MotionSqueeze: Neural Motion Feature Learning for Video Understanding

Heeseung Kwon[1,2] Manjin Kim[1] Suha Kwak[1] Minsu Cho[1,2]

[1]POSTECH* [2]NPRC†

http://cvlab.postech.ac.kr/research/MotionSqueeze/

Abstract. Motion plays a crucial role in understanding videos and most state-of-the-art neural models for video classification incorporate motion information typically using optical flows extracted by a separate off-the-shelf method. As the frame-by-frame optical flows require heavy computation, incorporating motion information has remained a major computational bottleneck for video understanding. In this work, we replace external and heavy computation of optical flows with internal and light-weight learning of motion features. We propose a trainable neural module, dubbed *MotionSqueeze*, for effective motion feature extraction. Inserted in the middle of any neural network, it learns to establish correspondences across frames and convert them into motion features, which are readily fed to the next downstream layer for better prediction. We demonstrate that the proposed method provides a significant gain on four standard benchmarks for action recognition with only a small amount of additional cost, outperforming the state of the art on Something-Something-V1&V2 datasets.

Keywords: video understanding, action recognition, motion feature learning, efficient video processing.

1 Introduction

The most distinctive feature of videos, from those of images, is motion. In order to grasp a full understanding of a video, we need to analyze its motion patterns as well as the appearance of objects and scenes in the video [20,27,31,38]. With significant progress of neural networks on the image domain, convolutional neural networks (CNNs) have been widely used to learn appearance features from video frames [5,31,35,40] and recently extended to learn temporal features using spatio-temporal convolution across multiple frames [2,35]. The results, however, have shown that spatio-temporal convolution alone is not enough for learning motion patterns; convolution is effective in capturing translation-equivariant patterns but not in modeling relative movement of objects [39,46]. As a result, most

*Pohang University of Science and Technology, Pohang, Korea

†The Neural Processing Research Center, Seoul, Korea

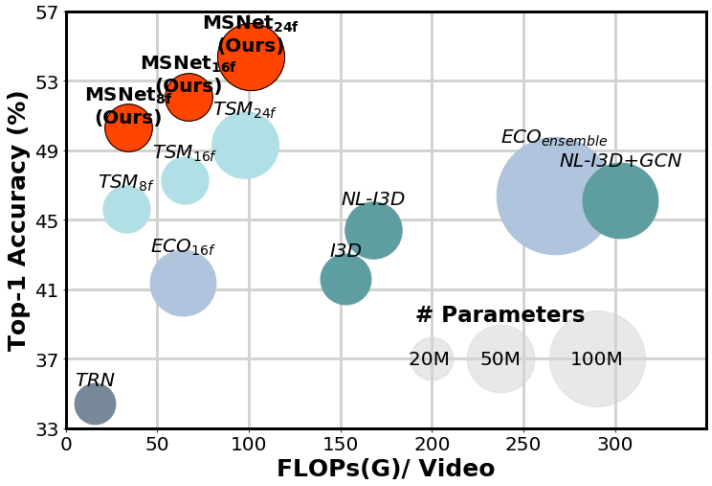


Fig. 1: Video classification performance comparison on Something-Something-V1 [10] in terms of accuracy, computational cost, and model size. The proposed method (MSNet) achieves the best trade-off between accuracy and efficiency compared to state-of-the-art methods of TSM [21], TRN [47], ECO [48], I3D [2], NL-I3D [41], and GCN [42]. Best viewed in color.

state-of-the-art methods still incorporate explicit motion features, *i.e.,* dense optical flows, extracted by an external off-the-shelf methods [2, 21, 31, 37, 43]. This causes a major computational bottleneck in video-processing models for two reasons. First, calculating optical flows frame-by-frame is a time-consuming process; obtaining optical flows of a video is typically an order of magnitude slower than feed-forwarding the video through a deep neural network. Second, processing optical flows often requires a separate stream in the model to learn motion representations [31], which results in doubling the number of parameters and the computational cost. To address these issues, several methods have attempted to internalize motion modeling [7, 20, 27, 34]. They, however, all either impose a heavy computation on their architectures [7, 27] or underperform other methods using external optical flows [20, 34].

To tackle the limitation of the existing methods, we propose an end-to-end trainable block, dubbed the *MotionSqueeze* (MS) module, for effective motion estimation. Inserted in the middle of any neural network for video understanding, it learns to establish correspondences across adjacent frames efficiently and convert them into effective motion features. The resultant motion features are readily fed to the next downstream layer and used for final prediction. To validate the proposed MS module, we develop a video classification architecture, dubbed the MotionSqueeze network (MSNet), that is equipped with the MS module. In comparison with recent methods, shown in Figure 1, the proposed method provides the best trade-off in terms of accuracy, computational cost, and model size in video understanding.

2 Related work

Video classification architectures. One of the main problems in video understanding is to categorize videos given a set of pre-defined target classes. Early methods based on deep neural networks have focused on learning spatio-temporal or motion features. Tran *et al.* [35] propose a 3D CNN (C3D) to learn spatio-temporal features while Simonyan and Zisserman [31] employ an independent temporal stream to learn motion features from precomputed optical flows. Carreira and Zisserman [2] design two-stream 3D CNNs (two-stream I3D) by integrating two former methods, and achieve the state-of-the-art performance at that time. As the two-stream 3D CNNs are powerful but computationally demanding, subsequent work has attempted to improve the efficiency. Tran *et al.* [37] and Xie *et al.* [43] propose to decompose 3D convolutional filters into 2D spatial and 1D temporal filters. Chen *et al.* [3] adopt group convolution techniques while Zolfaghari *et al.* [48] propose to study mixed 2D and 3D networks with the frame sampling method of temporal segment networks (TSN) [40]. Tran *et al.* [36] analyze the effect of 3D group convolutional networks and propose the channel-separated convolutional network (CSN). Lin *et al.* [21] propose the temporal shift module (TSM) that simulates 3D convolution using 2D convolution with a part of input feature channels shifted along the temporal axis. It enables 2D convolution networks to achieve a comparable classification accuracy to 3D CNNs. Unlike these approaches, we focus on efficient learning of motion features.

