a series of frames while our method is designed to capture the motion information related to optical flow.

Chapter 3

Optical Flow Guided Feature

Our proposed OFF is inspired by the famous brightness constant constraint defined by traditional optical flow [16]. It is formulated as follows:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t), \tag{3.1}$$

where $I(x, y, t)$ denotes the pixel at the location (x, y) of a frame at time t. For frames t and $(t + \Delta t)$, Δx and Δy are the spatial pixel displacement in x and y axes respectively. It assumes that for any point that moves from (x, y) at frame t to $(x + \Delta x, y + \Delta y)$ at frame $t + \Delta t$, its brightness keeps unchanged over time. When we apply this constraint at the feature level, we have

$$f(I; w)(x, y, t) = f(I; w)(x + \Delta x, y + \Delta y, t + \Delta t), \tag{3.2}$$

where f is a mapping function for extracting features from the image I. w denotes the parameters in the mapping function. The mapping function f can be any differentiable function. In this paper, we employ trainable CNNs consisted of stacks of convolution , ReLU, and pooling operations. According to the definition of optical flow, we assume that $p = (x, y, t)$ and obtain the equation as follows:

$$\frac{\partial f(I; w)(p)}{\partial x}\Delta x + \frac{\partial f(I; w)(p)}{\partial y}\Delta y + \frac{\partial f(I; w)(p)}{\partial t}\Delta t = 0. \tag{3.3}$$

By dividing Δt in both sides of Equation 3.3, we obtain

$$\frac{\partial f(I; w)(p)}{\partial x}v_x + \frac{\partial f(I; w)(p)}{\partial y}v_y + \frac{\partial f(I; w)(p)}{\partial t} = 0, \tag{3.4}$$

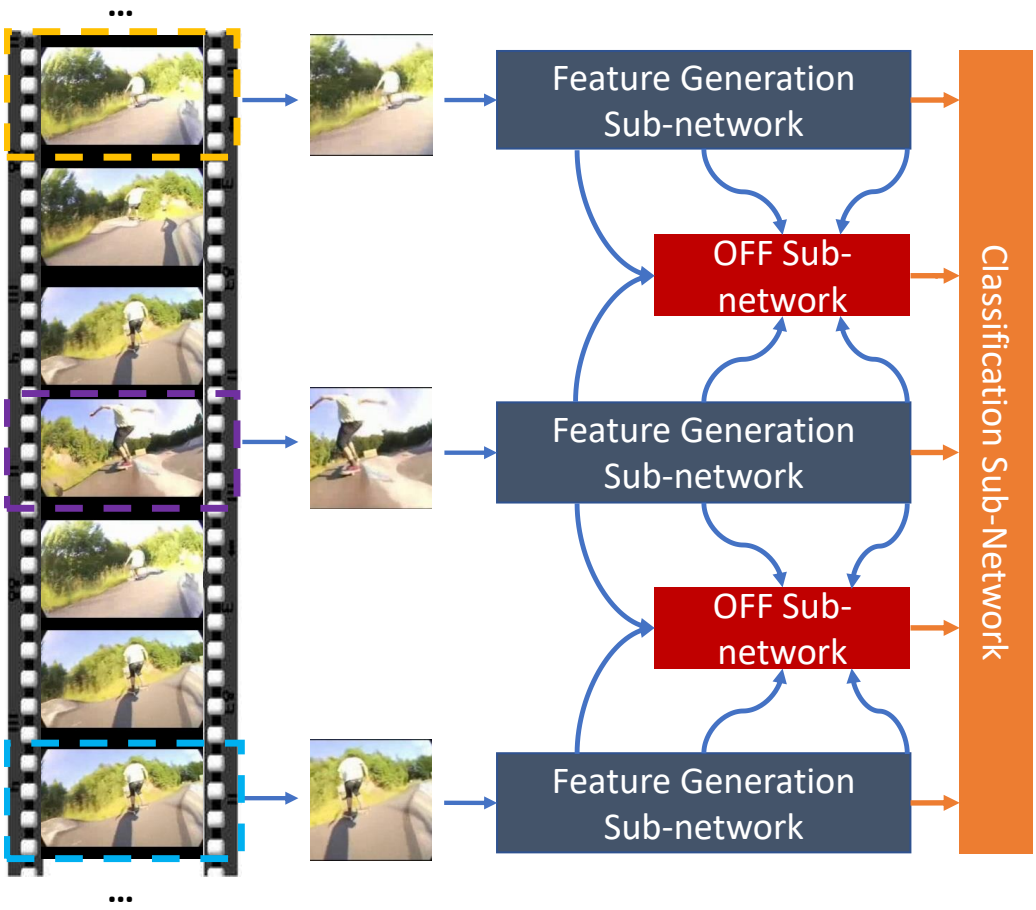

Figure 3.1: **Network architecture overview.** The feature generation sub-network extracts feature for each frame sampled from the video. Based on the features from two adjacent frames extracted by the feature generation sub-networks, a OFF sub-network is applied to generate the OFF for further classification. The scores from all sub-networks are fused to get the final result.

where $p = (x, y, t)$, and (v_x, v_y) denotes the two dimensional velocity of feature point at p. $\frac{\partial f(I;w)(p)}{\partial x}$ and $\frac{\partial f(I;w)(p)}{\partial y}$ are the spatial gradients of $\partial f(I;w)(p)$ in x and y axes respectively. $\frac{\partial f(I;w)}{\partial t}$ is the temporal gradient along time axis.

