MVFNet: Multi-View Fusion Network for Efficient Video Recognition

Wenhao Wu[1], **Dongliang He**[1*], **Tianwei Lin**[1], **Fu Li**[1], **Chuang Gan**[2], **Errui Ding**[1]
[1] Department of Computer Vision Technology (VIS), Baidu Inc.
[2] MIT-IBM Watson AI Lab
{wuwenhao01, hedongliang01, lintianwei01, lifu, dingerrui}@baidu.com, ganchuang1990@gmail.com

Abstract

Conventionally, spatiotemporal modeling network and its complexity are the two most concentrated research topics in video action recognition. Existing state-of-the-art methods have achieved excellent accuracy regardless of the complexity meanwhile efficient spatiotemporal modeling solutions are slightly inferior in performance. In this paper, we attempt to acquire both efficiency and effectiveness simultaneously. First of all, besides traditionally treating $H \times W \times T$ video frames as space-time signal (viewing from the Height-Width spatial plane), we propose to also model video from the other two Height-Time and Width-Time planes, to capture the dynamics of video thoroughly. Secondly, our model is designed based on 2D CNN backbones and model complexity is well kept in mind by design. Specifically, we introduce a novel multi-view fusion (MVF) module to exploit video dynamics using separable convolution for efficiency. It is a plug-and-play module and can be inserted into off-the-shelf 2D CNNs to form a simple yet effective model called MVFNet. Moreover, MVFNet can be thought of as a generalized video modeling framework and it can specialize to be existing methods such as C2D, SlowOnly, and TSM under different settings. Extensive experiments are conducted on popular benchmarks (*i.e.*, Something-Something V1 & V2, Kinetics, UCF-101, and HMDB-51) to show its superiority. The proposed MVFNet can achieve state-of-the-art performance but maintain 2D CNN's complexity.

Introduction

With the rapid development of the Internet and mobile devices, video data has exploded over the past years. Huge video information has far exceeded the processing capacity of the conventional manual system and attracted increased research interest in video understanding. Video recognition, as a fundamental task in video analytics, has become one of the most active research topics. It becomes increasingly demanding in a wide range of applications such as video surveillance, video retrieval, and personalized recommendation. Hence, recognition accuracy and inference complexity are of equal importance for the large scale applications.

Recently, significant progress has been achieved in video action recognition following the deep convolutional network paradigm (Simonyan and Zisserman 2014a; Wang et al. 2016; Carreira and Zisserman 2017; Wang et al. 2018b; Lin, Gan, and Han 2019; Feichtenhofer et al. 2019). Full-3D CNNs such as C3D (Tran et al. 2015) and I3D (Carreira and Zisserman 2017), are intuitive spatial-temporal networks which are natural extension over their 2D counterparts to directly tackle 3D volumetric video data. Especially, the good performance achieved in I3D (Carreira and Zisserman 2017) at the cost of thousands of GFLOPs[1]. In these methods, spatial and temporal features are jointly learned in an unconstrained way. However, the huge number of model parameters and computational burdens can largely limit the practicality of these methods.

Then, some works (Qiu, Yao, and Mei 2017; Xie et al. 2018; Tran et al. 2018) try to factorize 3D convolutional kernel into spatial (*e.g.*, 1×3×3) and temporal part (*e.g.*, 3×1×1) separately to reduce the cost. In practice, however, compared with their 2D counterparts, the increased computational overhead is still not negligible. The recent state-

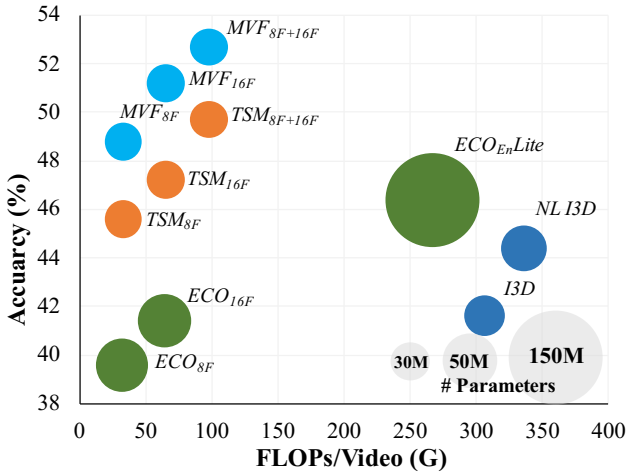


Figure 1: MVF achieves state-of-the-art performance on Something-Something V1 dataset and get better accuracy-computation trade-off than I3D, ECO and TSM. Figure credit: TSM (Lin, Gan, and Han 2019).

*Corresponding author

[1]GFLOPs is short for float-point operations in Giga and is widely used to measure model complexity.

of-the-art model TSM (Lin, Gan, and Han 2019), which achieves a good trade-off between performance and complexity, shifts feature along the temporal dimension instead of temporal convolution to model temporal dynamics in videos. TSM approximates spatiotemporal modeling done in 3D CNN while introduces zero FLOPs for the 2D CNN backbone. This inspires us to focus on designing efficient 2D CNN based architectures to learn more representative features for action recognition.

In this paper, we seek to design action recognition models to acquire both performance and efficiency. First of all, we propose to model dynamics in $H \times W \times T$ video signal from multiple viewpoints for performance improvement and we introduce an efficient spatiotemporal module, termed as *Multi-View Fusion Module* (MVF Module). MVF is a novel plug-and-play module and can turn an existing 2D CNN into a powerful spatiotemporal feature extractor with minimal overhead. Specifically, the MVF module adopts three independent 1D channel-wise convolutions over the T, H and W dimensions respectively to capture multi-view information. To make this module more efficient, we decompose the feature maps into two parts, one acts as inputs of the three 1D channel-wise convolutions for multi-view spatiotemporal modeling and the other is directly concatenated with the outputs of the first part for making the original activation still accessible. In practice, we integrate the MVF module into the standard ResNet block to construct our MVF block. Then the final video architecture MVFNet is constructed by stacking multiple blocks. Interestingly, MVFNet can be regarded as the generalized spatiotemporal model and several existing methods such as C2D, SlowOnly (Feichtenhofer et al. 2019), and TSM (Lin, Gan, and Han 2019) can be specialized from MVFNet with different settings.