Learning motions in a video. While two-stream-based architectures [8,9,30,31] have demonstrated the effectiveness of pre-computed optical flows, the use of optical flows typically degrades the efficiency of video processing. To address the issue, Ng *et al.* [26] use a multi-task learning of both optical flow estimation and action classification and Stroud *et al.* [32] propose to distill motion features from pre-trained two-stream 3D CNNs. These methods do not use pre-computed optical flows during inference, but still need them at the training phase. Other methods design network architectures that learn motions internally without external optical flows [7,16,20,27,34]. Sun *et al.* [34] compute spatial and temporal gradients between appearance features to learn motion features. Lee *et al.* [20] and Jiang *et al.* [16] propose a convolutional module to extract motion features by spatial shift and subtraction operation between appearance features. Despite their computational efficiency, they do not reach the classification accuracy of two-stream networks [31]. Fan *et al.* [7] implement the optimization process of TV-L1 [44] as iterative neural layers, and design an end-to-end trainable architecture (TVNet). Piergiovanni and Ryoo [27] extend the idea of TVNet by calculating channel-wise flows of feature maps at the intermediate layers of the CNN. These variational methods achieve a good performance, but require a high computational cost due to iterative neural layers. In contrast, our method learns to extract effective motion features with a marginal increase of computation.

Learning visual correspondences. Our work is inspired by recent methods that learn visual correspondences between images using neural networks [6,11,19,24,28,33]. Fischer *et al.* [6] estimate optical flows using a convolutional neural

network, which construct a correlation tensor from feature maps and regresses displacements from it. Sun *et al.* [33] use a stack of correlation layers for coarse-to-fine optical flow estimation. While these methods require dense ground-truth optical flows in training, the structure of correlation computation and subsequent displacement estimation is widely adopted in other correspondence problems with different levels of supervision. For example, recent methods for semantic correspondence, *i.e.*, matching images with intra-class variation, typically follow a similar pipeline to learn geometric transformation between images in a more weakly-supervised regime [11, 19, 24, 28]. In this work, motivated by this line of research, we develop a motion feature module that does not require any correspondence supervision for learning.

Similarly to our work, a few recent methods [22, 45] have attempted to incorporate correspondence information for video understanding. Zhao *et al.* [45] use correlation information between feature maps of consecutive frames to replace optical flows. The size of their full model, however, is comparable to the two-stream networks [31]. Liu *et al.* [22] propose the correspondences proposal (CP) module to learn correspondences in a video. Unlike ours, they focus on analyzing spatio-temporal relationship within the whole video, rather than motion, and the model is not fully differentiable and thus less effective in learning. In contrast, we introduce a fully-differentiable motion feature module that can be inserted in the middle of any neural network for video understanding.

The main contribution of this work is three-fold.

- We propose an end-to-end trainable, model-agnostic, and lightweight module for motion feature extraction.
- We develop an efficient video recognition architecture that is equipped with the proposed motion module.
- We demonstrate the effectiveness of our method on four different benchmark datasets and achieve the state-of-the-art on Something-Something-V1&V2.

3 Proposed approach

The overall architecture for video understanding is illustrated in Figure 2. Let us assume a neural network that takes a video of T frames as input and predicts the category of the video as output, where convolutional layers are used to transform input frames into frame-wise appearance features. The proposed motion feature module, dubbed *MotionSqueeze (MS) module*, is inserted to produce frame-wise motion features using pairs of adjacent appearance features. The resultant motion features are added to the appearance features for final prediction. In this section, we first explain the MS module, and describe the details of our network architecture for video understanding.

3.1 MotionSqueeze (MS) module

The MS module is a learnable motion feature extractor, which can replace the use of explicit optical flows for video understanding. As described in Figure 3,

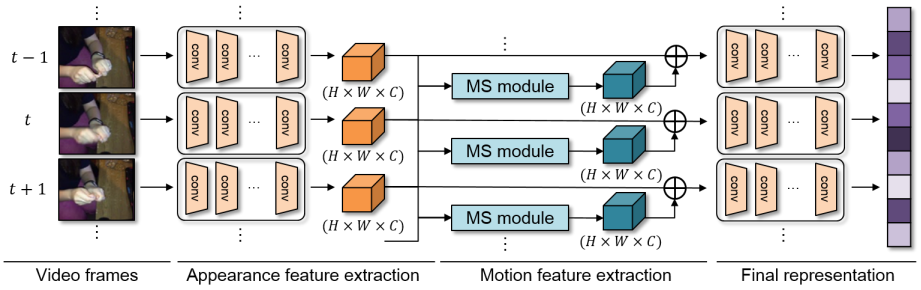


Fig. 2: Overall architecture of the proposed approach. The model first takes T video frames as input and converts them into frame-wise appearance features using convolutional layers. The proposed *MotionSqueeze (MS) module* generates motion features using the frame-wise appearance features, and combines the motion features into the next downstream layer. $\oplus$ denotes element-wise addition.

given two feature maps from adjacent frames, it learns to extract effective motion features in three steps: correlation computation, displacement estimation, and feature transformation.