As a special case, when $f(I;w)(p) = I(p)$, then $f(I;w)(p)$ simply represents pixel at p. In this special case, (v_x, v_y) are called optical flow. Optical flow is obtained by solving an optimization problem with the constraint in Equation 3.4 for each p [1, 4, 2]. Here in this case, the term $\frac{\partial f(I;w)(p)}{\partial t}$ represents the difference between RGB frames. Previous works have shown that the temporal difference between frames is useful in video related tasks [44], however, there is no theoretical evidence to help explain why this simple idea works that well. Here, we can find its correlation to spatial features and optical flow.

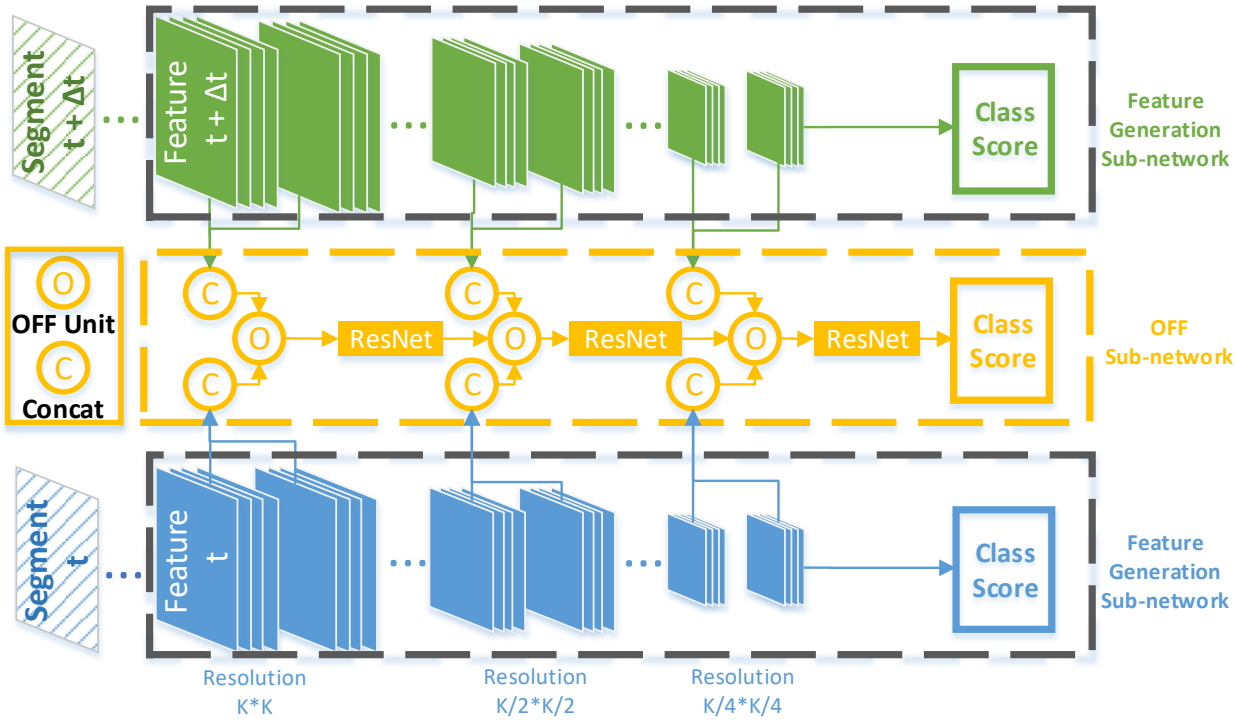

Figure 3.2: **Network architecture overview for two segments.** The inputs are two segments in blue and green colors that are separately fed into the feature generation sub-network to obtain basic features. In our experiment, the backbone for each feature generation sub-network is the BN-Inception [35]. Here K represents the largest side length of the square feature map selected to undergo the OFF sub-network for obtaining the OFF features. The OFF sub-network consists of several OFF units, and several residual blocks [15] are connected between OFF units from different levels of resolution. These residual blocks constitute a ResNet-20 when seen as a whole. The scores obtained by different sub-networks are supervised independently. Detailed structure of the OFF unit is shown in Figure 4.1.

