

Real-time Action Recognition with Enhanced Motion Vector CNNs

Bowen Zhang^{1,2} Limin Wang^{1,3} Zhe Wang¹ Yu Qiao^{1*} Hanli Wang²

¹Shenzhen key lab of Comp. Vis. & Pat. Rec., Shenzhen Institutes of Advanced Technology, CAS, China

²Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai, China

³Computer Vision Lab, ETH Zurich, Switzerland

Abstract

The deep two-stream architecture [23] exhibited excellent performance on video based action recognition. The most computationally expensive step in this approach comes from the calculation of optical flow which prevents it to be real-time. This paper accelerates this architecture by replacing optical flow with motion vector which can be obtained directly from compressed videos without extra calculation. However, motion vector lacks fine structures, and contains noisy and inaccurate motion patterns, leading to the evident degradation of recognition performance. Our key insight for relieving this problem is that optical flow and motion vector are inherent correlated. Transferring the knowledge learned with optical flow CNN to motion vector CNN can significantly boost the performance of the latter. Specifically, we introduce three strategies for this, initialization transfer, supervision transfer and their combination. Experimental results show that our method achieves comparable recognition performance to the state-of-the-art, while our method can process 390.7 frames per second, which is 27 times faster than the original two-stream method.

1. Introduction

Action recognition aims to enable computer automatically recognize human action in real world video. Recent years have witnessed extensive research efforts and significant research progresses in this area. The existence of large action datasets and worldwide competitions, like UCF101 [25], HMDB51 [15], and THUMOS14 [11] promote researches in this area. Early approaches in this area utilize a Bag-of-Visual-Words paradigm and its variants [20], which mainly consists of feature extraction, feature encoding, and classification steps. The performance of these approaches highly depends on the hand crafted features. Previous studies show that iDT descriptors and Fisher vector representation yield superior performance on various

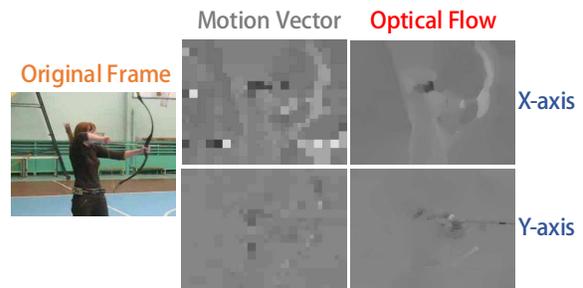


Figure 1. Comparison of motion vector and optical flow in x and y components. We can see that motion vector contains lots of noisy movement information and it is much coarser than optical flow.

datasets [28]. More recently researchers exploit deep learning for action recognition. One successful example along this line is the two-stream framework [23] which utilizes both RGB CNN and optical flow CNN for classification and achieves the state-of-the-art performance on several large action datasets. However, two-stream CNNs cannot process videos at real-time. The calculation of optical flow is time consuming which hinders the processing speed of two-stream CNNs.

This paper aims to develop a real-time action recognition method with high performance based on the successful two-stream framework. This is challenging, since optical flow itself is computationally expensive and cannot be estimated in real-time with most current algorithms [1, 4]. It takes 60 ms [1] to calculate optical flows per frame in K40 GPU, which is far from the requirement of real-time processing. To circumvent this difficulty, instead of using optical flow, this paper leverages motion vector as the input of CNN, which can be decoded directly from standard video compressed files with very low computational cost.

Motion vectors represent movement patterns of image blocks which resemble optical flows in terms of describing local motions. Early research [12] indicates that motion vectors include useful information for action recognition. However, the purpose of motion vector is not to unveil the temporal relationship of two macro blocks as accurate as possible, but to exploit temporal redundancy between ad-

*Corresponding author.

adjacent frames to reduce the bit rate in video compression. Thus motion vector only contains coarse movement patterns which are usually not precise. Moreover, motion vector lacks fine motion information of pixels. Directly replacing optical flows with motion vectors will severely degrade recognition performance of CNN as observed in our experiments.

Here our key insight to improve the recognition performance of motion vector CNN is that optical flow and motion vector share some similar characteristics which allows us to transfer the fine features/knowledge learned in optical flow CNN (OF-CNN) to that of motion vectors (MV-CNN). Both optical flow and motion vector can be extracted at each frame, and both of them contain motion information of local regions. Motion vector contains coarse and inaccurate motion information while optical flow carries fine and accurate ones. Due to the fine quality of optical flow fields, OF-CNN can learn elaborate filters and achieve better performance than MV-CNN. These facts inspire us to leverage the knowledge learned with OF-CNN to enhance MV-CNN. More specifically, we take OF-CNN learned in optical flow domain as a teacher net, and teach student MV-CNN in spirit of the knowledge distillation techniques proposed by [7]. We call this new CNN as optical flow enhanced motion vector CNN. Note we only require optical flow in the training phase, while the testing can be conducted with motion vector solely. Our experiments demonstrate that this novel strategy significantly boosts the performance of motion vectors, and achieves comparable performance with optical flow based action recognition methods [23].