Correlation computation. Let us denote the two adjacent input feature maps by $\mathbf{F}^{(t)}$ and $\mathbf{F}^{(t+1)}$, each of which is 3D tensors of size $H \times W \times C$. The spatial resolution is $H \times W$ and the C dimensional features on spatial position $\mathbf{x}$ by $\mathbf{F}_{\mathbf{x}}$. A correlation score of position $\mathbf{x}$ with respect to displacement $\mathbf{p}$ is defined as

$$s(\mathbf{x}, \mathbf{p}, t) = \mathbf{F}^{(t)}_{\mathbf{x}} \cdot \mathbf{F}^{(t+1)}_{\mathbf{x}+\mathbf{p}}, \tag{1}$$

where $\cdot$ denotes dot product. For efficiency, we compute the correlation scores of position $\mathbf{x}$ only in its neighborhood of size $P = 2k+1$ by restricting a maximum displacement: $\mathbf{p} \in [-k, k]^2$. For t_{th} frame, a resultant correlation tensor $\mathbf{S}^{(t)}$ is of size $H \times W \times P^2$. The cost of computing the correlation tensor is equivalent to that of 1×1 convolutions with P^2 kernels; the correlation computation can be implemented as 2D convolutions on t_{th} feature map using $t+1_{\text{th}}$ feature map as P^2 kernels. The total FLOPs in a single video amounts to $THWCP^2$. We apply a convolution layer before computing correlations, which learns to weight informative feature channels for learning visual correspondences. In practice, we set the neighborhood $P = 15$ given the spatial resolution 28×28 and apply an $1 \times 1 \times 1$ layer with $C/2$ channels. For correlation computation, we adopt C++/Cuda implemented version of correlation layer in FlowNet [6].

Displacement estimation. From the correlation tensor $\mathbf{S}^{(t)}$, we estimate a displacement field for motion information. A straightforward but non-differentiable method would be to take the best matching displacement for position $\mathbf{x}$ by $\text{argmax}_{\mathbf{p}} s(\mathbf{x}, \mathbf{p}, t)$. To make the operation differentiable, we can use a weighted average of displacements using softmax, called *soft-argmax* [13, 19], which is de-

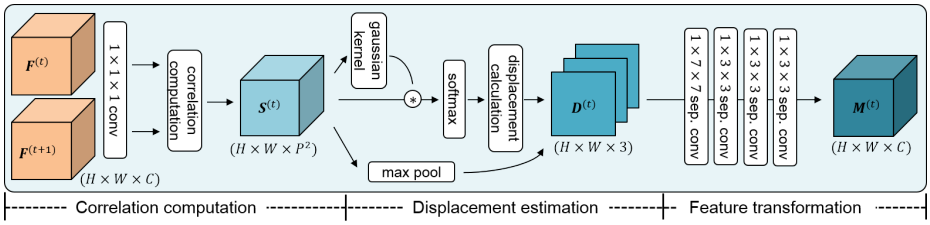


Fig. 3: Overall process of MotionSqueeze (MS) module. The MS module estimates motion across two frame-wise feature maps ($\mathbf{F}^{(t)}, \mathbf{F}^{(t+1)}$) of adjacent frames. A correlation tensor $\mathbf{S}^{(t)}$ is obtained by computing correlations, and then a displacement tensor $\mathbf{D}^{(t)}$ is estimated using the tensor. Through the transformation process of convolution layers, the final motion feature $\mathbf{M}^{(t)}$ is obtained. See text for details.

fined as

$$d(\mathbf{x}, t) = \sum_{\mathbf{p}} \frac{\exp(s(\mathbf{x}, \mathbf{p}, t))}{\sum_{\mathbf{p}'} \exp(s(\mathbf{x}, \mathbf{p}', t))} \mathbf{p}. \tag{2}$$

This method, however, is sensitive to noisy outliers in the correlation tensor since it is influenced by all correlation values. We thus use the *kernel-soft-argmax* [19] that suppresses such outliers by masking a 2D Gaussian kernel on the correlation values; the kernel is centered on each target position so that the estimation is more influenced by closer neighbors. Our kernel-soft-argmax for displacement estimation is defined as

$$d(\mathbf{x}, t) = \sum_{\mathbf{p}} \frac{\exp(g(\mathbf{x}, \mathbf{p}, t) s(\mathbf{x}, \mathbf{p}, t)/\tau)}{\sum_{\mathbf{p}'} \exp(g(\mathbf{x}, \mathbf{p}', t) s(\mathbf{x}, \mathbf{p}', t)/\tau)} \mathbf{p}, \tag{3}$$

where

$$g(\mathbf{x}, \mathbf{p}, t) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{\mathbf{p} - \operatorname{argmax}_{\mathbf{p}} s(\mathbf{x}, \mathbf{p}, t)}{\sigma^2}\right). \tag{4}$$

Note that $g(\mathbf{x}, \mathbf{p}, t)$ is the Gaussian kernel and we empirically set the standard deviation σ to 5. τ is a temperature factor adjusting the softmax distribution; as τ decreases, softmax approaches argmax. We set $\tau = 0.01$ in our experiments.

In addition to the estimated displacement map, we use a confidence map of correlation as auxiliary motion information, which is obtained by pooling the highest correlation on each position $\mathbf{x}$:

$$s^*(\mathbf{x}, t) = \max_{\mathbf{p}} s(\mathbf{x}, \mathbf{p}, t). \tag{5}$$

The confidence map may be useful for identifying displacement outliers and learning informative motion features.

We concatenate the (2-channel) displacement map and the (1-channel) confidence map into a displacement tensor $\mathbf{D}^{(t)}$ of size $H \times W \times 3$ for the next step of motion feature transformation. An example of them is visualized in Figure 4.