We generalize the representation of optical flow from pixel $I(p)$ to feature $f(I;w)(p)$. In this general case, $[v_x, v_y]$ are called the feature flow. We can see from Equation 3.4 that $\vec{F}(I;w)(p) = [\frac{\partial f(I;w)(p)}{\partial x}, \frac{\partial f(I;w)(p)}{\partial y}, \frac{\partial f(I;w)(p)}{\partial t}]$ is orthogonal to the vector $[v_x, v_y, 1]$ containing feature-level optical flow. $\vec{F}(I;w)(p)$ changes as the feature-level optical flow changes. Therefore, $\vec{F}(I;w)(p)$ is guided by the feature-level optical flow. We call $\vec{F}(I;w)(p)$ as *Optical Flow guided Feature (OFF)*. The OFF $\vec{F}(I;w)(p)$ encodes the spatial-temporal information orthogonally and complementarily to the feature-level optical flow (v_x, v_y). In the next section, detailed implementation of OFF and its usage for action recognition are introduced.

Chapter 4

Using Optical Flow Guided Feature in Convolutional Neural Network

---

4.0.1 Network Architecture

Network Architecture Overview. Figure 3.1 shows an overview of the whole network architecture. The network consists of three sub-networks for different purposes: feature generation sub-network, OFF sub-network and classification sub-network. The feature generation sub-network generates basic features using common CNN structures. In the OFF sub-network, the OFF features are extracted using the features from the feature generation sub-network, and then several residual blocks are stacked for obtaining the refined features. The features from the previous two sub-networks are then used by the classification sub-network for obtaining the action recognition results. The Figure 3.2 exhibits the more detailed network structure with the inputs of two segments. As shown in Figure 3.2, we extract features from multiple layers on a specific level with the same resolution by concatenating them together and feed them into one OFF unit. The whole network has 3 OFF units with different scales. The details about the structure of each sub-network is discussed as follows.

Feature Generation Sub-network. The basic features $f(I)$ (equivalent to the representation $f(I; w)$ in previous section) are extracted from the input image using several convolutional layers with Rectified Linear Unit (ReLU) for non-linear function and max-pooling for down-sampling. We select BN-Inception [35] as the network structure to extract feature maps. The feature generation sub-network can be replaced by any other network architecture.

OFF Sub-network. The OFF sub-network consists of several OFF units. Different units

use basic features $f(I)$ from different depths. As shown in Figure 4.1, an OFF unit contains an OFF layer to generate the OFF. Each OFF layer contains a 1×1 convolutional layer for each piece of feature, and a set of operators including sobel and element-wise subtraction for OFF generation. After the OFF is obtained, the OFF unit will concatenate them together with features from the lower level, then the combined features will be output to the following residual blocks.

The OFF layer is responsible for generating the OFF from the basic features $f(I)$. Figure 4.1 shows the detailed implementation the OFF layer. According to Equation 3.3, the OFF should consist of both spatial and temporal gradient of the feature. Denote $f(I, c)$ as the cth channel of the basic feature $f(I)$. Denote $\mathcal{F}_x$ and $\mathcal{F}_y$ as the OFF for gradients of x and y directions respectively, which correspond to spatial gradients. We apply the Sobel operator for spatial gradient generation as follows:

$$\mathcal{F}_x = \left\{ \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} * f(I, c) \middle| c = 0 \ldots, N_c - 1 \right\} \tag{4.1}$$

$$\mathcal{F}_y = \left\{ \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * f(I, c) \middle| c = 0, \ldots, N_c - 1 \right\} \tag{4.2}$$

where $*$ denotes a convolution operation, and the constant N_c indicates the number of channels of the feature $f(I)$. Denote $\mathcal{F}_t$ as the OFF for gradients at the temporal directions. Temporal gradient is obtained by element-wise subtraction as follows:

$$\mathcal{F}_t = \{ f_t(I, c) - f_{t-\Delta t}(I, c) | c = 0, \ldots, N_c - 1 \} \tag{4.3}$$

With the features $\mathcal{F}_x$, $\mathcal{F}_y$, and $\mathcal{F}_t$ obtained above, we concatenate them together with the features from the lower level as the output of the OFF layer. We use a 1×1 convolutional layer

before the sobel and subtraction operations to reduce the number of channels. In our experiments, the channel dimension is reduced to 128 regardless of how many the input channels are. Then the feature is fed into the OFF unit to calculate the OFF we defined in previous section. After the OFF is obtained, several residual blocks designed in [15] are connected between the OFF units at different levels of resolution as refinement. The dimensionality of OFF is further reduced in the residual block adjacent to the OFF unit for saving computation and the number of parameters. The residual blocks on different levels of resolution finally constitute a ResNet-20. Note that there is no Batch Normalization [17] operation applied in our residual network in order to avoid the over-fitting problem.

The OFF unit can be applied for CNN layers on different levels. The inputs of one OFF unit include the basic deep features from two segments, and the feature from the OFF unit on the previous feature level if it exists. In this way, the OFF at the previous semantic level can be used for refining the OFF at the current semantic level.