The main contributions of this paper are summarized as below. Firstly, we propose a real-time CNN based action recognition method which achieves comparable performance with the state-of-the-art two-stream approach [23]. Secondly, we firstly introduce motion vector as the input of CNN to avoid the heavy computational cost of optical flow. Finally, we propose techniques to transfer the knowledge of optical flow CNN to motion vector CNN, which significantly improves the recognition performance.

2. Related Work

Action recognition has been widely studied in recent years. Early approaches extracted local spatio-time descriptors from input video and encoded these descriptors with Bag of Visual Words or its variants for classification. Laptev [17] proposed spatio-time interest points by extending Harris corner into spatio-time dimension. Wang *et al.* [28] further exploited trajectories to model temporal relationship of continuous frames. Furthermore, Kviatkovsky *et al.* [16] proposed a covariance descriptor to realize online action recognition. Popular local descriptors for video representation include HOG [2], HOF [17], MBH [3] and TBC [29]. And feature encoding techniques include hard quantization

[24], VLAD [9], and Fisher Vector [22]. [34, 30, 31] exploited mid-level representations by proposing MoFAP and Motionlets.

Recent renaissance of deep neural network remarkably accelerates the progresses in image classification. Convolutional Neural Networks (CNNs) [18] can learn powerful features from large scale image datasets, which greatly alleviates the difficulty of designing hand-crafted features. Extensive experiments have demonstrated that CNN can achieve superior performance on various image and video classification tasks, e.g. ImageNet object classification [14], face recognition [26], and event classification [5]. These successes inspire researchers to extend CNN for video classification tasks [13, 23, 27]. Karpathy *et al.* [13] proposed several convolutional neural network (CNN) architectures based on stacked RGB images for video classification. They designed several fusion strategy for RGB flows to utilize temporal information in stacked RGB frames. [10, 27] modeled temporal information by designing 3D filter to directly learn feature from videos. Unlike 2D filters used in image classification, 3D filters can learn temporal relationship from continuous RGB frames. Simonyan and Zisserman [23] proposed the two-stream architecture which exploits two CNNs to model RGB and optical flow respectively. This method achieved excellent performance in practice. We will use two-stream network as baseline of this study. Based on two-stream CNNs, Wang *et al.* [33] proposed Trajectory-pooled Deep-Convolutional Descriptors to obtains the merits of CNNs and trajectory based method. Wang *et al.* [35] further extended two-stream CNNs to very deep two-stream CNNs and achieve superior results on several datasets. Recent works also showed that the temporal structure in video contains discriminative information. Ng *et al.* [19] utilized the recurrent LSTM architecture to capture temporal structure of actions. Wu *et al.* [36] showed that integrating LSTM and two-stream methods can further improve the recognition performance. The main focus of this paper is to accelerate action recognition with deep learning while preserving the high performance.

3. Motion Vector for Deep Action Recognition

Although two-stream CNNs [23] achieve state-of-the-arts performance in action recognition, it is computational expensive and cannot be deployed for real-time process. Two-stream CNNs consist of spatial net and temporal net, which take RGB image and optical flow as input respectively. In the testing phase, feed forward computation of CNNs can be conducted in short time (around 30ms) with GPU implementation. The most computationally expensive step in the original two-stream framework comes from the calculation of optical flows. With efficient implementation [12], it takes around 360ms to calculate optical flow (Farneback's Flow [4]) for one frame in CPU. Even with

GPU acceleration [23], the calculation still takes 60 ms (Brox’s Flow [1]) which cannot meet the requirement of real-time process. Thus optical flow is the main bottleneck that prohibits the classical two-stream to be real-time.

Similar to optical flow, motion vector is a two-dimensional vector used for describing the moving offsets of local blocks with respect to a reference frame. Motion vector is widely used in various video compression standards, thus can be obtained directly in video decoding process without extra calculation. These facts make motion vector an attractive feature for efficient video content analysis. Actually, the previous work [12] has used motion vectors together with VLAD encoding for action recognition and achieve good performance. Different from this work [12], however, we explore motion vector in the deep CNN framework. Here, the major challenge comes from the fact that motion vector has low resolution and is imprecise for describing fine motions. This fact can largely harm the recognition performance of CNN if we directly train networks with motion vectors.

To relieve this problem, we propose several training methods to enhance motion vector CNN (MV-CNN) for better recognition performance. Our key insight is that the knowledge and features learned with optical flow CNN (OF-CNN) can be helpful for MV-CNN. Thus, we may leverage OF-CNN as a teacher net to guide the training of MV-CNN.

3.1. Motion Vector

We begin with a short introduction to motion vectors and then analyze the difficulty of training motion vector CNNs (MV-CNNs). Motion vectors are designed for describing macro blocks movement from one frame to the next, and are widely used in various video compression standards such as MPEG series, HEVC. Temporal redundancy of two neighboring frames yields important cues for compressing video data. Motion vectors exploit temporal relationship in neighboring frames by recording how macro blocks move in the next and are one of the essential ingredients in modern video coding algorithms. Figure 1 illustrates an example of motion vectors. As motion vectors are already calculated and encoded in compressed videos, we can obtain them at very low computational cost.

