MARS: Motion-Augmented RGB Stream for Action Recognition

Nieves Crasto[1] Philippe Weinzaepfel[1] Karteek Alahari[2] Cordelia Schmid[2]
[1]NAVER LABS Europe* [2]Inria†

Abstract

Most state-of-the-art methods for action recognition consist of a two-stream architecture with 3D convolutions: an appearance stream for RGB frames and a motion stream for optical flow frames. Although combining flow with RGB improves the performance, the cost of computing accurate optical flow is high, and increases action recognition latency. This limits the usage of two-stream approaches in real-world applications requiring low latency. In this paper, we introduce two learning approaches to train a standard 3D CNN, operating on RGB frames, that mimics the motion stream, and as a result avoids flow computation at test time. First, by minimizing a feature-based loss compared to the Flow stream, we show that the network reproduces the motion stream with high fidelity. Second, to leverage both appearance and motion information effectively, we train with a linear combination of the feature-based loss and the standard cross-entropy loss for action recognition. We denote the stream trained using this combined loss as Motion-Augmented RGB Stream (MARS). As a single stream, MARS performs better than RGB or Flow alone, for instance with 72.7% accuracy on Kinetics compared to 72.0% and 65.6% with RGB and Flow streams respectively.

1. Introduction

The emergence of convolutional neural networks (CNNs) [13, 19, 36], together with larger datasets [10, 18] have recently led to remarkable progress in action recognition [2, 32, 37]. To integrate temporal information with CNNs, three main ideas have been proposed. Simonyan and Zisserman [32] introduced a two-stream approach where one stream models appearance by taking RGB frames as input, and the other processes optical flow frames to leverage motion information. Tran *et al*. [37] proposed an architecture with 3D convolutions on RGB frames, *i.e*., convolutions operating over both space and time. And lastly, recurrent neural networks, such as LSTMs, have been used to aggregate information iteratively over frames [5]. Recent methods [2, 38, 44] are based on the combination of the two-stream approach with 3D convolutions in each stream that are trained using large datasets [18].

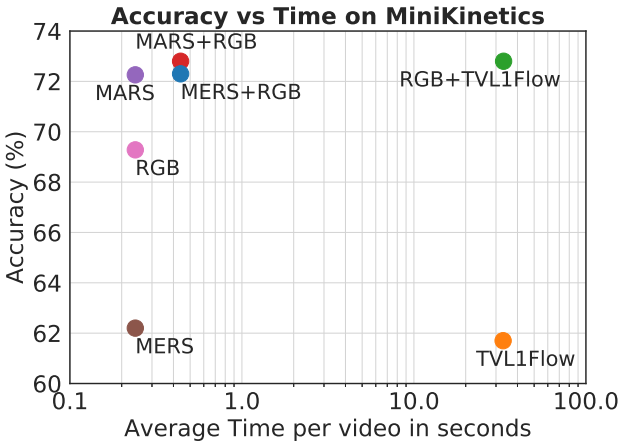


Figure 1: Accuracy *vs*. time on MiniKinetics for different streams using 16-frame clips. Time is averaged over all videos. Flows are estimated using TV-L1 [45]. Our MERS approach successfully mimics the Flow stream, while being significantly faster, as it avoids flow computation. Combining MERS with the RGB stream (MERS+RGB), we achieve accuracy comparable to RGB+Flow at a significantly lower computation cost. Rather than using two streams, our MARS approach is twice as fast as MERS+RGB and maintains the same performance. Note that computation time only depends on the input size, irrespective of the dataset: MARS is ∼100 times faster than RGB+TVL1Flow.

In summary, the strategy of combining 3D CNN-based RGB and Flow streams produces the best results, but it does have significant drawbacks. Firstly, two-stream approaches require explicit and accurate optical flow extraction from RGB frames, which is computationally expensive, as shown in the accuracy *vs*. time plot on the MiniKinetics dataset [44] in Figure 1. From the plot, we observe that Flow and RGB+Flow are significantly slower than RGB. Efficient methods for computing flow do exist [34], but they are not as effective when combined with the RGB stream

*Work done while N. Crasto was an intern at Inria and NAVER LABS.
†Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France

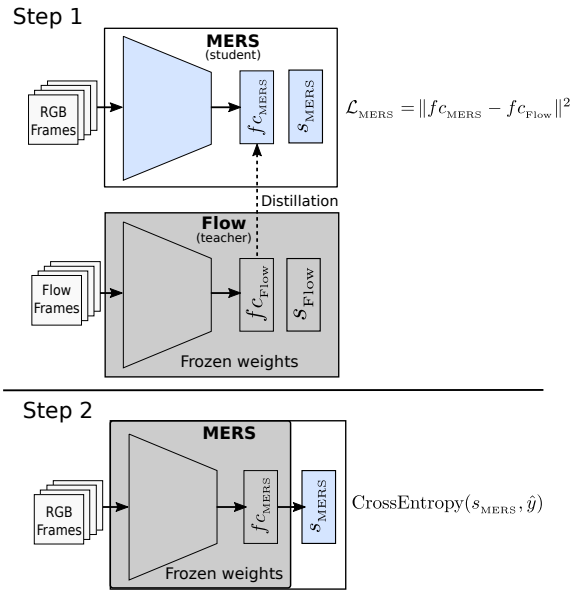


Figure 2: Training to mimic the Flow stream. We first train the Flow stream to classify actions using optical flow clips with cross entropy loss and freeze its weights. To mimic flow features using RGB frames, in step 1, we backpropagate the MSE loss through all the layers of MERS except the last layer. In step 2, we separately train the last layer of MERS with a cross entropy loss.