Classification Sub-network. The classification sub-network takes features from different sources and uses multiple inner-product classifiers to obtain multiple classification scores. The classification scores of all sampled frames are then combine by averaging for each feature generation sub-network, or OFF sub-network. The OFF at a semantic level can be used to produce a classification score at the training stage, which is learned using its corresponding loss. Such strategy has been proved to be useful in many tasks [35, 46, 22]. In the testing phase, scores from different sub-networks could be assembled for better performance.

4.0.2 Network Training

Action recognition is treated as a multi-class classification problem. Followed by the settings in TSN, as there are multiple classification scores produced by each segment, we need to fuse them all in each sub-network separately to generate a video-level score for loss calculation. Here, for the OFF sub-networks, the features produced by the output of OFF sub-network for the tth segment on level l is denoted by $\mathcal{F}_{t,l}$. The classification score for segment t on the level l using $\mathcal{F}_{t,l}$ is denoted by $\mathbf{G}_{t,l}$. The aggregated video-level score at level l is denoted by G_l. The video-level action classification score G_l is obtained by:

$$G_l = \mathcal{G}(\mathbf{G}_{0,l}, \ldots, \mathbf{G}_{1,l}, \ldots, \mathbf{G}_{N_{t-1}-1,l}), \tag{4.4}$$

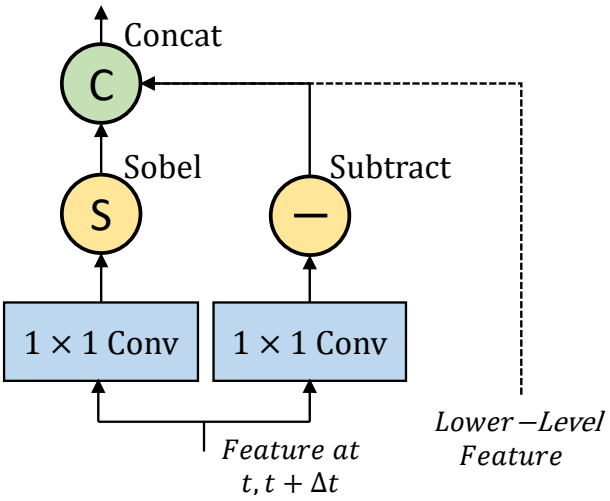

Figure 4.1: **Detailed architecture of OFF unit.** A 1x1 convolution layer is connected to the input basic feature for dimension reduction. After that, we utilize the Sobel operator and element-wise subtraction to calculate the spatial and temporal gradients respectively. The combination of gradients constitutes the OFF, and the sobel operator, subtracting operator and the 1×1 convolution layers before them constitute a OFF layer.

where N_t denotes the number of frames for extracting features. The aggregation function denoted by $\mathcal{G}$ is used for summarizing the scores predicted from different segments along time. Following the investigations in TSN, $\mathcal{G}$ is implemented by average pooling for better performance [44]. As for the feature generation sub-network, the above equations are also applicable. While as we do not need intermediate supervision for feature generation sub-network, the feature $\mathcal{F}_{t,l}$ at level l for segment t is simply equivalent to the final feature output of the sub-network.

To update the parameters of the whole network, the loss is set to be the standard categorical cross-entropy loss. As the sub-network for each feature level is supervised independently, a loss function is used for each level as:

$$\mathcal{L}_l(y, G_l) = -\sum_{c=1}^{C} y_c (G_{l,c} - \log \sum_{j=1}^{C} e^{G_{l,j}}). \tag{4.5}$$

where C is the number of action categories, $G_{l,c}$ is the estimated score for class c from the features at level l, and y_c represents the ground-truth class label. By using this loss function we can optimize the network parameters through back-propagation. Detailed implementation of training is described as follows.

Two-stage Training Strategy. Training of the whole network consists of two stages. The

first stage indeed is to apply existing approaches, e.g. TSN [44], to train the feature generation sub-network. At the second stage, we train the OFF and classification sub-network with all the weights in feature generation sub-network frozen. The weights of OFF sub-network and classification sub-network are learned from scratch. The whole network could be further fine-tuned in an end-to-end manner, however, we do not find significant gain in this stage. To simplify the training process, we only train the network using the first two stages.

Intermediate Supervision during Training. Intermediate supervision has been proven to be practical training strategy in many other computer vision tasks [22, 46, 47, 25, 6]. As the OFF sub-networks are fed by intermediate inputs, here we add the intermediate supervision on each level to get better OFFs on each level of resolution.

Reducing the Memory Cost. As our framework consists of several sub-networks, it costs more memory than the original TSN framework, which extracts and stores motion frames before training CNNs, and trains several networks independently. In order to reduce the computational and memorial cost, we sample less frames in the training phase than in the testing phase, and still obtain satisfactory results.