However, it is challenging to train MV-CNNs with high performance. This is because that motion vector is designed for video compression where precious motion information are not obligatory. Compared with optical flow, motion vectors exhibit coarse structure and may contain noisy and inaccurate movements. As shown in Figure 1, motion vector contains macro blocks with different sizes in motion estimation, ranging from 8×8 pixels to 16×16 pixels. Unlike dense optical flows, which are pixel-level and provide fine movement information of single pixel, motion vectors only

yield block-level motion information. Thus motion vectors exhibit much coarser structures than optical flows. Fine details contained in optical flows can not be delivered in motion vectors, such as the structure of bow in Archery videos and the baseball hold by the man’s hand. These information loss can badly impact the final performance of MV-CNN.

Secondly, noisy information contained in motion vector poses a barrier for MV-CNN to achieve high performance. Unlike modern optical flow algorithm [4, 1], motion vectors simply use three or four comparison steps to find the most matching block. There are much more noisy patterns contained in motion vectors than optical flow fields as shown in Figure 1. The inaccurate information and existence of noisy point are due to the fact that video compression algorithms need to balance between the speed of encoding and the compression rate. Therefore, motion vectors can only provide noisy block movement information, which hamper performance of temporal CNN.

Thirdly, not every frame contains motion vectors. Partial reason for this is that frames are clustered as group of pictures (GOP). One typical GOP contains three types of frames: I-frame, P-frame and B-frame. I-frame is an intra-coded frame encoded based on its own, which means I-frame contains no movement information. P-frame and B-frame are acronyms of predicted frame and bi-predictive frame respectively. They contain movement information. Clearly, in action recognition with CNNs, empty I-frame can hinder CNN training process and degrade the performance. In this paper, we deal with this problem by simply replacing I-frame with the motion vectors of previous frame.

3.2. Real-time Action Recognition Frameworks

Our proposed real-time action recognition framework contains two components (Figure 2). The first component is video decoder, which extracts RGB images and motion vectors from input compressed videos. The second component follows the two-stream architecture [23], which can be decomposed into spatial CNN (RGB-CNN) and temporal CNN (MV-CNN). The main difference comes from the fact that we use motion vector as input for temporal CNN while two-stream uses optical flows. As both RGB images and motion vectors can be obtained directly from the video decoding process, our method avoids the computationally expensive step to estimate optical flow, which is most time-costly in the original two-stream framework.

In the training phase, we extract RGB images and frames of motion vectors from video. These images and motion vectors inherit the labels of original videos. To augment training samples, we crop image and MV frames in spatial domain. Then we train RGB-CNN and MV-CNN with these cropped samples, respectively. In the testing phase, raw images and motion vector frames from testing video

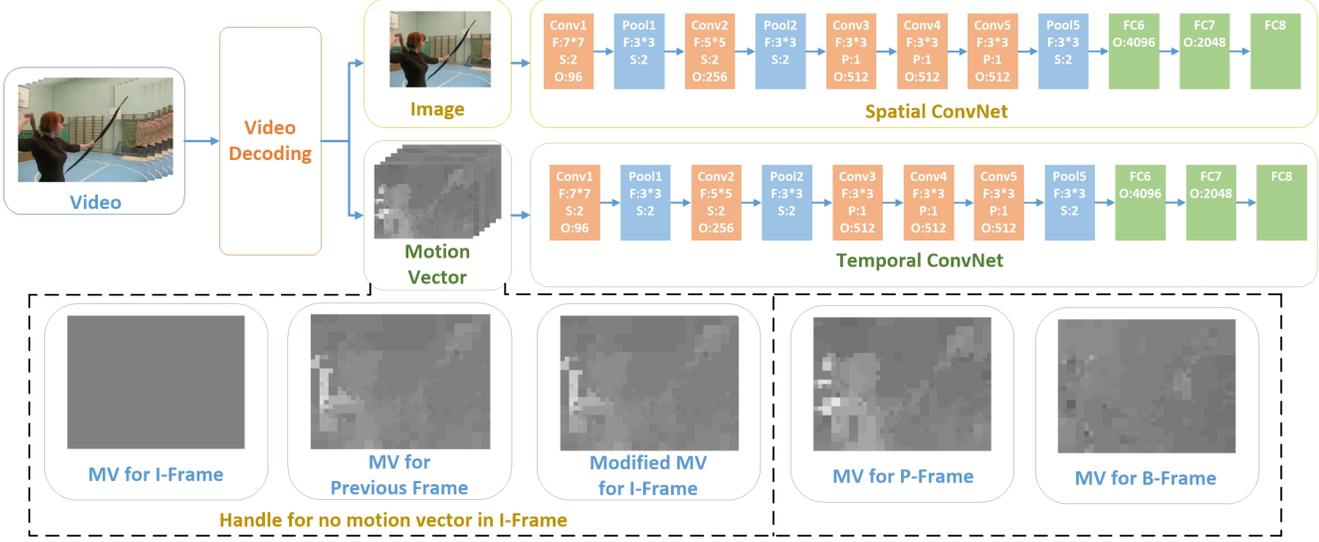


Figure 2. Structure for real-time action recognition system. In spatial and temporal CNN, F stands for kernel size and S means stride step. O represents for output number and P is pad size.

are fed forward into RGB-CNN and MV-CNN. The action recognition decision will be made by weighted average of prediction scores from two CNNs. The weight for spatial and temporal CNN are set as 1 and 2.