Feature transformation. We convert the displacement tensor $\mathbf{D}^{(t)}$ to an effective motion feature $\mathbf{M}^{(t)}$ that is readily incorporated into downstream layers. The tensor $\mathbf{D}^{(t)}$ is fed to four *depth-wise separable convolution* [14] layers, one $1 \times 7 \times 7$ layer followed by three $1 \times 3 \times 3$ layers, and transformed into a motion feature $\mathbf{M}^{(t)}$ with the same number of channels C as that of the original input $\mathbf{F}^{(t)}$. The depth-wise separable convolution approximates 2D convolution with a significantly less computational cost [4, 29, 36]. Note that all depth-wise and point-wise convolution layers are followed by batch normalization [15] and ReLU [25]. As in the temporal stream layers of [31], this feature transformation process is designed to learn task-specific motion features with convolution layers by interpreting the semantics of displacement and confidence. As illustrated in Figure 2, the MS module generates motion feature $\mathbf{M}^{(t)}$ using two adjacent appearance features $\mathbf{F}^{(t)}$ and $\mathbf{F}^{(t+1)}$ and then add it to the input of the next layer. Given T frames, we simply pad the final motion feature $\mathbf{M}^{(T)}$ with $\mathbf{M}^{(T-1)}$ by setting $\mathbf{M}^{(T)} = \mathbf{M}^{(T-1)}$.

3.2 MotionSqueeze network (MSNet).

The MS module can be inserted into any video understanding architecture to improve the performance by motion feature modeling. In this work, we introduce standard convolutional neural networks (CNNs) with the MS module, dubbed *MSNet*, for video classification. We adopt the ImageNet-pretrained ResNet [12] as the CNN backbone and insert TSM [21] for each residual block of the ResNet. TSM enables 2D convolution to obtain the effect of 3D convolution by shifting a part of input feature channels along the temporal axis before the convolution operation. Following the default setting in [21], we shift 1/8 of the input features channels forward and another 1/8 of the channels backward in each TSM.

The overall architecture of the proposed model is shown in Figure 2; a single MS module is inserted after the third stage of the ResNet. We fuse the motion feature into the appearance feature by element-wise addition:

$$\mathbf{F}'^{(t)} = \mathbf{F}^{(t)} + \mathbf{M}^{(t)}. \tag{6}$$

In section 4.5, we extensively evaluate different fusion methods, *e.g.*, concatenation and multiplication, and show that additive fusion is better than the others. After fusing both features, the combined feature is passed through the next downstream layers. The network outputs over T frames are temporally averaged to produce a final output and the cross-entropy with softmax is used as a loss function for training. By default setting, MSNet learns both appearance and motion features jointly in a single network at the cost of only 2.5% and 1.2% increase in FLOPs and the number of parameters, respectively.

4 Experiments

4.1 Datasets

Something-Something V1&V2 [10] are trimmed video datasets for human action classification. Both datasets consist of 174 classes with 108,499 and 220,847 videos in total, respectively. Each video contains one action and the duration spans from 2 to 6 seconds. Something-Something V1&V2 are motion-oriented datasets where temporal relationships are more salient than in others.

Kinetics [17] is a popular large-scale video dataset, consisting of 400 classes with over 250,000 videos. Each video lasts around 10 seconds with a single action.
HMDB51 [18] contains 51 classes with 6,766 videos. Kinetics and HMDB-51 focus more on appearance information rather than motion.

4.2 Implementation details

Clip sampling. In both training and testing, instead of an entire video, we use a clip of frames that are sampled from the video. We use the segment-based sampling method [40] for the Something-Something V1&V2 while adopting the dense frame sampling method [2] for Kinetics and HMDB-51.

Training. For each video, we sample a clip of 8 or 16 frames, resize them into 240×320 images, and crop 224×224 images from the resized images [48]. The minibatch SGD with Nestrov momentum is used for optimization, and the batch size is set to 48. We use scale jittering for data augmentation. For the Something-Something V1&V2, we set the training epochs to 40 and the initial learning rate to 0.01; the learning rate is decayed by 1/10 after 20_{th} and 30_{th} epochs. For Kinetics, we set the training epochs to 80 and the initial learning rate to 0.01; the learning rate is decayed by 1/10 after 40 and 60 epochs. In training our model on HMDB-51, we fine-tune the Kinetics-pretrained model as in [21, 37]. We set the training epochs to 35 and the initial learning rate to 0.001; the learning rate is decayed by 1/10 after 15_{th} and 30_{th} epochs.

Inference. Given a video, we sample a clip and test its center crop. For Something-Something V1&V2, we evaluate both the single clip prediction and the average prediction of 10 randomly-sampled clips. For Kinetics and HMDB-51, we evaluate the average prediction of uniformly-sampled 10 clips from each video.

4.3 Comparison with state-of-the-art methods

Table 1 summarizes the results on Something-Something V1&V2. Each section of the table contains results of 2D CNN methods [20, 22, 40, 47], 3D CNN methods [16, 23, 42, 43, 48], ResNet with TSM (TSM ResNet) [21], and the proposed method, respectively. Most of the results are copied from the corresponding papers, except for TSM ResNet; we evaluate the official pre-trained model of TSM ResNet using a single center-cropped clip per video in terms of top-1 and top-5 accuracies. Our method, which uses TSM ResNet as a backbone, achieves 50.9%

Table 1: Performance comparison on Something-Something V1&V2. The symbol † denotes the reproduced by ours.