Extensive experimental results on multiple well-known datasets, including Kinetics-400 (Kay et al. 2017), Something-Something V1 and V2 (Goyal et al. 2017), UCF-101 (Soomro, Zamir, and Shah 2012) and HMDB-51 (Kuehne et al. 2011) show the superiority of our solution. As shown in Fig. 1, MVFNet achieves excellent performance with quite limited overhead on Something-Something V1 and it is superior compared with existing state-of-the-art frameworks. The same conclusion can be drawn on other datasets. Codes and models are available[2]. Overall, our major contributions are summarized as follows:

- Instead of only temporal modeling, we propose to exploit dynamic inside the three dimensional video signal from multiple viewpoints. A novel MVF module is designed to better exploit spatiotemporal dynamics.
- The MVF module works in a plug-and-play way and can be integrated easily with existing 2D CNN backbones. Our MVFNet is a generalized video modeling network and it can specialize to become recent state-of-the-arts.
- Extensive experiments on five public benchmark datasets demonstrate that the proposed MVFNet outperforms the state-of-the-art methods with computational cost (GFLOPs) comparable to 2D CNN.

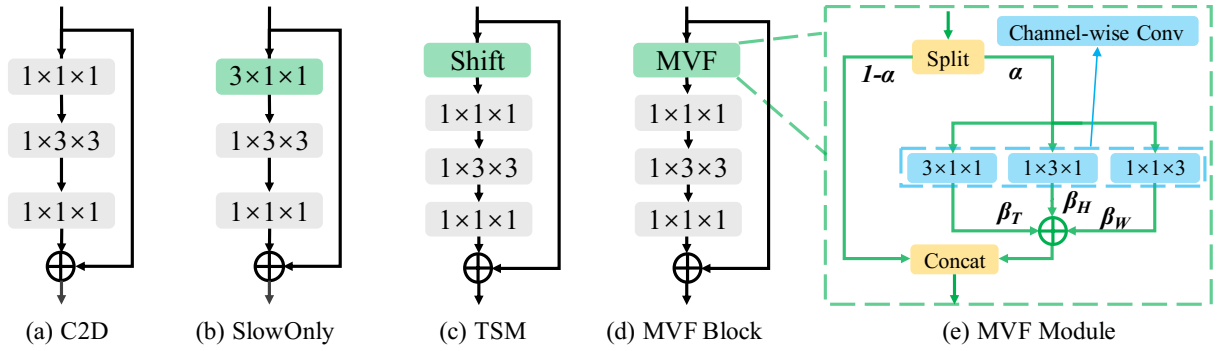
[2]https://github.com/whwu95/MVFNet

Related Work

2D CNNs were extensively applied to conduct video recognition. Over the past years, inspired by the great success of deep convolution frameworks in image recognition (He et al. 2016; Simonyan and Zisserman 2014b; Ioffe and Szegedy 2015), many methods have been proposed to explore the application of deep convolutional architectures on action recognition in videos. Among these methods, the Two-Stream architecture is a popular extension of 2D CNNs to handle video (Simonyan and Zisserman 2014a; Zhang et al. 2016), which can learn video representations respectively from RGB and optical flows or motion vector. To further boost performance, TSN (Wang et al. 2016) proposed a sparse temporal sampling strategy for the two-stream structure. TRN (Zhou et al. 2018) proposed by focusing on the multi-scale temporal relations among sampled frames. More recently, TSM (Lin, Gan, and Han 2019), STM (Jiang et al. 2019), GST (Luo and Yuille 2019), GSM (Sudhakaran, Escalera, and Lanz 2020), TEI (Liu et al. 2020), TEA (Li et al. 2020b) perform efficient temporal modeling.

3D CNNs and (2+1)D CNN variants based approach is another typical branch. C3D (Tran et al. 2015) is the first work in this line which directly learn spatiotemporal features from the video clip with 3D convolution. However, C3D has a huge number of parameters which makes it harder to train and more prone to over-fitting than 2D counterparts. To alleviate such problems, I3D (Carreira and Zisserman 2017) proposed to inflate the ImageNet pre-trained 2D convolution into 3D convolution for initialization. Following the I3D paradigm for spatiotemporal modeling, S3D (Xie et al. 2018), P3D (Qiu, Yao, and Mei 2017), R(2+1)D (Tran et al. 2018) and StNet (He et al. 2019) are proposed to reduce computation overhead of 3D convolution while remaining the spatiotemporal modeling property. These (2+1)D CNN variants decompose 3D convolution into 2D spatial convolution followed by 1D temporal convolution on either per convolution operation basis or per 3D convolution network block basis. There exist several other networks that merge 2D and 3D information in CNN blocks to enhance the feature abstraction capability and resort to shallower backbones for efficiencies, such as ECO (Zolfaghari, Singh, and Brox 2018) and ARTNet (Wang et al. 2018a). More recently, SlowFast (Feichtenhofer et al. 2019) explored the potential of different temporal speeds with two different 3D CNN architectures (*e.g.*, Slow-Only and Fast-Only) to mimic two-stream fusion of 3D CNNs. In fact, our MVFNet can easily replace the Slow path in SlowFast.