However, the time duration between segments may be varied if we sample different number of segments between training and testing. According to our definition in equation 3.3, only when the denotation Δt is a fixed constant, the equation 3.4 could be derived from the equation 3.3. If we sample different frames between training and testing, the time interval Δt may be inconsistent, which makes our definition to be invalid and influences the final performance. In order to keep time interval consistent between training and testing, we design the sampling scheme carefully. Therefore, during training, we sample frames from a video as follows:

Let α be the number of frames sampled for training, and β be the number for testing. In training phase, a video with length L, $L >= \beta$ would be divided into β segments. Each segment has length $\lfloor L/\beta \rfloor$. We randomly select p from $0, 1, \ldots, L - 1 - (\alpha - 1) * \lfloor L/\beta \rfloor$, where p is treated as a frame seed. Then the whole training set is constructed as $\{p, p + \lfloor L/\beta \rfloor, ..., p + (\alpha - 1) * \lfloor L/\beta \rfloor\}$, which has interval $\lfloor L/\beta \rfloor$. In testing phase, we sample the images using the same interval $\lfloor L/\beta \rfloor$ as that in the training phase.

4.0.3 Network Testing

As there are multiple classification scores produced by different sub-networks, we need to fuse them all in testing phase for better performance. In this study, we assemble scores from the feature generation sub-network and the last level of OFF sub-network by a simple summing operation. We select to test our model based on a state-of-the-art framework TSN [44]. The testing setting under the TSN framework is illustrated as follows:

Testing under TSN Framework. In the testing stage of TSN, 25 segments are sampled from RGB, RGB difference, and optical flow. However, the number of frames in each segment is different among these modalities. We use the original settings adopted by TSN to sample 1, 5, 5 frames per segment for RGB, RGB difference, and optical flow respectively. The input of our network is 25 segments, where the tth segment is treated as the Frame t in Figure 3.2. In this case, the features extracted by a separate branch of our feature generation sub-network is for a segment instead of a frame when using TSN. Other settings are kept to be the same as those in TSN.

Chapter 5

Experiments and Evaluations

5.1 Experiments and Evaluations

In this section, datasets and implementation details used in experiments will be first introduced. Then we will explore the OFF and compare it with other modalities under current state-of-the-art frameworks. Moreover, as our method can be extended to other modalities such as RGB difference and optical flow, we will show how such a simple operation could improve the performance for input with different modalities. Finally, we will discuss the meaning and difference between the OFF and other motion modalities such as optical flow and RGB difference.

5.1.1 Datasets and Implementation Details

Evaluation Datasets. The experimental results are evaluated on two popular video action datasets, UCF-101 [31] and HMDB-51 [21]. The UCF-101 dataset has 13320 videos and is divided into 101 classes, while the HMDB-51 contains 6766 videos and 51 classes. Our experiments follow the officially offered scheme which divides a dataset into 3 training and testing splits and finally calculating the average accuracy over all 3 splits. We prepare the optical flow between frames before training by directly using the OpenCV implemented algorithm [49].

Implementation Details. We train our model with 4 NVIDIA TITAN X GPU, under the implementation on Caffe [18] and OpenMPI. We first train the feature generation sub-networks using the same strategy provided in the corresponding method [44]. Then at the second stage,

Method	Speed (fps)	Acc.
TSN(RGB) [44]	680	85.5%
TSN(RGB+RGB Diff) [44]	340	91.0%
TSN(Flow) [44]	14	87.9%
TSN(RGB+Flow) [44]	14	94.0%
RGB+EMV-CNN [51]	390	86.4%
MDI+RGB [3]	<131	76.9%
Two-Stream I3D (RGB+Flow) [5]	<14	93.4%
RGB+OFF(RGB)+ RGB Diff+OFF(RGB Diff)	206	**93.3%**

Table 5.1: Experimental results of accuracy and efficiency for different real-time video action recognition methods on *UCF-101 over three splits*. Here the notation *Flow* represents the motion modality Optical Flow. Note that our OFF based algorithm could achieve the state-of-the-art performance among real-time algorithms.

we train the OFF sub-networks from scratch with all parameters in the feature generation sub-networks frozen. The mini-batch stochastic gradient descent algorithm is adopted here to learn the network parameters. When the feature generation sub-networks are fed by RGB frames, the whole training procedure for OFF sub-network takes 20000 iterations to converge with the learning rate initialized at 0.02 and decreased to its 0.1 using multi-step policy at the iteration 10000, 15000 and 18000. When input changes to temporal modality like optical flow, the learning rate is initialized at 0.05, and other policies are kept the same with what have been proposed in RGB. The batch size is set to 128 and all the training strategies described in previous sections are applied. When evaluating on UCF-101 and HMDB-51, we add dropout modules on spatial stream of OFF. There is no difference on training parameters for different modalities. However, when the input is RGB difference or optical flow, it would cost more time in both training and testing stages as more frames are read into the network.

5.1.2 Experimental Investigations on OFF.

In this section, we will investigate the performance of OFF under the TSN framework. The analysis for the performance of single and multiple modalities, and the performance comparison between the state-of-the-art will be shown. All the results for OFF based networks are trained with the same network backbone and strategies illustrated in previous sections for fair comparison.