We use the architecture of ClarifaiNet [37] for both RGB-CNN and MV-CNN, since ClarifaiNet keeps a balance between efficiency and performance. For spatial net, dropout layers are added after FC6 and FC7 layer. We follow [23] by setting dropout ratio for spatial CNN to 0.9 to avoid over-fitting. Spatial net is pre-trained on ImageNet ILSVRC-2012 datasets. Batch size for spatial and temporal nets are set as 256. The learning rate is initialized as 10^{-3} and decreases into 10^{-4} after 14k steps. The training procedure stops at 20k iterations.

Our temporal CNN is slightly different with ClarifaiNet by replacing all ReLU layers with PReLU [6] layers, which exhibits better performance and quick convergence. Dropout ratio for temporal net is set to 0.9 and 0.8 after FC6 and FC7 layer respectively. Temporal CNN is trained from scratch by stacking 10-frame motion vectors as input. Its learning rate starts from 10^{-2} , and drops to 10^{-3} at 30k steps. We then decrease the learning rate to 10^{-4} after 70k iterations. The whole learning process stops at 90k steps.

4. Enhanced Motion Vector CNNs

As analyzed above, motion vectors lack fine details and contain noisy and inaccurate motion patterns, which makes training motion vector CNN (MV-CNN) more challenging. We observe that simply replacing optical flow with motion vector can lead to significant recognition performance degradation of around 7%. In this section we try to keep the high speed merit of motion vectors while achieve the

high performance as optical flow. Here our key insight to circumvent this difficulty is that motion vector and optical flow are inherent correlated to each other. This fact enables us to leverage the rich knowledge and fine features learned in optical flow CNN (OF-CNN) to enhance MV-CNN. This can be seen as a knowledge transfer problem from optical flow domain to motion vector domain. More specifically, we take OF-CNN as a teacher net and use it to teach the MV-CNN net. It should be noticed that OF-CNN teaches MV-CNN only in the training phase. Thus we do not need to calculate optical flows in testing phase, and the system speed performance will not be affected by the proposed algorithm.

In particular, we propose three strategies to transfer knowledge from OF-CNN to MV-CNN. To begin with, several notations are introduced at first. Parameters for teacher CNN in optical flow domain is denoted by $T_p = \{T_p^1, T_p^2, \dots, T_p^n\}$, where n represents the total number of layers. As for student CNN (MV-CNN), its parameter is defined as $S_p = \{S_p^1, S_p^2, \dots, S_p^n\}$. In this paper, we assume MV-CNN has the same network structure as OF-CNN, while the techniques can be easily generalized to those with different structures.

4.1. Teacher Initialization

Extensive works show that the initialization of network parameters can largely impact the final performance. Both optical flow and motion vector describe the motion information of local regions, and are inherently correlated. This fact inspires us to initialize the parameters of MV-CNN as those of its teacher's net OF-CNN,

$$S_p^t = T_p^t, t = 1, \dots, n. \quad (1)$$

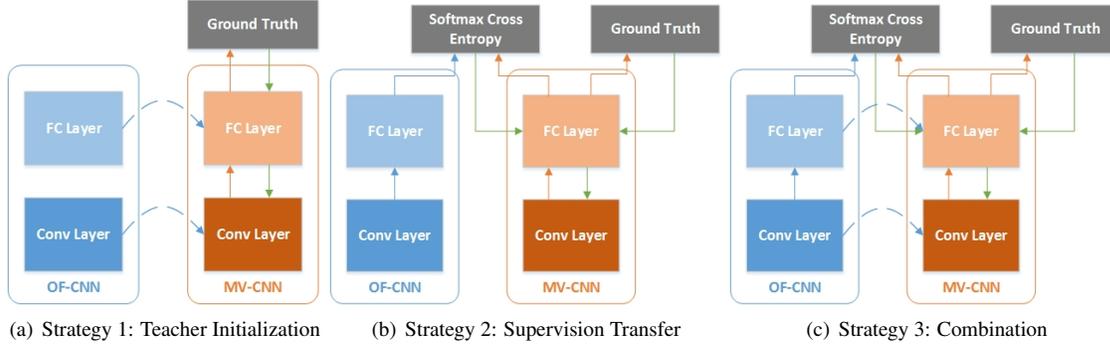


Figure 3. Structure of three strategies. Blue dash lines represent copying the initial weights from teacher net to student net. Green lines are the backward propagation path. Blue full lines mean feed forward paths of teacher flow. Orange lines are feed forward paths of student net.

Then we fine-tune the parameters of MV-CNN with the motion vector samples until convergence. This process can also be seen as pre-training MV-CNN with the fine optical flow samples, which transfers the knowledge learned by teacher net to student net directly.

For implementation details, the learning rate is initiated from 10^{-3} , and then drops to 10^{-4} and 10^{-5} at 30k and 70k steps respectively. The training stops at 90k iterations.