The most closely related work to ours is CoST (Li et al. 2019), which also learns features from multiple views for action recognition. However, there are substantial differences between the CoST and the proposed MVF module. CoST learns collaborative spatiotemporal features through weight sharing among three regular 2D 3×3 convolutions on T-H, T-W, and H-W planes and replaces the middle 3×3 convolution of ResNet block with CoST operation. CoST targets on performance and the three 2D 3×3 convolutions are applied to the whole input feature map. On the contrary, we learn independent features via three different 1D channel-wise convolutions over T, H, and W dimensions and col-



(a) C2D (b) SlowOnly (c) TSM (d) MVF Block (e) MVF Module

Figure 2: Illustration of various residual blocks for video action recognition. (a) shows a simple 2D ResNet-block. (b) shows the I3D$_{3\times1\times1}$ type block, which decouples the spatial and temporal filters by inflating the first 1×1 kernel to $3\times1\times1$. (c) shows a TSM block, which shifts partial feature maps along the temporal dimension for efficient temporal modeling. (d) shows our MVF block which integrate the MVF module into standard ResNet block. (e) depict the architecture of the MVF module.

laborative learning does not work for our module. Besides of channel-wise 1D convolution, our MVF module performs multi-view modeling using part of the whole input feature map rather than the whole input feature map as done in CoST. Compared with CoST, our MVFNet achieves superior performance with much smaller computation cost.

There is also active research on dynamic inference (Wu et al. 2020), adaptive frame sampling techniques (Wu et al. 2019b,a; Korbar, Tran, and Torresani 2019) and self-supervised video representation learning (Chen et al. 2021; Han, Xie, and Zisserman 2020), which we think can be complementary to our approach.

Approach

In this section, first we present the technical details of our novel multi-view fusion (MVF) module which encodes spatiotemporal features effectively and efficiently with minimal overhead. Then we describe details of the MVFNet building block and how to insert this block into the off-the-shelf architecture of 2D CNNs. Finally, we present that MVF is a generalization of several existing methods such as C2D, SlowOnly (Feichtenhofer et al. 2019) and TSM (Lin, Gan, and Han 2019).

Multi-View Fusion Module

As shown in Fig 2, we compare the proposed method to several common competitive methods. C2D is a simple 2D baseline architecture without any temporal interaction before the last global average pooling. A natural generalization of C2D is to turn it into a 3D convolutional counterpart. As 3D convolutions are computationally intensive, SlowOnly (the Slow path in SlowFast (Feichtenhofer et al. 2019)) inflated the first 1×1 kernel in a residual block to $3\times1\times1$ instead of inflating the 3×3 kernel in a residual block to $3\times3\times3$, and achieved promising performance. To model temporal structure efficiently, TSM (Lin, Gan, and Han 2019) involves some hand-crafted designs of shifting part of channels at each timestamp forward and backward along the temporal dimension, thus introduces zero FLOPs compared with C2D. An overview of our proposed multi-view fusion module is shown in Fig. 2(e), the input feature map channels are split into two parts, one for making part of the information in the original activation accessible, and the other for multi-view spatiotemporal modeling. In our module, multi-view modeling is performed via channel-wise convolution along the temporal, horizontal and vertical dimensions respectively, then the outputs of the three convolutions is element-wise added. Finally, the two part of features are concatenated together to combine the original activation and the multi-view modeling activation.

Formally, let tensor $X \in \mathbb{R}^{C\times T\times H\times W}$ denote the input feature maps where C is the number of input channels and T, H, W are the temporal and spatial dimensions. Then let X^1, X^2 be the two splits of X along the channel dimension where $X^1 \in \mathbb{R}^{\alpha C\times T\times H\times W}$, $X^2 \in \mathbb{R}^{(1-\alpha)C\times T\times H\times W}$ and α denoted the proportion of input channels for multi-view spatiotemporal features.

$$O_T = \sum_i K^T_{c,i} \cdot X^1_{c,t+i,h,w}, \tag{1}$$

$$O_H = \sum_i K^H_{c,i} \cdot X^1_{c,t,h+i,w}, \tag{2}$$

$$O_W = \sum_i K^W_{c,i} \cdot X^1_{c,t,h,w+i}, \tag{3}$$

where c, t, x, y is the index of different dimensions of channel, time, height and width. K^T, K^H, K^W is the channel-wise convolutional kernels used for modeling H-W, W-$T$and H-T views respectively. In this paper, the kernel size of three channel-wise convolution is $3\times1\times1$, $1\times3\times1$ and $1\times1\times3$. Then, the feature maps O_T, O_H, O_W from three different views are fused by a weighted summation as

$$O^1 = \delta(\beta_T \cdot O_T + \beta_H \cdot O_H + \beta_W \cdot O_W), \tag{4}$$

where δ is activation function, $O^1 \in \mathbb{R}^{\alpha C\times T\times H\times W}$ is the activated feature maps, $\beta_T, \beta_H, \beta_W$ represents the weight value of the corresponding view. In this paper, we simply set $\beta_T = \beta_H = \beta_W = 1$. Finally, we get the MVF module output $Y \in \mathbb{R}^{C\times T\times H\times W}$ as

$$Y = Concat(X^2, O^1), \tag{5}$$

where $Concat$ represents concatenation operation along dimension of channel.

Network Architecture

We here describe how to instantiate the *Multi-View Fusion Network* (MVFNet). The proposed MVF module is a flexible and efficient spatiotemporal module, which can be easily plugged into the existing 2D CNNs with a strong ability to learning the spatioltemporal features in videos. Therefore, our method is able to not only use the pre-trained ImageNet model for initialization to have faster training, but also bring very limited extra computation cost compared with 3D CNNs or (2+1)D CNNs. In practice, our MVFNet obviously improves the performance on different types of large-scale video datasets: scene-related one such as Kinetics-400, and temporal-related one such as Something-Something.

We observed that many recent state-of-the-art action recognition methods (Lin, Gan, and Han 2019; Feichtenhofer et al. 2019; Li et al. 2019; Liu et al. 2020) usually use ResNet as the backbone network due to its simple and modular structure. For a fair comparison with these state-of-the-arts, in our experiments, we instantiate the MVFNet using ResNet as the backbone. Specifically, we integrate the proposed module into standard ResNet block to construct our MVF block then form our MVFNet. The overall design of the MVF block is illustrated in Fig. 2(d), MVF module is inserted before the first convolution of ResNet Block.