RGB	OFF (RGB)	RGB Diff	OFF (RGB Diff)	Flow	OFF (Flow)	Speed (fps)	Acc.
✓						680	85.5%
✓	✓					450	90.0%
✓		✓				340	90.7%
✓	✓	✓				257	92.0%
✓	✓	✓	✓			206	93.0%
✓				✓		14	93.5%
✓	✓			✓		14	95.1%
✓	✓			✓	✓	14	95.5%

Table 5.2: Experimental results for different modalities using the OFF on *UCF-101 Split1*. Here Flow denotes the optical flow. OFF(*) denotes the use of OFF for the input *. For example, OFF(RGB) denotes the use of OFF for RGB input. The speed here illustrates the time cost for network forward. The results for RGB and RGB + Flow are from [44]. The OFF(RGB) provides a strong 4.5% improvement when fusing with RGB.

Efficiency Evaluation. In this experiment, we evaluate the efficiency between the OFF based method and other state-of-the-art methods. The experimental results for efficiency and accuracy for different algorithms are summarized in Table 5.1. OFF(RGB) denotes our use of OFF for the network with RGB input, in this case, the OFF is acquired from spatial deep features. As one special case, the denotation *RGB Diff* represents the OFF calculated directly from consecutive RGB frames on the input level instead of on the feature level. After applying the OFF calculation to RGB frames, the processed inputs could be fed into the feature generation sub-network and the generated feature maps could be again used to calculate their corresponding OFF features on the feature level. The other methods we compared here includes TSN [44] with different inputs, motion vector based RGB+EMV-CNN [51], dynamic image based CNN [3] and current state-of-the-art 3D-CNN with two stream [5]. From the Table 5.1, by applying the OFF to the spatial features and the RGB inputs, we can achieve a competitive accuracy 93.3% with only RGB inputs on the UCF-101 over three splits, which is even comparable with some Two-Stream based methods such as [5, 44]. Besides, our methods is still very efficient under this kind of settings. The whole network could run over 200 fps, while other methods listed here are either inefficient or not so effective as the Two-Stream based approaches.

Effectiveness Evaluation. In this part, we try to investigate the robustness of OFF when applying to different kinds of input. According to the definition in equation 3.4, we can replace the image I from RGB image to optical flow or RGB difference image to extract OFF on

	RGB	Hyp-Net + RGB	OFF(RGB) + RGB
Acc.	85.5%	86.0%	90.0%

Table 5.3: Experimental results of accuracy for hypercolumn network and the comparison with OFF on UCF-101 Split1. The denotation "Hyp-Net" indicates the output of hypercolumn network.

feature level for further experiments. Based on the scores predicted by different modalities, we can further improve the classification performance by fusing them together [29, 9, 44, 51]. We carry out the experimental results with various score fusing schemes on UCF-101 split 1, and summarize them in Table 5.2. Table 5.2 shows the results when different kinds of modalities are introduced as the network input. From each block separated by a horizon line, we can find that the OFF is complementary to other kinds of modalities, e.g. RGB and optical flow, and could get a remarkable gain every time the OFF is introduced. Besides, interestingly, the OFF is still working when the input modality is already describing the motion information. This phenomenon indicates that the *acceleration information* between frames might also make a difference in describing the temporal patterns.

Comparison with the Hypercolumns CNN. As our network extracts intermediate deep features from a pre-trained CNN, such *hypercolumn* based network structure may lead to additional gain on specific datasets [14]. Experiment and analysis are conducted to investigate whether the OFF is playing a key role for the improvement. The network architecture and all training strategies for the hypercolumn CNN are the same as that in OFF except for the removal of OFF unit, in other words, the hypercolumn network here is constructed as the same structure of OFF sub-network without OFF unit. In this case, the features from feature generation sub-networks are directly fed into the OFF sub-networks without the calculation of OFF.

From the experimental results shown in Table 5.3, it is clear that, despite the hypercolumn network could get a slight 0.5% improvement on UCF-101 split 1, its final accuracy is still apparently less than the one obtained by OFF(RGB). Therefore, a conclusion could be drawn that it is the OFF calculation rather than the hypercolumn structure that plays the key role in achieving the significant gain.

Comparison with the State-of-the-art. Above all, after the exploration and analysis of the OFF, we show our final result. As what has been done in TSN, we also assemble the

Method	UCF-101	HMDB-51
iDT [40]	86.4%	61.7%
Two-Stream [29]	88.0%	59.4%
Two-Stream TSN [44]	94.0%	68.5%
Three-Stream TSN [44]	94.2%	69.4%
Two-Stream+LSTM [48]	88.6%	-%
TDD+iDT [42]	91.5%	65.9%
LTC+iDT [38]	91.7%	64.8%
KVMDF [53]	93.1%	63.3%
STP [45]	94.6%	68.9%
STMN+iDT [12]	94.9%	72.2%
ST-VLMPF+iDT [7]	94.3%	73.1%
L^2STM [33]	93.6%	66.2%
Two-Stream I3D [5]	93.4%	66.4%
Two-Stream I3D (with Kinetics 300k) [5]	98.0%	80.7%
Ours	**96.0%**	**74.2%**