model	flow	#frame	FLOPs ×clips	#param	SomethingV1 top-1	SomethingV1 top-5	SomethingV2 top-1	SomethingV2 top-5
TSN [40]		8	16G×1	10.7M	19.5	-	33.4	-
TRN [47]		8	16G×N/A	18.3M	34.4	-	48.8	-
TRN Two-stream [47]	✓	8+8	16G×N/A	18.3M	42.0	-	55.5	-
MFNet [20]		10	N/A×10	-	43.9	73.1	-	-
CPNet [22]		24	N/A×96	-	-	-	57.7	84.0
ECO$_{En}$*Lite* [48]		92	267×1	150M	46.4	-	-	-
ECO Two-stream [48]	✓	92+92	N/A×1	300M	49.5	-	-	-
I3D from [42]		32	153G×2	28.0M	41.6	72.2	-	-
NL-I3D from [42]		32	168G×2	35.3M	44.4	76.0	-	-
NL-I3D+GCN [42]		32	303G×2	62.2M	46.1	76.8	-	-
S3D-G [43]		64	71G×1	11.6M	48.2	78.7	-	-
DFB-Net [23]		16	N/A×1	-	50.1	79.5	-	-
STM [16]		16	67G×30	24.0M	50.7	80.4	64.2	89.8
TSM [21]		8	33G×1	24.3M	45.6	74.2	58.8	85.4
TSM [21]		16	65G×1	24.3M	47.3	77.1	61.2	86.9
TSM$_{En}$ [21]		16+8	98G×1	48.6M	49.7	78.5	62.9	88.1
TSM Two-stream [21]	✓	16+16	129G×1	48.6M	52.6	81.9	65.0†	89.4†
TSM Two-stream [21]	✓	16+16	129G×6	48.6M	-	-	66.0	90.5
MSNet-R50 (ours)		8	34G×1	24.6M	50.9	80.3	63.0	88.4
MSNet-R50 (ours)		16	67G×1	24.6M	52.1	82.3	64.7	89.4
MSNet-R50$_{En}$ (ours)		16+8	101G×1	49.2M	54.4	83.8	66.6	90.6
MSNet-R50$_{En}$ (ours)		16+8	101G×10	49.2M	**55.1**	**84.0**	**67.1**	**91.0**

and 63.0% on Something-Something V1 and V2 at top-1 accuracy, respectively, which outperforms most of 2D CNN and 3D CNN methods, while using a single clip with 8 input frames only. Compared to the TSM ResNet baseline, our method obtains a significant gain of about 5.3% points and 4.2% points at top-1 accuracy at the cost of only 2.5% and 1.2% increase in FLOPs and parameters, respectively. When using 16 frames, our method further improves achieving 52.1% and 64.7% at top-1 accuracy, respectively. Following the evaluation procedure of two-stream networks, we evaluate the ensemble model (MSNet-R50$_{En}$) by averaging prediction scores of the 8-frame and 16-frame models. With the same number of clips for evaluation, it achieves top-1 accuracy 1.8% points and 1.6% points higher than TSM two-stream networks with 22% less computation, even no optical flow needed. Our 10-clip model achieves 55.1% and 67.1% at top-1 accuracy on Something-Something V1 and V2, respectively, which is the state-of-the-art on both of the datasets. As shown in Figure 1, our model provides the best trade-off in terms of accuracy, FLOPs, and the number of parameters.

4.4 Comparison with other motion representation methods

Table 2 summarizes comparative results with other motion representation methods [7, 16, 20, 27, 34] based on RGB frames. The comparison is done on Kinetics

Table 2: Performance comparison with motion representation methods. The symbol ‡ denotes that we only report the backbone FLOPs.

model	flow	#frame	FLOPs ×clips	speed (V/s)	Kinetics Top-1	HMDB51 Top-1
ResNet-50 from [27]		32	132G×25	22.8	61.3	59.4
R(2+1)D [37]		32	152G×115	8.7	72.0	74.3
MFNet from [27]		10	80G‡×10	-	-	56.8
OFF(RGB) [34]		1	N/A×25	-	-	57.1
TVNet [7]		18	N/A×250	-	-	71.0
STM [16]		16	67G×30	-	73.7	72.2
Rep-flow (ResNet-50) [27]		32	132G‡×25	3.7	68.5	76.4
Rep-flow (R(2+1)D) [27]		32	152G‡×25	2.0	75.5	77.1
ResNet-50 Two-stream from [27]	✓	32+32	264G×25	0.2	64.5	66.6
R(2+1)D Two-stream [37]	✓	32+32	304G×115	0.2	73.9	**78.7**
OFF(RGB+Flow+RGB Diff) [34]	✓	1+5+5	N/A×25	-	-	74.2
TSM (reproduced)		8	33G×10	64.1	73.5	71.9
MSNet-R50 (ours)		8	34G×10	54.2	75.0	75.8
MSNet-R50 (ours)		16	67G×10	31.2	**76.4**	77.4

and HMDB51 since the previous methods commonly report their results on them. Each section of the table contains results of conventional 2D and 3D CNNs, motion representation methods [7, 16, 20, 27, 34], two-stream CNNs with optical flows [12, 37], and the proposed method, respectively. OFF, MFNet, and STM [16, 20, 34] use a sub-network or lightweight modules to calculate temporal gradients of frame-wise feature maps. TVNet [7] and Rep-flow [27] internalize iterative TV-L1 flow operations in their networks. As shown in Table 2, the proposed model using 16 frames outperforms all the other conventional CNNs and the motion representation methods [2, 7, 12, 16, 20, 27, 34, 37], while being competitive with the R(2+1)D two-stream [37] that uses pre-computed optical flows. Furthermore, our model is highly efficient than all the other methods in terms of FLOPs, clips, and the number of frames.

Run-time. We also evaluate in Table 2 the inference speeds of several models to demonstrate the efficiency of our method. All the run-times reported are measured on a single GTX Titan Xp GPU, ignoring the time of data loading. For this experiment, we set the spatial size of the input to 224×224 and the batch size to 1. The official codes are used for ResNet, TSM ResNet, and Rep-flow [12, 21, 27] except for R(2+1)D [37] we implemented. In evaluating Rep-flow [27], we use 20 iterations for optimization as in the original paper. The speed of the two-stream networks [31, 37] includes computation time for TV-L1 method on the GPU. The run-time results clearly show the cost of iterative optimizations used in two-stream networks and Rep-flow. In contrast, our model using 16 frames is about 160× faster than the two-stream networks. Compared to Rep-flow ResNet-50, our method performs about 4× faster due to the absence of the iterative optimization process in Rep-flow.