Unless specified, in our experiments we choose the 2D ResNet-50 as our backbone for its trade-off between the accuracy and speed. Following recent common practice (Lin, Gan, and Han 2019), there are no temporal downsampling operations in this instantiation. The ResNet backbone is initialized from ImageNet pre-trained weights and our MVF modules do not need any specialized initialization strategy, simple random gaussian initialization work pretty well.

Relation to Existing Methods

Here we discuss the connection between MVF and other methods shown in Fig. 2. Details are described in the following:

- With α=0, MVF module degenerates to C2D which only focus on learning the spatial feature representation.
- With α=1 and β_H=β_W=0, MVF module collapses to SlowOnly (Feichtenhofer et al. 2019) with depth-wise separable convolution. Put simply, the first $1\times1\times1$ convolution of ResNet block perform as a point-wise convolution, then the $3\times1\times1$ channel-wise convolution followed by the $1\times1\times1$ point-wise convolution naturally form the depth-wise separable convolution.
- With α=1/4 and β_H=β_W=0, MVF module could be viewed as a learnable TSM (Lin, Gan, and Han 2019). More specifically, in fact TSM could be viewed as a channel-wise $3\times1\times1$ convolution, where convolution kernel is fixed as [0, 0, 1] for forward shift, and [1, 0, 0] for backward shift.

Above all, we conclude that MVFNet can be regarded as the generalized spatiotemporal model and above approaches can be specialized from MVFNet with different settings.

Experiments

Datasets and Evaluation Metrics

We evaluate our method on three large-scale video recognition benchmarks, including Kinetics-400 (K400) (Kay et al. 2017), Something-Something (Sth-Sth) V1&V2 (Goyal et al. 2017), and other two small-scale datasets, UCF-101 (Soomro, Zamir, and Shah 2012) and HMDB-51 (Kuehne et al. 2011). Kinetics-400 contains 400 human action categories and provides around 240k training videos and 20k validation videos. For Something-Something datasets, the actions therein mainly include object-object and human-object interactions, which require strong temporal relation to well categorizing them. V1 includes about 110k videos and V2 includes 220k video clips for 174 fine-grained classes. To elaborately study the effectiveness of our method on these two types of datasets, we use Kinetics-400 and Sth-Sth V1 for the ablation experiments. Moreover, transfer learning experiments on the UCF-101 and HMDB-51, which are much smaller than Kinetics and sth-sth, is carried out to show the transfer capability of our solution.

We report top-1 and top-5 accuracy (%) for Kinetics and top-1 accuracy (%) for Something-Something V1&V2. For UCF-101 and HMDB-51, we follow the original evaluation scheme using mean class accuracy. Also, we report the computational cost (in FLOPs) as well as the number of model parameters to depict model complexity. In this paper, we only use the RGB frames of these datasets for experiments.

Implementation Details

Training. We utilize 2D ResNet (He et al. 2016) as our backbone and train the model in an end-to-end manner. We use random initialization for our MVF module. Following the similar practice in (Wang et al. 2018b) on Kinetics-400, we sample frames from a set of consecutive 64 frames per video. For Something-Something V1 & V2, we observe that the duration of most videos normally has less than 64 frames, thus we employ the similar uniform sampling strategy to TSN (Wang et al. 2016) to train our model. In our experiments, we sample 4, 8 or 16 frames as a clip. The size of the short side of these frames is fixed to 256 and then random scaling is utilized for data augmentation. Finally, we resize the cropped regions to 224×224 for network training.

On the Kinetics-400 dataset, the learning rate is 0.01 and will be reduced by a factor of 10 at 90 and 130 epochs (150 epochs in total) respectively. For Something-Something V1 & V2 dataset, our model is trained for 50 epochs starting with a learning rate 0.01 and reducing it by a factor of 10 at 30, 40 and 45 epochs. For these large-scale datasets, our models are initialized by pre-trained models on ImageNet (Deng et al. 2009). For UCF-101 and HMDB-51, we followed the common practice to fine-tune from Kinetics pre-trained weights and start training with a learning rate of 0.01 for 25 epochs. The learning rate is decayed by a factor 10 every 10 epochs. For all of our experiments, we utilize SGD with momentum 0.9 and weight decay of 1e-4 to train our models on 8 GPUs. Each GPU processes a mini-batch of 8 video clips by default. When changing to a larger batch

Setting	Sth-sth v1				Kinetics-400			
	#F	Top-1	Top-5	FLOPs	#F	Top-1	Top-5	FLOPs
α=0	8	17.12	43.46	32.88G	4	71.87	90.02	16.44G
α=1/8	8	49.74	78.09	32.90G	4	**74.21**	91.34	16.45G
α=1/4	8	49.24	77.91	32.92G	4	74.18	**91.46**	16.46G
α=1/2	8	**50.48**	**79.14**	32.96G	4	74.21	91.42	16.48G
α=1	8	49.73	77.94	33.04G	4	73.75	91.40	16.52G

(a) Parameter choices of $\boldsymbol{\alpha}$. Backbone: R-50.

Stages	Blocks	Sth-sth v1, $\boldsymbol{\alpha}$=1/2				Kinetics-400, $\boldsymbol{\alpha}$=1/8			
		#F	Top-1	Top-5	FLOPs	#F	Top-1	Top-5	FLOPs
None	0	8	17.12	43.46	32.88G	4	71.87	90.02	16.44G
res{5}	3	8	46.02	75.60	32.90G	4	73.46	91.09	16.44G
res{4,5}	9	8	**50.48**	**79.14**	32.96G	4	74.21	91.34	16.45G
res{3,4,5}	13	8	49.72	78.82	33.04G	4	74.08	91.51	16.46G
res{2,3,4,5}	16	8	49.95	77.96	33.12G	4	**74.22**	**91.56**	16.47G

(b) The number of MVF Blocks inserted into R-50.