4.2. Supervision Transfer

Initialization of MV-CNN with OF-CNN’s parameters is simple, but the initial knowledge transferred to MV-CNN may be lost during the fine-tuning process with motion vector samples. To relieve this problem, we introduce the supervision transferring approach which takes account of additional supervision from the teacher net in the training process of MV-CNN. OF-CNN can extract effective representation from input frame. Here we take the representation obtained in the FC layer of OF-CNN as a new supervision for training MV-CNN.

This technique is in spirit similar to Hinton *et al.*’s recent work on knowledge distillation [7]. The aim of knowledge distillation is to compress a cumbersome network (teacher) into a small network (student) which achieves similar performance as the cumbersome one. The cumbersome network and small network have the same input. Different from this work, in our problems, the teacher (OF-CNN) and the student (MV-CNN) take different types of input, *i.e.* optical flow vs. motion vector, but have the same network structure. Moreover, our objective is not to compress the student to a small net, but to enhance the performance of the student with low quality input.

Formally, for a given frame I with optical flow feature o and motion vector v , we calculate the output of the last FC layer of teacher CNN as: $T^n(o) = \text{softmax}(T^{n-1}(o))$, where ‘softmax’ function is used to transform the feature T^{n-1} to a probability score of multiple classes. Similarly, the output of student’s last layer is defined as: $S^n(v) =$

$\text{softmax}(S^{n-1}(v))$.

To transfer knowledge from teacher to student, we hope $S^n(v)$ can approximate $T^n(o)$ as closely as possible. We introduce a *teacher supervision loss* function to minimize the difference between $S^n(v)$ and $T^n(o)$. The cross-entropy loss is used to measure the difference. Following [7], we introduce a temperature $Temp$ to soften the next-to-last layer output. The softmax output of teacher net is softened as $P_T = \text{softmax}(T^{n-1}/Temp)$, Student net’s softmax output for second target is similarly defined as: $P_S = \text{softmax}(S^{n-1}/Temp)$, Then the teacher supervision loss (TSL) with cross-entropy function is defined by :

$$L_{TSL} = - \sum_{i=1}^k P_T(i) \log P_S(i), \quad (2)$$

where k is the dimension of the student’s output (same as that of the teacher).

In addition to the teacher supervision loss, we also minimize the cross entropy between student’s output and the ground truth Q , which is given by,

$$L_{GT} = - \sum_i \mathbb{1}[Q = i] \log S^n(i), \quad (3)$$

where S^n and Q represent the hard output distribution vectors and the ground truth label respectively.

Our final loss function combines the teacher supervision loss (Eq.2) and the ground truth loss (Eq.3):

$$L = L_{TSL} + w \times L_{GT} \quad (4)$$

where w is a weight to balance these two terms. Thus, the student MV-CNN can receive supervision signal from both the teacher OF-CNN and the ground truth. It should be noted that in supervision transfer, the weights of teacher model are frozen. In this way, the knowledge is transferred from the teacher net to the student one.

For implementation details, we set the initial learning rate as 10^{-3} . We decay the learning rate to 10^{-4} at 50k and 10^{-5} at 70k steps. The whole training procedure stops at 90k iterations.

Temporal CNN	Accuracy
OF-CNN [23]	81.2%
MV-CNN trained from scratch	74.4%
EMV-CNN with ST	77.5%
EMV-CNN with TI	78.2%
EMV-CNN with ST+TI	79.3%

Table 1. Comparison of temporal CNN performance for Optical Flow based approach and Motion vector based Method on UCF101 (Split1). ST stands for Supervision Transfer and TI means Teacher Initialization.

4.3. Combination

In the third strategy, we combine the initialization with teacher’s parameters and supervision transfer to further enhance the performance of student’s net. We first train the teacher OF-CNN with optical flow fields. Then we initialize the student MV-CNN with OF-CNN’s parameters. After this, we train the student net with supervision signal from both the teacher and the ground truth (Eq. 4). In this way, the student net not only inherits teacher’s parameters from initialization, but also mimics teacher net’s prediction during fine tuning procedure. This allows us to combine the merits of two previous strategies and further boost the generalization ability of student MV-CNN.

5. Experiment

In this section we first describe the evaluation datasets. Then we report and analyze the experimental results.

5.1. Datasets and Evaluation Protocol

The evaluation is conducted on two challenging datasets: UCF101 [25] and THUMOS14 [11]. UCF101 contains 13,320 videos. The datasets has three splits for training and testing. We follow the standard setup and report average accuracy over three splits on this datasets.

THUMOS14 is a dataset for action recognition challenge 2014. It contains 13,320 videos for training, 1,010 videos for validation, and 1,574 videos for testing. Unlike UCF101, THUMOS14 uses untrimmed videos for validation and testing. Lots of irrelevant frames make the training and testing of CNNs more difficult. We use training and validation datasets to train our CNNs. Official evaluation tool is utilized to evaluate our system performance. According to the standard setup of this dataset, we report the mean Average Precision (mAP) on testing dataset. As videos in THUMOS14 are untrimmed and have large number of frames, we conduct CNN testing at every 20 frames.