Table 3: Performance comparison with different displacement estimations.

model	FLOPs	top-1	top-5
baseline	14.6G	41.5	71.8
S	14.8G	43.8	74.9
KS	14.9G	44.6	75.4
KS + CM	14.9G	45.5	76.5
KS + CM + BD	15.1G	46.0	76.7

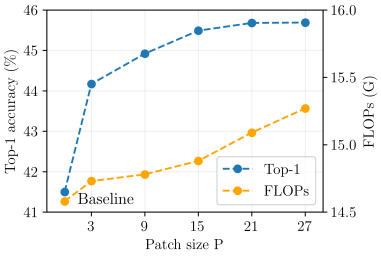


Fig. 4: Top-1 accuracy and FLOPs with different patch sizes.

Table 4: Performance comparison with different positions of the MS module.

model	FLOPs	top-1	top-5
baseline	14.6G	41.5	71.8
res_2	15.6G	45.1	76.1
res_3	14.9G	45.5	76.5
res_4	14.7G	42.6	73.2
res_5	14.6G	41.1	71.8
$res_{2,3,4}$	16.0G	45.7	76.8

Table 5: Performance comparison with different fusing strategies.

model	FLOPs	top-1	top-5
baseline	14.6G	41.5	71.8
MS only	14.1G	38.8	70.7
multiply	14.9G	44.5	75.9
concat.	15.7G	45.0	76.1
add	14.9G	45.5	76.5

4.5 Ablation studies

We conduct ablation studies of the proposed method on Something-Something V1 [10] dataset. We use ImageNet pre-trained TSM ResNet-18 as a default backbone and use 8 input frames for all experiments in this section.

Displacement estimation in MS module. In Table 3, we experiment with different variants of the displacement tensor $\mathbf{D}^{(t)}$ in the MS module. We first compare soft-argmax ('S') and kernel-soft-argmax ('KS') for displacement estimation. As shown in the upper part of Table 3. the kernel-soft-argmax outperforms the soft-argmax, showing the noise reduction effect of Gaussian kernel. In the lower part of Table 3, we evaluate the effect of additional features: confidence maps ('CM') and backward displacement tensor ('BD'). The backward displacement tensor is estimated from $\mathbf{F}^{(t+1)}$ to $\mathbf{F}^{(t)}$. We concatenate the forward and backward displacement tensors, and then pass them to the feature transformation layers. We obtain 0.9% points gain by appending the confidence map to the displacement tensor. Furthermore, by adding backward displacement we obtain another 0.5% points gain at top-1 accuracy, indicating that forward and backward displacement maps complement each other to enrich motion information. We use the kernel-soft-argmax with the confidence map ('KS + CM') as a default method for all other experiments.

Size of matching region. In Figure 4, we evaluate performance varying the spatial size of matching regions of the MS module. Even with a small matching region $P = 3$, it provides a noticeable performance gain of over 2.7% points

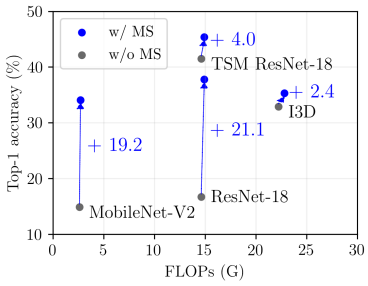


Fig. 5: Top-1 accuracy and FLOPs with MS module on different backbones.

Table 6: Performance comparison with two-stream networks.

model	flow	FLOPs	top-1	top-5
baseline		14.6G	41.5	71.8
Two-stream$_{8+(8\times5)}$	✓	31.4G	46.8	77.3
Two-stream$_{8+(8\times1)}$	✓	28.9G	44.7	75.2
Two-stream$_{8+(8\times1)(low)}$	✓	28.9G	44.1	74.9
MSNet		14.9G	45.5	76.5

to the baseline. The performance tends to increase as the matching region becomes larger due to the larger displacement it can handle between frames. The performance is saturated after $P = 15$.

Position of MS module. In Table 4, we evaluate different positions of the MS module. We denote that res_N by the N-th stage of the ResNet. For each stage, it is inserted right after its final residual block. The result shows that while the MS module is beneficial in most cases, both accuracy and efficiency gains depend on the position of the module. It performs the best at res_3; appearance features from res_2 are not strong enough for accurate feature matching while spatial resolutions of appearance features from res_4 and res_5 are not high enough. The position of the module also affects FLOPs; the computational cost quadratically increases with spatial resolution due to convolution layers of the feature transformation. When inserting multiple MS modules ($res_{2,3,4}$) at the backbone, it marginally improves top-1 accuracy as 0.2% points. Multiple modules appear to generate similar motion information even in different levels of features.

Fusing strategy of MS module. In Table 5, we evaluate different fusion strategies for the MS module; 'MS only', 'multiply', 'concat', and 'add'. In the case of 'MS only', we only pass $\mathbf{M}^{(t)}$ into downstream layers without $\mathbf{F}^{(t)}$. We apply element-wise multiplication and element-wise addition, respectively, for 'multiply' and 'add'. In the case of 'concat', we concatenate $\mathbf{F}^{(t)}$ and $\mathbf{M}^{(t)}$, whose channel size is transformed to C via an $1 \times 1 \times 1$ convolution layer. 'MS only' is less accurate than the baseline because visual semantic information is discarded. While both 'multiply' and 'concat' clearly improve the accuracy, 'add' achieves the best performance with 45.5% at top-1 accuracy. We find that additive fusion is the most effective and stable in amplifying appearance features of moving objects.