Table 5.4: Performance comparison to the state-of-the-art methods on UCF-101 and HMDB-51 over 3 splits.

classification scores obtained by different kinds of modalities. We sum the scores produced by each modality together, and get the final version output in Table 5.4. All the results are evaluated in the UCF-101 and HMDB-51 over 3 splits. Our results are obtained by assembling the scores from RGB, OFF(RGB), optical flow and their corresponding version of OFF(optical flow) together. When we add one more score from OFF(RGB Diff), a slight 0.3% gain is obtained compared to the version without it, and finally results in 96.0% on UCF-101 and 74.2% on HMDB-51. Note that we do not introduce improved Dense Trajectories (iDT)[40] into our network as the input. The components of inputs we need to prepare in advance for our final version result only consist of RGB and optical flow.

We compare our result with both the traditional approaches and deep learning based approaches. We obtain $2.0\%/5.7\%$ gain compared with the baseline Two-Stream TSN [44] on UCF-101 [31] and HMDB-51 [21] respectively. Note that the final version TSN takes 3 modalities (RGB, Optical Flow and iDT) as network input. The other compared methods listed in Table 5.4 include iDT [40], Two-Stream ConvNet [29], Two-Stream + LSTM [48], Temporal Deep-convolutional Descriptors (TDD) [42], Long-term Temporal Convolutions (LTC) [38], Key Volume Mining Deep Framework (KVMDF) [53], and also the current state-of-the-art

methods such as Spatio-Temporal Pyramid (STP) [45], Saptio-Temporal Multiplier Network (STMN) [12], Spatio-Temporal Vector [7], Lattice LSTM (L^2STM) [33], and I3D [5]. The method I3D could achieve spectacular performance (98.0% on UCF-101, 80.7% on HMDB-51, over 3 splits) when proposing a new large dataset *Kinetics* for pre-train. While without the pre-training, the method I3D could achieve 93.4% on UCF-101 Split1. From the comparison with all the listed methods, we conclude that our OFF based method allow for state-of-the-art performance in video action recognition.

Chapter 6

Conclusions

6.1 Discussion

In this work, we have discussed the design and use of the motion representations in the task of video action recognition. While the temporal information could be used in many other tasks including temporal action detection, video object detection, video segmentation, video super-resolution, and video compression etc. All these video-based tasks require for accurate and fast motion modalities. As the OFF in this work is only validated in the fundamental task like video classification, in the future, we will apply the OFF on other video-related tasks.

6.2 Conclusion

In this work, we have presented *Optical Flow guided Feature (OFF)*, a novel motion representation derived from and guided by the optical flow. OFF is both fast and robust. By plugging the OFF into CNN framework, the result with only RGB as input on UCF-101 is even comparable to the result obtained by Two-Stream (RGB+Optical Flow) approaches, and at the same time, the OFF plugged network is still very efficient with the speed over 200 frames per second. Besides, it has been proven that the OFF is still complementary to other motion representations like optical flow. Based on this representation, we proposed an new CNN architecture for video action recognition. This architecture outperforms many other state-of-the-art video action recognition methods on two popular video datasets UCF-101 and HMDB-51, and could be used to accelerate the speed of the video based tasks.

References

[1] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *IJCV*, 12(1):43–77, 1994.

[2] J. Bigun, G. H. Granlund, and J. Wiklund. Multidimensional orientation estimation with applications to texture analysis and optical flow. *T-PAMI*, 13(8):775–790, 1991.

[3] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *CVPR*, pages 3034–3042, 2016.

[4] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, pages 25–36. Springer, 2004.

[5] J. Carreira and A. Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. *arXiv preprint arXiv:1705.07750*, 2017.

[6] X. Chu, W. Yang, W. Ouyang, C. Ma, A. L. Yuille, and X. Wang. Multi-context attention for human pose estimation. In *CVPR*, July 2017.

[7] I. Cosmin Duta, B. Ionescu, K. Aizawa, and N. Sebe. Spatio-Temporal Vector of Locally Max Pooled Features for Action Recognition in Videos. In *CVPR*, pages 3097–3106, 2017.

[8] N. Dalal, B. Triggs, C. Schmid, N. Dalal, B. Triggs, C. Schmid, H. Detection, and U. Oriented. Human Detection Using Oriented Histograms of Flow and Appearance. In *ECCV*, pages 428–441, 2006.

[9] A. Diba, A. M. Pazandeh, and L. Van Gool. Efficient two-stream motion and appearance 3d cnns for video classification. *arXiv preprint arXiv:1608.08851*, 2016.

[10] A. Diba, V. Sharma, and L. Van Gool. Deep temporal linear encoding networks. *arXiv preprint arXiv:1611.06678*, 2016.

[11] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, pages 3468–3476, 2016.

[12] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal multiplier networks for video action recognition. In *CVPR*, 2017.