Views	Sth v1		K400	
	#F	Top-1	#F	Top-1
T	8	49.13	4	73.72
T-H	8	49.22	4	74.01
T-W	8	49.31	4	73.88
T-H-W	8	**50.48**	4	**74.21**
T-H-W (S)	8	47.21	4	73.81

(c) Study on the different views of MVF module. Backbone: R-50. S denotes weight sharing.

Method	Sth v1	K400	FLOPs	Params
	Top-1	Top-1		
C2D	17.1	71.4	32.9G	24.3M
TSM	47.2	74.1	32.9G	24.3M
SlowOnly	-	74.9	41.9G	32.4M
CoST*	-	-	45.8G	24.3M
MVFNet	**50.5**	**76.0**	32.9G	24.3M

(d) Study on the effectiveness of MVFNet. Backbone: R-50, 8f input. * indicates our implementation.

	#F	Top-1	FLOPs
R-50	4	74.21	16.45G
	8	75.99	32.90G
	16	77.04	65.81G
R-101	4	75.98	31.36G
	8	77.46	62.72G
	16	78.42	125.45G

(e) Advanced backbones for MVFNet on Kinetics-400.

	Model	Top-1	FLOPs
Mb-V2	C2D	64.4	1.25G
	MVF	67.5	1.25G
R-50	C2D	71.9	16.44G
	MVF	74.2	16.48G

(f) Different backbones for MVFNet on Kinetics-400. Mb-V2 denotes MobileNet-V2.

Table 1: Ablation studies on Something-Something V1 and Kinetics-400. We show top-1 and top-5 classification accuracy (%), as well as computational complexity measured in FLOPs (floating-point operations) for a single clip input of spatial size 224^2. #F indicates the number of frames sampled from each video clip. We follow the common setting to sample multiple clips per video (10 for Kinetics-400, 2 for Something-Something V1).

size b of each GPU for higher gpu memory usage, we use linear scaling initial learning rate (0.01×b/8).

Inference. Following the widely used practice in (Lin, Gan, and Han 2019; Liu et al. 2020), two ways for inference are considered to trade-off accuracy and speed. (a) For high accuracy, we follow the common setting in (Wang et al. 2018b; Feichtenhofer et al. 2019) to uniformly sample multiple clips from a video along its temporal axis. We sample 10 clips for Kinetics-400 and 2 clips for others. For each clip, we resize the short side to 256 pixels and take 3 spatial crops in each frame. Finally, we average the softmax probabilities of all clips as the final prediction. (b) For efficiency, we only use 1 clip per video and a central region of size 224×224 is cropped for evaluation.

Ablation Studies

To comprehensively evaluate our proposed MVF module, in this section we provide ablation studies on both Kinetics-400 and Something-Something V1 datasets which represent the two types of datasets. Table 1 shows a series of ablations. Accordingly, the effectiveness of each component in our framework is analyzed as follows.

Paramter Choice. As shown in Table 1a, we compare networks with different proportion of input channels ($\alpha = 0, 1/8, 1/4, 1/2, 1$) for multi-view spatiotemporal feature. Here we add MVF blocks into res_{4-5} for efficiency. Especially, when $\alpha = 0$, MVFNet becomes exactly C2D. Our approach achieves considerable absolute improvement over C2D baseline on both datasets (+33.36% for Sth-Sth v1, +2.34% for Kinetics-400), which demonstrates the effectiveness of the MVF blocks. For Kinetics-400, we observe that the change in $\alpha = 1/8, 1/4, 1/2$ appeared to have little impact on performance thus we choose $\alpha = 1/8$ for efficiency in the following experiments. As for Something-Something V1, our method with $\alpha = 1/2$ achieves the highest Top-1 accuracy compared with the other settings, so 1/2 is adopted in the following experiments.

The Number of MVF Blocks. We denote the conv2_x to conv5_x of ResNet architecture as res_2 to res_5. To figure out how many MVF blocks can obtain a good trade-off, we gradually add MVF blocks from res_5 to res_2 in ResNet-50. According to the results in Table 1a, we set α to $1/2$ for Something-something V1 and $1/8$ for Kinetics-400. As shown in Table 1b, on Kinetics-400, the improvement brought by MVF blocks on res_{4-5}, res_{3-5} or res_{2-5} is comparable. res_{2-5} outperforms res_{4-5} by 0.01% but 7 extra MVF-blocks are needed. On Something-something V1, MVF-blocks added to res_{4-5} achieves the highest top-1 accuracy. Thus we use MVF block in stages of res_{4-5} in the following experiments for efficiency.

Different Views. We also evaluate impacts of different views in MVF module. As illustrated in Fig. 2(e), we can get different multi-view fusion by controlling β_T, β_H and β_W. For example, we simply set β_T=β_H=β_W=1 to get view of T-H-W. Also, typical T-H, T-W, T views can be easily obtained. From Table 1c, we can see that T-H-W outperforms T by 1.35% and 0.49% on Something-Something V1 and Kinetics-400, respectively. Moreover, we also try the collaborative feature learning in CoST (Li et al. 2019) by sharing the he convolution kernels among different views. However, with weight sharing among different views, accuracy gets degraded by 3.2% and 0.4% on two datasets, respectively.