For experiments on both datasets, the speed evaluation is measured as frames per seconds (fps) on a single-core CPU (E5-2640 v3) and a K40 GPU.

CNN	MAP
RGB CNN	57.7%
OF-CNN	55.3%
RGB CNN+OF-CNN	66.1%
MV-CNN	29.8%
EMV-CNN	41.6%
RGB CNN+MV-CNN	58.7%
RGB CNN+EMV-CNN	61.5%

Table 2. Performance of EMV-CNNs and MV-CNNs on THUMOS 14 dataset. We also report the results of two-stream CNNs.

Dataset	MV (fps)	CNN (fps)	Total (fps)
UCF101	735.3	833.3	390.7
THUMOS14	781.3	833.3	403.2

Table 3. Speed of each components in Real-time Action Recognition System. MV stands for motion vector extraction, while CNN means convolutional neural network processing.

Dataset	Spatial Resolution	Brox’s Flow[1] (GPU) (fps)	MV (CPU) (fps)
UCF101	320×240	16.7	735.3
THUMOS14	320×180	17.5	781.3

Table 4. Comparison of speed for optical flow fields and motion vectors. MV means motion vector.

5.2. Implementation Details

In order to learn robust features from CNNs, we use three data augmentation strategies. Firstly, we randomly crop a 224×224 patch from image set. Random cropping can provide more training data for CNNs to learn better features. Secondly, we horizontally flip cropped patches by random. Furthermore, following [35]¹, we use a scale jittering strategy to help CNN to learn robust features. We crop a patch from dataset on three scales 1, 0.875, and 0.75, which yield patches of size 224×224 , 196×196 and 168×168 respectively. The patches are then resized to 224×224 after this multi-scale strategy. In testing phase, we crop one 224×224 patch from the center of testing image. No data augmentation strategy is used in evaluation phase.

Our teacher CNN is trained on TV-L1 optical flow [21] that achieves 81.6% on UCF101 Split1, which is comparable with the performance in the original paper 81.2% [23].

5.3. Parameter Sensitivity

We first analyze the parameter sensitivity. There are two important parameters in our real-time action recognition system: temperature $Temp$ for Supervision Transfer

¹<https://github.com/yjxiong/caffe>

and weight w for soft target. Following [7] we set soft target weight $w = Temp^2$ to balance the gradients of soft target. We test three different temperature settings on UCF101 split1 by setting $Temp$ to 1, 2 and 3. The corresponding soft target weight w is set as 1, 4 and 9. Accuracy grows up from 79.2% to 79.3% from temperature 1 to 2 and goes down to 78.9% at temperature 3. We can see that the recognition results of different temperatures are very close, which implies that our method is robust to temperature. According to this study, we set $Temp = 2$ in the following experiments.

5.4. Evaluation of MV-CNNs and EMV-CNNs

In this subsection, we compare and analyze different training strategies for motion vector CNN (MV-CNN) on UCF101 Split1 and THUMOS14. The result are shown in Table 1 and Table 2. As [23] did not provide results on THUMOS14, we re-implement two-stream CNNs on this dataset.

First, from these results, we can conclude that directly using motion vector to replace optical flow degrades system’s performance. Compared with OF-CNN, MV-CNN trained from scratch degrades performance by 7% and 25%, which achieves 74.4% and 29.8% on UCF101 Split1 and THUMOS14 respectively. It indicates that coarse structure and imprecise movement information in motion vector harm the performance of CNNs. The performance on THUMOS14 is particularly large, as validation and testing videos in THUMOS14 are untrimmed and labels for a large part of frames are not correct. Furthermore, lots of shots shift in videos aggravate the difficulties of training MV-CNN on THUMOS14 dataset.

Second, comparing Enhanced Motion Vector CNN (EMV-CNN) and MV-CNN, we observe a performance improvement of 4.9% and 12% on UCF101 Split1 and THUMOS14 respectively. This indicates that our newly designed method for training EMV-CNN is effective and can improve the generalization ability of MV-CNNs.

Finally, combined with spatial CNN, EMV-CNN still outperforms MV-based CNN, which indicates that the knowledge of EMV-CNN can be more complementary to spatial CNN. In addition, combining EMV-based with spatial CNN exhibits only a minor performance loss compared with OF-based CNN on UCF101. Specifically, we study the effect of our proposed transferring techniques as follows.

Teacher initialization technique can provide an improvement of 3.8%. From this result, we can see, similar to the study in image classification[?], teacher initialization on OF-CNN can benefit the MV-CNN by providing a good initial point to train. Supervision transfer strategy can provide 3.1% performance improvement on UCF101 Split1. It shows that knowledge provided by OF-CNN during supervision transfer process in training phase can be helpful to MV-CNN. Combing supervision transfer and teacher ini-

EMV+RGB-CNN	UCF101(Split1) (fps)	THUMOS14 (fps)
5 crops+mirror	88.0	89.0
1 crop	390.7	403.2

Table 5. Speed comparison for 1 crop and 5 crops+mirror in UCF101 and THUMOS14. EMV+RGB-CNN stands for two-stream based CNN with EMV-CNN and RGB-CNN.