[13] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars. Rank pooling for action recognition. *T-PAMI*, 39(4):773–787, 2017.

[14] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, pages 447–456, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[16] B. G. Horn, Berthold K.P.; Schunck. Determining Optical Flow. *Artificial Intelligence*, 17:185–203, 1981. ISSN 0004-3702.

[17] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[19] A. Klaser, M. Marszalek, and C. Schmid. A Spatio-Temporal Descriptor Based on 3D-Gradients. In *BMVC*, pages 275–1, 2008.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[21] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In *ICCV*, pages 2556–2563, 2011.

[22] A. Newell, K. Yang, and J. Deng. Stacked Hourglass Networks for Human Pose Estimation. In *ECCV*, pages 483–499. Springer, 2016. ISBN 978-3-319-46484-8. doi: 10.1007/978-3-319-46484-8_29. URL http://dx.doi.org/10.1007/978-3-319-46484-8{_}29.

[23] 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.

[24] W. Ouyang, X. Zeng, and X. Wang. Learning mutual visibility relationship for pedestrian detection with a deep model. *IJCV*, 120(1):14–27, 2016.

[25] W. Ouyang, K. Wang, X. Zhu, and X. Wang. Chained cascade network for object detection. In *ICCV*, Oct 2017.

[26] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of Visual Words and Fusion Methods for Action Recognition: Comprehensive Study and Good Practice. *Computer Vision and Image Understanding*, 150:109–125, 2016. ISSN 10773142. doi: 10.1016/j.cviu.2016.03.013. URL http://arxiv.org/abs/1405.4506.

[27] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM'MM*, pages 357–360, 2007. ISBN 9781595937025. doi: 10.1145/1291233.1291311. URL http://dl.acm.org/citation.cfm?id=1291311{%}7B{%}25{%}7D5Cnhttp://portal.acm.org/citation.cfm?doid=1291233.1291311.

[28] Y. Shi, Y. Tian, Y. Wang, W. Zeng, and T. Huang. Learning long-term dependencies for action recognition with a biologically-inspired deep network. In *CVPR*, pages 716–725, 2017.

[29] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, pages 568–576, 2014.

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[31] 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. URL http://arxiv.org/abs/1212.0402.

[32] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, pages 4597–4605, 2015.

[33] L. Sun, K. Jia, K. Chen, D. Y. Yeung, B. E. Shi, and S. Savarese. Lattice Long Short-Term Memory for Human Action Recognition. *arXiv preprint arXiv:1708.03958*, 2017.

[34] S. Sun, Z. Kuang, L. Sheng, W. Ouyang, and W. Zhang. Optical flow guided feature: A fast and robust motion representation for video action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. In *CVPR*, pages 1–9, 2015.

[36] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, pages 4489–4497, 2015.

[37] D. Tran, J. Ray, Z. Shou, S.-F. Chang, and M. Paluri. ConvNet Architecture Search for Spatiotemporal Feature Learning. *arXiv preprint arXiv:1708.05038*, 2017.

[38] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *arXiv preprint arXiv:1604.04494*, 2016.

[39] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *T-PAMI*, 2017.

[40] H. Wang and C. Schmid. Action Recognition with Improved Trajectories. In *ICCV*, pages 3551–3558, 2013.

[41] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *CVPR*, pages 3169–3176. IEEE, 2011.

[42] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *ICCV*, pages 4305–4314, 2015.

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

[44] 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. *ECCV*, pages 20–36, 2016.

[45] Y. Wang, M. Long, J. Wang, and P. S. Yu. Spatiotemporal Pyramid Network for Video Action Recognition. In *CVPR*, pages 1529–1538, 2017.

[46] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *CVPR*, pages 4724–4732, 2016.

[47] W. Yang, S. Li, W. Ouyang, H. Li, and X. Wang. Learning feature pyramids for human pose estimation. In *ICCV*, Oct 2017.

[48] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, pages 4694–4702, 2015.

[49] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. *Pattern Recognition*, pages 214–223, 2007.

[50] X. Zeng, W. Ouyang, J. Yan, H. Li, T. Xiao, K. Wang, Y. Liu, Y. Zhou, B. Yang, Z. Wang, et al. Crafting gbd-net for object detection. *T-PAMI*, 2017.

[51] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang. Real-time action recognition with enhanced motion vector CNNs. In *CVPR*, pages 2718–2726, 2016.

[52] H. Zhao, M. Tian, S. Sun, J. Shao, J. Yan, S. Yi, X. Wang, and X. Tang. Spindle net: Person re-identification with human body region guided feature decomposition and fusion. In *CVPR*, July 2017.

[53] W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao. A key volume mining deep framework for action recognition. In *CVPR*, pages 1991–1999, 2016.

[54] Y. Zhu, Z. Lan, S. Newsam, and A. G. Hauptmann. Hidden Two-Stream Convolutional Networks for Action Recognition. *arXiv preprint arXiv:1704.00389*, 2017.