Comparison with Other Temproal Modules. Here we make a comparison with methods described in Sec. under the same setting of backbone and inputs. We list FLOPs and the number of parameter for all models in Table 1d, our

Method	Backbone	Frames×Crops×Clips	GFLOPs	Top-1	Top-5
I3D (Carreira et al. 2017)	Inception V1	64×N/A×N/A	108×N/A	72.1%	90.3%
S3D-G (Xie et al. 2018)	Inception V1	64×3×10	71.4×30	74.7%	93.4%
TSN (Wang et al. 2016)	Inception V3	25×10×1	80×10	72.5%	90.2%
ECO-RGB$_{En}$ (Zolfaghari et al. 2018)	BNIncep+Res3D-18	92×1×1	267×1	70.0%	-%
R(2+1)D (Tran et al. 2018)	ResNet-34	32×1×10	152×10	74.3%	91.4%
X3D-M (Feichtenhofer 2020)	-	16×3×10	6.2×30	76.0%	92.3%
STM (Jiang et al. 2019)	ResNet-50	16×3×10	67×30	73.7%	91.6%
TSM (Lin, Gan, and Han 2019)	ResNet-50	8×3×10	33×30	74.1%	91.2%
SlowOnly (Feichtenhofer et al. 2019)	ResNet-50	8×3×10	41.9×30	74.9%	91.5%
TEINet (Liu et al. 2020)	ResNet-50	8×3×10	33×30	74.9%	91.8%
TEA (Li et al. 2020b)	ResNet-50	8×3×10	33×30	75.0%	91.8%
Slowfast (Feichtenhofer et al. 2019)	R50+R50	(4+32)×3×10	36.1×30	75.6%	92.1%
NL+I3D (Wang et al. 2018b)	ResNet-50	32×3×10	70.5×30	74.9%	91.6%
NL+I3D (Wang et al. 2018b)	ResNet-50	128×3×10	282×30	76.5%	92.6%
MVFNet	ResNet-50	8×3×10	32.9×30	**76.0**%	**92.4**%
MVFNet	ResNet-50	16×3×10	65.8×30	**77.0**%	**92.8**%
ip-CSN (Tran et al. 2019)	ResNet-101	32×3×10	82×30	76.7%	92.3%
SmallBig (Li et al. 2020a)	ResNet-101	32×3×4	418×12	77.4%	93.3%
SlowOnly (Feichtenhofer et al. 2019)	ResNet-101	16×3×10	185×30	77.2%	-%
NL+I3D (Wang et al. 2018b)	ResNet-101	128×3×10	359×30	77.7%	93.3%
Slowfast (Feichtenhofer et al. 2019)	R101+R101	(8+32)×3×10	106×30	77.9%	93.2%
Slowfast (Feichtenhofer et al. 2019)	R101+R101	(16+64)×3×10	213×30	**78.9%**	**93.5%**
TPN (Yang et al. 2020)	ResNet-101	32×3×10	374×30	**78.9%**	**93.9%**
MVFNet	ResNet-101	8×3×10	62.7×30	77.4%	92.9%
MVFNet	ResNet-101	16×3×10	125.4×30	78.4%	93.4%
MVFNet$_{En}$	R101+R101	(16+8)×3×10	188.1×30	**79.1**%	**93.8**%

Table 2: Comparison with the state-of-the-art models on Kinetics-400. Similar to (Feichtenhofer et al. 2019), we report the inference cost by computing the GFLOPs (of a single view) × the number of views (temporal clips with spatial crops). N/A denotes the numbers are not available for us.

MVFNet is more lightweight (32.9G *vs.* 41.9G) than the SlowOnly branch of SlowFast (Feichtenhofer et al. 2019) and achieves a better accuracy than it (**76.0**% *vs.* 74.9%). Also, our MVFNet outperforms the TSM (Lin, Gan, and Han 2019) with a large margin on both datasets (Sth V1: **50.5**% *vs.* 47.2%, K400: **76.0**% *vs.* 74.1%) while remaining similar computational cost. For a fair comparison with CoST (Li et al. 2019), we implement the architecture based on our baseline backbones by adding the CoST module into res4 and res5. Feeding 8-frame clips, as expected, our MVFNet-R50 is more lightweight than CoST-R50 (32.9G *vs.* 46G). Table 1d shows the superiority of our MVFNet is quite impressive.

Deeper Backbone. In Table 1e we compare various instantiations of MVFNet models on Kinetics-400. Thus far, all experiments used ResNet-50 as the backbone, we further study the performance of our MVFNet with a deeper backbone (*i.e.*, ResNet-101 (R-101)). For models involving R-101, we use the same hyper-parameter settings as R-50 above. As expected, using advanced backbones is complementary to our method. Comparing with the R-50 counterparts, our MVFNet gets additional improvement on R-101. We also investigate generalization of our models on longer input videos. Our models work well on longer sequences and all models have better results with longer inputs.

Different Backbones. We further study the performance of MVFNet with MobileNet-V2 which is much smaller than the ResNet backbone. As shown in Table 1f, comparing with the 4frame-C2D, our 4frame-MVFNet achieves a better accuracy (67.5% vs. 64.4%) with the same MobileNet-V2 backbone on Kinetics-400.

Comparison with State-of-the-arts

Results on Kinetics-400. We make a comprehensive comparison in Table 2, where our MVFNet outperforms the recent SOTA approaches on Kinetics-400. Here we only list the models using RGB as inputs to perform comparisons. Compared with 2D CNN based models, when utilizing 8 frames as input, our MVFNet outperforms these current state-of-the-art efficient methods (*e.g.*, TSM, TEINet and TEA) with a clear margin (**76.0**% *vs.* 74.1%/74.9%/75.0%). Compared with computationally expensive models, MVFNet-R50 with 8-frame outperforms NL I3D-R50 with 32-frame (**76.0**% *vs.* 74.9%) but only uses **2.1**× less GFLOPs, and MVFNet-R50 with 16-frame uses **4.2**× less GFLOPs than NL I3D-R50 with 128-frame but achieves a better accuracy (**77.0**% *vs.* 76.5%). Moreover, 8-frame MVFNet-R101 achieves a competitive accuracy (**77.4**% *vs.* 77.2%, 77.4%, 77.7%) when using **3**×, **5.7**×, **6.7**× less GFLOPs than SlowOnly, NL I3D and SmallBig respectively. To mimics two-steam fusion with two temporal rates as done in SlowFast, we perform score fusion over 16-frame MVFNet-R101 and 8-frame MVFNet-R101. Our MVFNet$_{En}$ obtains better performance than SlowFast