EMV+RGB-CNN	UCF101(Split1)	THUMOS14
5 crops+mirror	86.6%	61.2%
1 crop	85.7%	61.5%

Table 6. Comparison of performance for 1 crop and 5 crops+mirror in UCF101 and THUMOS14. EMV+RGB-CNN stands for two-stream based CNN with EMV-CNN and RGB-CNN.

	Accuracy	FPS
MV+FV (CPU) (re-implement) [12]	78.5%	132.8
C3D (1 net) (GPU) [27]	82.3%	313.9
C3D (3 net) (GPU) [27]	85.2%	-
iDT+FV (CPU) [28]	85.9%	2.1
Two-stream CNNs (GPU) [23]	88.0%	14.3
EMV+RGB-CNN	86.4%	390.7

Table 7. Comparison of speed and performance with state-of-the-art on UCF101

tialization strategy can further boost the performance. As indicated in [7], supervision transfer can be used to regularize MV-CNN from over-fitting. Supervision strategy utilizes the soft codes produced by OF-CNNs during training process. These codes are further softened by dividing $Temp$. Unlike hard ground truth labels, these soft targets encode more rich information and guide the training of EMV-CNN, contributing to improve the generalization ability of EMV-CNNs.

5.5. Speed Evaluation

In this subsection, we analyze the speed of different components in our action recognition approach. In our implementation, We use CPU to extract motion vector while use GPU to conduct the feed forward process of CNN. It should be noticed that our system process a volume of 10 frames of MV and 1 frame of RGB together. Our speed is measured based on frames instead of volume.

We first compare the speed performance of 1 crop with 5 crops+mirror on UCF101 and THUMOS14 in Table 5 and Table 6. We can observe that 1 crop only achieves a slight performance degradation than 5 crops+mirror, while 1 crop is 4 times faster than the latter.

Next, we evaluate speed performance of each component on UCF101 datasets and THUMOS14 datasets. The spatial resolution of video in UCF101 dataset is 320×240 , while

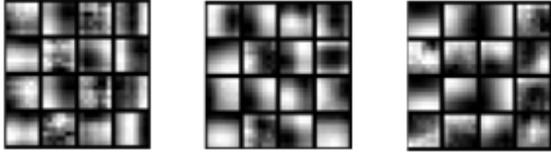


Figure 4. Samples of filters for Conv1 layer. Left to right: MV-CNN, EMV-CNN and OF-CNN.

for video in THUMOS14, its resolution is 320×180 . As shown in Table 3, our action recognition system achieves 390.7 fps and 403.2 fps on UCF101 and THUMOS14 respectively, which is one order faster than real-time requirement (25 fps).

Finally, we compare the efficiency of extracting motion vectors and optical flow in Table 4. Motion vector extraction is almost 30 times faster than real-time requirement. Although the spatial resolutions for videos in UCF101 and THUMOS14 are different, the time consumption for extraction of motion vectors are similar. The estimation of motion vector in CPU is 44 times faster than calculating Brox’s flow[1] with GPU. The calculation of optical flows poses the main bottleneck in boost the speed of classical two-stream framework, which prohibits it to be conducted in real-time.

5.6. Comparison with the State of the Art

In this section, we compare our method with several state-of-the-art methods. Since the I/O is related to hardware and operating system, we report computational time cost without I/O. Unlike those using Support Vector Machine (SVM) to perform classification [27, 28], our action recognition approach has an *end-to-end structure* that combines feature extraction and classification together.

We first compare speed and accuracy performance on UCF101 (3 Splits) dataset. Results are given in Table 7. Our method is 3 times and 180 times faster than motion vector + fisher vector (MV FV) and iDT+FV respectively. It should be noted that this may not be a fair comparison since iDT and FV are implemented in CPU. At the same time, our method achieves higher accuracy than these methods. For MV FV, we use the public code offered by [12]. Although iDT used Farneback optical flows [4] which provides more precise movement information than motion vector, our methods still obtain higher performance than iDT.

We also make comparison with deep learning methods, namely two-stream CNNs and C3D. Our method achieves **390.7** fps on UCF101 and is the fastest among all methods compared (Table 7). And our recognition accuracy is 4.1% and 1.2% higher than C3D (1 net) and C3D (3 net). As for two-stream CNNs, our method achieves comparable results but ours is 27 times faster than them.

Finally, we compare our speed and mean average precision with [8] on THUMOS14 dataset (Table 8). Our

	Accuracy	FPS
Objects (GPU) [8]	44.7%	-
iDT+CNN (CPU+GPU) [32]	62.0%	< 2.38
Motion (iDT+FV) (CPU) [8]	63.1%	2.38
Objects+Motion (CPU+GPU) [8]	71.6%	< 2.38
EMV+RGB-CNN	61.5%	403.2

Table 8. Comparison of speed and performance with state-of-the-art on THUMOS14

method achieves better performance than Objects for 16.8% and obtains comparable performance with Motion (iDT) and iDT+CNN, but exhibits worse performance than Objects+Motion. However, as Objects+Motion is built on the iDT features, our method is 200 times faster than it.