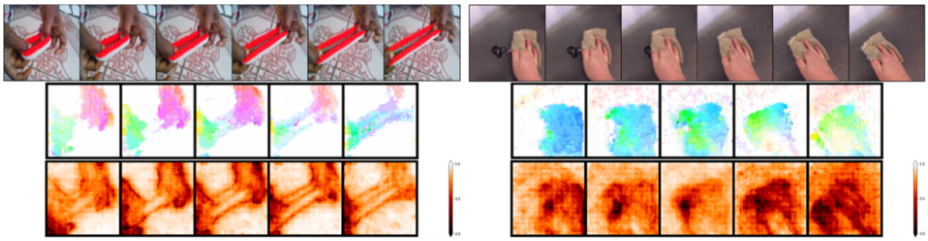
Effect of MS module on different backbones. In Figure 5, we also evaluate the effect of the MS module on ResNet-18, MobileNet-V2, and I3D. We insert one MS module where the spatial resolution of the feature map remains the same. For ResNet-18 and MobileNet-V2, we finetune models pre-trained on ImageNet.

We train I3D from scratch. Our MS module benefits both 2D CNNs and 3D CNNs to obtain higher accuracy. The module significantly improves ResNet-18 and MobileNet-V2 by 21.3% and 19.2% points, respectively, in top-1 accuracy. Since 2D CNNs do not use any spatio-temporal features, it obtains significantly higher gain from the MS module. The MS module also improves I3D and TSM ResNet-18 by 2.4% and 4.0% points, respectively, in top-1 accuracy. The gain on 3D CNNs, although relatively small, verifies that the motion features by the MS module are complementary even to the spatio-temporal features; the MS module learns explicit motion information across adjacent frames whereas TSM covers long-term temporal length using (pseudo-)temporal convolutions.

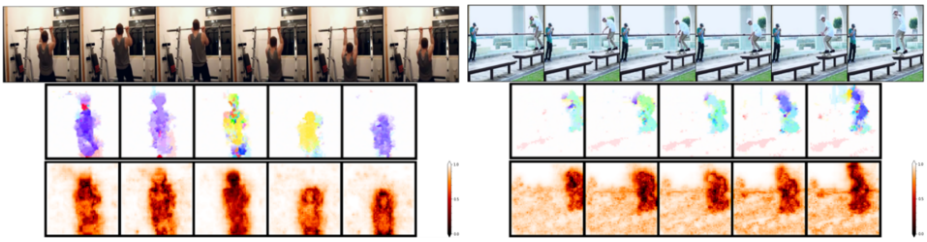
Comparison with two-stream networks. In Table 6, we compare the proposed method with variants of TSM two-stream networks [31] that use TV-L1 optical flows [44]. We denote the two-stream networks by Two-stream$_{N_r+(N_f\times N_s)}$ where N_r, N_f and N_s indicate the number of frames, optical flows, and their stacking size, respectively. For each frame, the two-stream networks use N_s stacked optical flows, which are extracted using the subsequent frames in the original video. Note that those frames for optical flow extraction are not used in our method (MSNet). The second row of Table 6, Two-stream$_{8+(8\times5)}$, shows the performance of standard TSM two-stream networks that use 5 stacked optical flows for the temporal stream. Using the multiple optical flows for each frame outperforms our model in terms of accuracy but requires substantially larger FLOPs as well as an additional computation for calculating optical flows. For a fair comparison, we report the performance of the two-stream networks, Two-stream$_{8+(8\times1)}$, that do not stack multiple optical flows. Our model outperforms the two-stream networks by 0.8% points at top-1 accuracy, with about two times fewer FLOPs. Note that both Two-stream$_{8+(8\times5)}$ and Two-stream$_{8+(8\times1)}$ use optical flows obtained from the original video with a higher frame rate than the input video clip (sampled frames); our method (MSNet) observes the input video clip only. We thus evaluate other two-stream networks, Two-stream$_{8+(8\times1)(low)}$, that uses low-fps optical flows as input; we sample a sequence of frames in 3 fps from the original video and extract TV-L1 optical flows using the sequence. As shown in Table 6, the top-1 accuracy gap between ours and the two-stream network increases to 1.4% points. The result implies that given low-fps videos, our method may further improve over the two-stream networks.

4.6 Visualization

In Figure 6, we present visualization results on Something-Something V1 and Kinetics datasets. They show that our MS module effectively learns to estimate motion without any direct supervision used in training. The first row of each subfigure shows 6 uniformly sampled frames from a video. The second and third rows show color-coded displacement maps [1] and confidence maps, respectively; we apply min-max normalization on the confidence map. The resolution of all the displacement and confidence maps is set to 56×56 for better visualization. As shown in the figures, the MS module captures reliable displacements in most



(a) Label: "Pulling two ends of something so that it gets stretched." (b) Label: "Wiping off something of something."



(c) Label: "Pull ups." (d) Label: "Skateboarding."

Fig. 6: Visualization on Something-Something-V1 (top) and Kinetics (bottom) datasets. RGB images, displacement maps, and the confidence maps are shown from the top row in each subfigure.

cases: horizontal and vertical movements (Figure 6a, 6c, 6d), rotational movements (Figure 6b), and non-severe deformation (Figure 6a, 6d). See the supplementary material for additional details and results. We will make our code and data available online.

5 Conclusion

We have presented an efficient yet effective motion feature block, the MS module, that learns to generate motion features on the fly for video understanding. The MS module can be readily inserted into any existing video architectures and trained by backpropagation. The ablation studies on the module demonstrate the effectiveness of the proposed method in terms of accuracy, computational cost, and model size. Our method outperforms existing state-of-the-art methods on Something-Something-V1&V2 for video classification with only a small amount of additional cost.

Acknowledgements. This work is supported by Samsung Advanced Institute of Technology (SAIT), and also by Basic Science Research Program (NRF-2017R1E1A1A010 77999, NRF-2018R1C1B6001223) and Next-Generation Information Computing Development Program (NRF-2017M3C4A7069369) through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT.