Method	Backbone	Frames×Crops×Clips	FLOPs	Pre-train	V1 Val Top-1 (%)	V2 Val Top-1 (%)
I3D (Wang et al. 2018)	3D ResNet50		153G×3×2	ImageNet	41.6	-
NL I3D (Wang et al. 2018)	3D ResNet50	32×3×2	168G×3×2	+	44.4	-
NL I3D+GCN (Wang et al. 2018)	3D ResNet50+GCN		303G×3×2	K400	46.1	-
ECO (Zolfaghari et al. 2018)	BNIncep+3D Res18	8×1×1	32G×1×1	K400	39.6	-
ECO_{En} (Zolfaghari et al. 2018)		92×1×1	267G×1×1		46.4	-
S3D-G (Xie et al. 2018)	Inception	64×1×1	71G×1×1	K400	48.2	-
TSN (Wang et al. 2016)	ResNet50	8×3×2	33G×3×2	ImageNet	20.5	30.4
TSM (Lin et al. 2019)	ResNet50	8×3×2	33G×3×2	ImageNet	47.2	61.2
		16×3×2	65G×3×2		48.4	63.1
STM (Jiang et al. 2019)	ResNet50	8×3×10	33G×3×10	ImageNet	49.2	62.3
		16×3×10	67G×3×10		50.7	64.3
TEINet (Liu et al. 2020)	ResNet50	8×3×10	33G×3×10	ImageNet	48.8	64.0
		16×3×10	66G×3×10		51.0	64.7
TEA (Li et al. 2020b)	ResNet50	8×3×10	35G×3×10	ImageNet	51.7	-
		16×3×10	70G×3×10		52.3	-
MVFNet	ResNet50	8×1×1	33G×1×1	ImageNet	48.8	60.8
		8×3×2	33G×3×2		50.5	63.5
		16×1×1	66G×1×1		51.0	62.9
		16×3×2	66G×3×2		**52.6**	**65.2**
		(16+8)×3×2	99G×3×2		**54.0**	**66.3**

Table 3: Performance and FLOPs consumptions of our method on the Something-Something V1 and V2 datasets compared with the state-of-the-art methods.

(**79.1%** *vs.* 78.9%), using less GFLOPs (188G *vs.* 213G). TPN-I3D-101 achieves top-1 accuracy of 78.9% with 32-frame clips as input, its complexity is as high as 374 GFLOPs. Our MVFNet$_{En}$ obtains better performance than TPN-R101, using 2×less GFLOPs (188G *vs.* 374G).

Results on Something-Something V1 & V2. The Something-Something datasets are more complicated than Kinetics. The comparison of our solution against existing state-of-the-arts are list in Table 3. These methods can be divided into two categories as shown in the two parts of Table 3. The upper part presents the 3D CNN based methods including S3D-G, ECO and I3D+GCN models. When compared with these methods, the obtained result shows substantial improvements and the #FLOPs of our model is much smaller than these models. The lower part is 2D CNN based methods. Compared with these lightweight models which also target at improving efficiency, our solution yields a superior accuracy of 52.6% on Something-Something V1 validation set and 65.2% on V2 with 16 frames as input. For readers' reference, here we also report the results of ensemble the models using 16 frames and 8 frames as inputs.

Transfer Learning on UCF-101 & HMDB-51

We also evaluate the performance of our method on UCF-101 and HMDB-51 to show the generalization ability of MVFNet on smaller datasets. We finetune our MVFNet with 16 frames as inputs on these two datasets using model pre-trained on Kinetics-400 and report the mean class accuracy over three splits. From Table 4, we see that our model shows a pretty transfer capability and the mean class accuracy is 96.6% and 75.7% on UCF-101 and HMDB-51, respectively. As shown in the middle part of Table 4, our MVFNet outperforms these 2D CNN based lightweight models. When comparing with the state-of-the-art models based on 3D convolutions such as I3D, R(2+1)D and S3D, our proposed MVFNet also obtains comparable or better performance.

Method	Backbone	UCF-101	HMDB-51
ECO_{En}	BNIncep+Res3D-18	94.8%	72.4%
ARTNet	ResNet-18	94.3%	70.9%
I3D	Inception V1	95.6%	74.8%
R(2+1)D	Inception V1	96.8%	74.5%
S3D-G	Inception V1	96.8%	75.9%
TSN	BNInception	91.1%	-
StNet	ResNet-50	93.5%	-
TSM	ResNet-50	95.9%	73.5%
STM	ResNet-50	96.2%	72.2%
TEINet	ResNet-50	96.7%	72.1%
MVFNet	ResNet-50	**96.6**%	**75.7**%

Table 4: Mean class accuracy on UCF-101 and HMDB-51 achieved by different methods which are transferred from their Kinetics models with RGB modality (over 3 splits).

Conclusion

In this paper, we presented the Multi-View Fusion (MVF) module to better exploit spatiotemporal dynamics in videos. The MVF module works in a plug-and-play way and can be easily integrated into standard ResNet block to form a MVF block for constructing our MVFNet. We conducted a series of empirical studies to verify the effectiveness of MVFNet for video action recognition. Without any 3D convolution or pre-calculation of optical flow, the experimental results show that our method achieves the new state-of-the-art results on both temporal information dependent and spatial information dominated datasets with computational cost comparable to 2D CNN. In the future, we think the learnable β to weight views will further boost the performance.

References

Carreira, J.; and Zisserman, A. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proc. CVPR*, 6299–6308.

Chen, P.; Huang, D.; He, D.; Long, X.; Zeng, R.; Wen, S.; Tan, M.; and Gan, C. 2021. RSPNet: Relative Speed Perception for Unsupervised Video Representation Learning. In *Proc. AAAI*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 248–255.

Feichtenhofer, C. 2020. X3D: Expanding Architectures for Efficient Video Recognition. In *Proc. CVPR*, 203–213.