5.7. Visualization of Filters

In order to further explore the effectiveness of EMV-CNN, we visualize filters of the first layer (Conv1) for MV-CNN, EMV-CNN and OF-CNN in Figure 4. It can be clearly noticed that filters of MV-CNN contain coarse and noisy information, which may be ascribed to the fact that motion vector lacks fine grained information as shown in Figure 1. Compared with filters of MV-CNN, EMV-CNN can obtain detailed information, which shows that our proposed learning scheme successfully transfer knowledge from OF-CNN to EMV-CNN. Filters of EMV-CNN are cleaner than MV-CNN and have fine structures, contributing to improve recognition performance.

6. Conclusions

In this paper we have proposed a motion vector CNN to accelerate the speed of deep learning methods for action recognition. Motion vectors can be extracted directly in video decoding process without extra computation. However, motion vectors lacks fine and accurate motion information which degrades recognition performance. To relieve this problem, we developed three knowledge transfer techniques to adapt the models of optical flow CNN to motion vector CNN, which significantly boost the recognition performance of the latter. Our method achieves 391 fps and 403 fps with high performance on UCF101 and THUMOS14 respectively.

7. Acknowledgement

This work is partly supported by National Natural Science Foundation of China (61472410, 61472281), Guangdong Innovative Research Program (2015B010129013, 2014B050505017), Shenzhen Research Program (KQCX2015033117354153, JSGG20150925164740726, CXZZ20150930104115529) and the Program for Professor of Special Appointment (Eastern Scholar) at the Shanghai Institutions of Higher Learning (GZ2015005).

References

- [1] T. Brox, A. Bruhn, N. Papenberger, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV'14*, pages 25–36, 2004.
- [2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR'05*, pages 886–893, 2005.
- [3] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV'06*, pages 428–441, 2006.
- [4] G. Farneback. Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, pages 363–370, 2003.
- [5] 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'15*, pages 2568–2577, 2015.
- [6] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.
- [7] G. E. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.
- [8] M. Jain, J. C. van Gemert, and C. G. Snoek. What do 15,000 object categories tell us about classifying and localizing actions? In *CVPR'15*, pages 46–55, 2015.
- [9] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR'10*, pages 3304–3311, 2010.
- [10] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *PAMI*, 35(1):221–231, 2013.
- [11] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. <http://csrcv.ucf.edu/THUMOS14/>, 2014.
- [12] V. Kantorov and I. Laptev. Efficient feature extraction, encoding, and classification for action recognition. In *CVPR'14*, pages 2593–2600, 2014.
- [13] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR'14*, pages 1725–1732, 2014.
- [14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS'12*, pages 1097–1105, 2012.
- [15] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *CVPR'11*, pages 2556–2563, 2011.
- [16] I. Kviatkovsky, E. Rivlin, and I. Shimshoni. Online action recognition using covariance of shape and motion. *CVIU*, 129:15–26, 2014.
- [17] I. Laptev. On space-time interest points. *IJCV*, 64(2-3):107–123, 2005.
- [18] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10), 1995.
- [19] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snipets: Deep networks for video classification. In *CVPR'15*, pages 4694–4702, 2015.
- [20] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *CoRR*, abs/1405.4506, 2014.
- [21] J. S. Pérez, E. Meinhardt-Llopis, and G. Facciolo. TV-L1 optical flow estimation. *IPOL Journal*, 3:137–150, 2013.
- [22] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *IJCV*, 105(3):222–245, 2013.
- [23] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS'14*, pages 568–576, 2014.
- [24] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV'13*, pages 1470–1477, 2003.
- [25] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012.
- [26] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR'14*, pages 1701–1708, 2014.
- [27] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: generic features for video analysis. *CoRR*, abs/1412.0767, 2014.
- [28] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV'13*, pages 3551–3558, 2013.
- [29] H. Wang, Y. Yi, and J. Wu. Human action recognition with trajectory based covariance descriptor in unconstrained videos. In *ACM MM'15*, pages 1175–1178, 2015.
- [30] L. Wang, Y. Qiao, and X. Tang. Mining motion atoms and phrases for complex action recognition. In *ICCV'13*, pages 2680–2687, 2013.
- [31] L. Wang, Y. Qiao, and X. Tang. Motionlets: Mid-level 3D parts for human motion recognition. In *CVPR'13*, pages 2674–2681, 2013.
- [32] L. Wang, Y. Qiao, and X. Tang. Action recognition and detection by combining motion and appearance features. In *THUMOS Action Recognition challenge*, 2014.
- [33] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR'15*, pages 4305–4314, 2015.
- [34] L. Wang, Y. Qiao, and X. Tang. MoFAP: A multi-level representation for action recognition. *IJCV*, 2015.
- [35] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards Good Practices for Very Deep Two-Stream ConvNets. *CoRR*, abs/1507.02159, July 2015.
- [36] Z. Wu, Y.-G. Jiang, J. Wang, J. Pu, and X. Xue. Exploring inter-feature and inter-class relationships with deep neural networks for video classification. In *ACM MM'14*, pages 167–176, 2014.
- [37] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV'14*, pages 818–833, 2014.