References

1. Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. International Journal of Computer Vision (IJCV) **92**(1), 1–31 (2011) 13
2. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 1, 2, 3, 8, 10
3. Chen, Y., Kalantidis, Y., Li, J., Yan, S., Feng, J.: Multi-fiber networks for video recognition. In: Proc. European Conference on Computer Vision (ECCV) (2018) 3
4. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 7
5. Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term recurrent convolutional networks for visual recognition and description. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015) 1
6. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2015) 3, 5
7. Fan, L., Huang, W., Gan, C., Ermon, S., Gong, B., Huang, J.: End-to-end learning of motion representation for video understanding. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 2, 3, 9, 10
8. Feichtenhofer, C., Pinz, A., Wildes, R.: Spatiotemporal residual networks for video action recognition. In: Proc. Neural Information Processing Systems (NeurIPS) (2016) 3
9. Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional two-stream network fusion for video action recognition. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 3
10. Goyal, R., Kahou, S.E., Michalski, V., Materzynska, J., Westphal, S., Kim, H., Haenel, V., Fruend, I., Yianilos, P., Mueller-Freitag, M., et al.: The" something something" video database for learning and evaluating visual common sense. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2017) 2, 8, 11
11. Han, K., Rezende, R.S., Ham, B., Wong, K.Y.K., Cho, M., Schmid, C., Ponce, J.: Scnet: Learning semantic correspondence. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2017) 3, 4
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 7, 10
13. Honari, S., Molchanov, P., Tyree, S., Vincent, P., Pal, C., Kautz, J.: Improving landmark localization with semi-supervised learning. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 5
14. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017) 7
15. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015) 7

16. Jiang, B., Wang, M., Gan, W., Wu, W., Yan, J.: Stm: Spatiotemporal and motion encoding for action recognition. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2019) 3, 8, 9, 10
17. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al.: The kinetics human action video dataset. arXiv preprint arXiv:1705.06950 (2017) 8
18. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: Hmdb: a large video database for human motion recognition. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2011) 8
19. Lee, J., Kim, D., Ponce, J., Ham, B.: Sfnet: Learning object-aware semantic correspondence. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 3, 4, 5, 6
20. Lee, M., Lee, S., Son, S., Park, G., Kwak, N.: Motion feature network: Fixed motion filter for action recognition. In: Proc. European Conference on Computer Vision (ECCV) (2018) 1, 2, 3, 8, 9, 10
21. Lin, J., Gan, C., Han, S.: Tsm: Temporal shift module for efficient video understanding. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2019) 2, 3, 7, 8, 9, 10
22. Liu, X., Lee, J.Y., Jin, H.: Learning video representations from correspondence proposals. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 4, 8, 9
23. Martinez, B., Modolo, D., Xiong, Y., Tighe, J.: Action recognition with spatial-temporal discriminative filter banks. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2019) 8, 9
24. Min, J., Lee, J., Ponce, J., Cho, M.: Hyperpixel flow: Semantic correspondence with multi-layer neural features. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2019) 3, 4
25. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proc. International Conference on Machine Learning (ICML) (2010) 7
26. Ng, J.Y.H., Choi, J., Neumann, J., Davis, L.S.: Actionflownet: Learning motion representation for action recognition. In: Proc. Winter Conference on Applications of Computer Vision (WACV) (2018) 3
27. Piergiovanni, A., Ryoo, M.S.: Representation flow for action recognition. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 1, 2, 3, 9, 10
28. Rocco, I., Arandjelovic, R., Sivic, J.: Convolutional neural network architecture for geometric matching. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 3, 4
29. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 7
30. Sevilla-Lara, L., Liao, Y., Güney, F., Jampani, V., Geiger, A., Black, M.J.: On the integration of optical flow and action recognition. In: Proc. German Conference on Pattern Recognition (GCPR) (2018) 3
31. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Proc. Neural Information Processing Systems (NeurIPS) (2014) 1, 2, 3, 4, 7, 10, 13
32. Stroud, J., Ross, D., Sun, C., Deng, J., Sukthankar, R.: D3d: Distilled 3d networks for video action recognition. In: Proc. Winter Conference on Applications of Computer Vision (WACV) (2020) 3

33. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 3, 4
34. Sun, S., Kuang, Z., Sheng, L., Ouyang, W., Zhang, W.: Optical flow guided feature: A fast and robust motion representation for video action recognition. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 2, 3, 9, 10
35. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2015) 1, 3
36. Tran, D., Wang, H., Torresani, L., Feiszli, M.: Video classification with channel-separated convolutional networks. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2019) 3, 7
37. Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 2, 3, 8, 10
38. Wang, H., Kläser, A., Schmid, C., Cheng-Lin, L.: Action recognition by dense trajectories. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2011) 1
39. Wang, L., Qiao, Y., Tang, X.: Action recognition with trajectory-pooled deep-convolutional descriptors. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015) 1
40. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporal segment networks: Towards good practices for deep action recognition. In: Proc. European Conference on Computer Vision (ECCV) (2016) 1, 3, 8, 9
41. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 2
42. Wang, X., Gupta, A.: Videos as space-time region graphs. In: Proc. European Conference on Computer Vision (ECCV). pp. 399–417 (2018) 2, 8, 9
43. Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K.: Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In: Proc. European Conference on Computer Vision (ECCV) (2018) 2, 3, 8, 9
44. Zach, C., Pock, T., Bischof, H.: A duality based approach for realtime tv-l 1 optical flow. Pattern Recognition pp. 214–223 (2007) 3, 13
45. Zhao, Y., Xiong, Y., Lin, D.: Recognize actions by disentangling components of dynamics. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 4
46. Zhao, Y., Xiong, Y., Lin, D.: Trajectory convolution for action recognition. In: Proc. Neural Information Processing Systems (NeurIPS) (2018) 1
47. Zhou, B., Andonian, A., Oliva, A., Torralba, A.: Temporal relational reasoning in videos. In: Proc. European Conference on Computer Vision (ECCV) (2018) 2, 8, 9
48. Zolfaghari, M., Singh, K., Brox, T.: Eco: Efficient convolutional network for online video understanding. In: Proc. European Conference on Computer Vision (ECCV) (2018) 2, 3, 8, 9