Feichtenhofer, C.; Fan, H.; Malik, J.; and He, K. 2019. Slowfast networks for video recognition. *Proc. ICCV* 6202–6211.

Goyal, R.; Kahou, S. E.; Michalski, V.; Materzynska, J.; Westphal, S.; Kim, H.; Haenel, V.; Fruend, I.; Yianilos, P.; Mueller-Freitag, M.; et al. 2017. The" Something Something" Video Database for Learning and Evaluating Visual Common Sense. In *Proc. ICCV*, 5842–5850.

Han, T.; Xie, W.; and Zisserman, A. 2020. Self-supervised Co-Training for Video Representation Learning. In *Neurips*.

He, D.; Zhou, Z.; Gan, C.; Li, F.; Liu, X.; Li, Y.; Wang, L.; and Wen, S. 2019. StNet: Local and global spatial-temporal modeling for action recognition. In *Proc. AAAI*, 8401–8408.

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

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, 448–456.

Jiang, B.; Wang, M.; Gan, W.; Wu, W.; and Yan, J. 2019. Stm: Spatiotemporal and motion encoding for action recognition. In *Proc. ICCV*, 2000–2009.

Kay, W.; Carreira, J.; Simonyan, K.; Zhang, B.; Hillier, C.; Vijayanarasimhan, S.; Viola, F.; Green, T.; Back, T.; Natsev, P.; et al. 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950* .

Korbar, B.; Tran, D.; and Torresani, L. 2019. Scsampler: Sampling salient clips from video for efficient action recognition. In *Proc. ICCV*, 6232–6242.

Kuehne, H.; Jhuang, H.; Garrote, E.; Poggio, T.; and Serre, T. 2011. HMDB: a large video database for human motion recognition. In *Proc. ICCV*, 2556–2563.

Li, C.; Zhong, Q.; Xie, D.; and Pu, S. 2019. Collaborative spatiotemporal feature learning for video action recognition. In *Proc. CVPR*, 7872–7881.

Li, X.; Wang, Y.; Zhou, Z.; and Qiao, Y. 2020a. SmallBigNet: Integrating Core and Contextual Views for Video Classification. In *Proc. CVPR*, 1092–1101.

Li, Y.; Ji, B.; Shi, X.; Zhang, J.; Kang, B.; and Wang, L. 2020b. TEA: Temporal Excitation and Aggregation for Action Recognition. In *Proc. CVPR*, 909–918.

Lin, J.; Gan, C.; and Han, S. 2019. TSM: Temporal Shift Module for Efficient Video Understanding. In *Proc. ICCV*, 7083–7093.

Liu, Z.; Luo, D.; Wang, Y.; Wang, L.; Tai, Y.; Wang, C.; Li, J.; Huang, F.; and Lu, T. 2020. TEINet: Towards an Efficient Architecture for Video Recognition. In *Proc. AAAI*, 11669–11676.

Luo, C.; and Yuille, A. L. 2019. Grouped spatial-temporal aggregation for efficient action recognition. In *Proc. ICCV*, 5512–5521.

Qiu, Z.; Yao, T.; and Mei, T. 2017. Learning spatio-temporal representation with pseudo-3d residual networks. In *Proc. ICCV*, 5533–5541.

Simonyan, K.; and Zisserman, A. 2014a. Two-stream convolutional networks for action recognition in videos. In *Neurips*, 568–576.

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

Soomro, K.; Zamir, A. R.; and Shah, M. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402* .

Sudhakaran, S.; Escalera, S.; and Lanz, O. 2020. Gate-Shift Networks for Video Action Recognition. In *Proc. CVPR*, 1102–1111.

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

Tran, D.; Wang, H.; Torresani, L.; and Feiszli, M. 2019. Video classification with channel-separated convolutional networks. In *Proc. ICCV*, 5552–5561.

Tran, D.; Wang, H.; Torresani, L.; Ray, J.; LeCun, Y.; and Paluri, M. 2018. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *Proc. CVPR*, 6450–6459.

Wang, L.; Li, W.; Li, W.; and Van Gool, L. 2018a. Appearance-and-Relation Networks for Video Classification. In *Proc. CVPR*, 1430–1439.

Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; and Van Gool, L. 2016. Temporal segment networks: Towards good practices for deep action recognition. In *Proc. ECCV*, 20–36.

Wang, X.; Girshick, R.; Gupta, A.; and He, K. 2018b. Non-Local Neural Networks. In *Proc. CVPR*, 7794–7803.

Wu, W.; He, D.; Tan, X.; Chen, S.; and Wen, S. 2019a. Multi-Agent Reinforcement Learning Based Frame Sampling for Effective Untrimmed Video Recognition. In *Proc. ICCV*, 6222–6231.

Wu, W.; He, D.; Tan, X.; Chen, S.; Yang, Y.; and Wen, S. 2020. Dynamic Inference: A New Approach Toward Efficient Video Action Recognition. In *Proceedings of CVPR Workshops*, 676–677.

Wu, Z.; Xiong, C.; Ma, C.-Y.; Socher, R.; and Davis, L. S. 2019b. Adaframe: Adaptive frame selection for fast video recognition. In *Proc. CVPR*, 1278–1287.

Xie, S.; Sun, C.; Huang, J.; Tu, Z.; and Murphy, K. 2018. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proc. ECCV*, 305–321.

Yang, C.; Xu, Y.; Shi, J.; Dai, B.; and Zhou, B. 2020. Temporal pyramid network for action recognition. In *Proc. CVPR*, 591–600.

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

Zhou, B.; Andonian, A.; Oliva, A.; and Torralba, A. 2018. Temporal relational reasoning in videos. In *Proc. ECCV*, 803–818.

Zolfaghari, M.; Singh, K.; and Brox, T. 2018. ECO: Efficient Convolutional Network for Online Video Understanding. In *Proc. ECCV*, 695–712.