(see Figure 4a, and also noted in [30]). Secondly, optical flow needs to be estimated before a forward pass of the network can be computed. Thus, two-stream approaches not only require heavy computational resources, but also lead to high latency for recognizing actions in an online scenario. As a consequence, they cannot be applied in real-world applications, even when the architecture is optimized [44].

In this paper, we propose two novel learning strategies, based on the concept of distillation [14] and learning under privileged information [39], to avoid flow computation at test time, while preserving the performance of two-stream approaches. To begin with, we train a standard 3D CNN that takes RGB as input, and hallucinates features from the Flow stream. More precisely, we minimize the difference between features from the layer preceding the last fully-connected layer of the network, and features at the same level from the motion stream (see Figure 2). In other words, our stream is similar to the RGB stream in terms of architecture and inputs, but is trained using a different loss function. We show that by using this approach, Flow features can be obtained from RGB frames without explicit optical flow computation during inference. For ease of notation, we denote this network as **M**otion-**E**mulated **RGB** **S**tream (MERS). MERS shows that, by accurately mimicking the Flow stream, one can effectively transfer knowledge gained from optical flow to a stream with RGB inputs based on 3D convolutions. More importantly, it also implies that flow

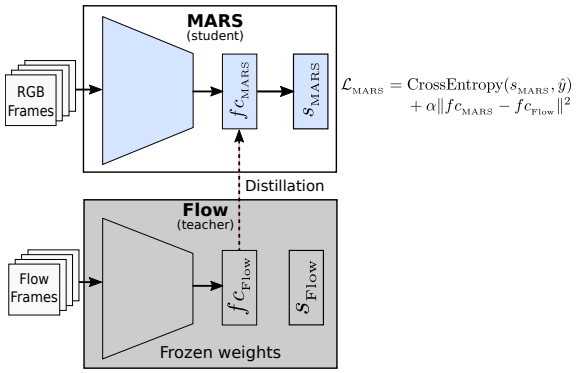


Figure 3: Training to leverage motion and appearance information. Initially, we train the Flow stream to classify actions using optical flow clips with cross entropy loss and freeze its weights. Our approach, MARS, leverages both motion and appearance information by backpropagating the cross entropy loss in addition to MSE loss between features, through all the layers of the network.

computation can be avoided at test time. By combining the standard RGB stream with our new stream (MERS) based only on RGB inputs, we obtain accuracy comparable to a two-stream approach (RGB+Flow), but at a significantly lower computational cost (see MERS and RGB+MERS *vs.* Flow and RGB+Flow in Figure 1).

We then go beyond mimicking flow features, and propose to combine appearance and motion information, effectively into a single stream. This is achieved by training a standard 3D CNN with RGB inputs to minimize the difference in features compared to the Flow stream (as in MERS), as well as to perform action recognition (cross entropy loss, as in a standard RGB stream), see Figure 3. We denote this network by MARS for **M**otion-**A**ugmented **RGB** **S**tream. Experiments highlight that a network trained using our novel approach, performs better than the individual RGB and Flow streams, and is comparable to the two-stream combination (RGB+Flow), with significantly lower computational cost, see Figure 1. This shows that MARS effectively leverages both appearance and motion information. Specifically, MARS obtains 72.7% accuracy on Kinetics compared to 72.0% and 65.6% for RGB and Flow respectively. Similarly on HMDB51 (split-1), MARS obtains 80.1% accuracy, compared to 73.5% and 75.9% for RGB and Flow stream, respectively. Code and models are available at http://www.europe.naverlabs.com/Research/Computer-Vision/Video-Analysis/MARS.

2. Related Work

Significant progress has been made with CNNs for image-based tasks like classification [13, 19], segmentation [3, 23] and object detection [12, 28], especially since 2012. Their impact on problems in the video domain was

not remarkable initially due to the inability of early models to capture temporal variations in video, and the lack of large video datasets. We discuss strategies developed to address this, *e.g*. two-stream architectures and new datasets, in the following. We also summarize recent work on distillation methods to transfer knowledge between two networks, as it is related to our proposed approach with transfer between RGB and Flow streams.

Two-stream networks. Simonyan and Zisserman [32] proposed a two-stream 2D CNN architecture, where one stream operates on RGB frames, and the other on optical flow. The two streams are trained to estimate action class labels, and the final label is obtained by averaging the scores of both the streams. Feichtenhofer *et al*. [8] presented improvements to this two-stream network with different strategies to fuse the two streams. Initial work in this two-stream paradigm was focused on 2D CNNs, but a transition to 3D CNNs was made since spatio-temporal features are better learned with 3D CNNs compared to their 2D equivalents [37]. This transition comes at: (i) a high computational cost, largely due a large number of parameters that need to be optimized [37, 38, 44], and (ii) the problem of overfitting due to small datasets. To deal with the problem of overfitting, Carreira and Zisserman [2] introduced the Kinetics dataset [18], which was large enough to successfully train 3D CNNs [11]. Using RGB and Flow streams pretrained on Kinetics, I3D [2] achieved the state of art on the HMDB51 [20] and UCF101 [33] datasets. Methods such as [38, 44] replaced 3D convolutions with separate spatial and temporal convolutions, which significantly reduces the number of parameters to learn, to alleviate the issues of computational cost. Irrespective of their differences, all the methods discussed above use hand-crafted optical flow features such as [1, 45] for the motion stream, creating a bottleneck for fast and online inference. Such improvements when combining the two streams suggest that despite spatio-temporal convolutions, the appearance stream fails to fully capture the information from the motion stream. In contrast, our approach avoids flow computation at test time, while maintaining the state-of-the-art performance of the two-stream framework.

Joint appearance and motion modeling. CNNs have also been used to estimate optical flow directly from RGB frames [6, 16, 27], instead of relying on hand-crafted optical flow methods, typically with an encoder-decoder structure. This makes the Flow stream end-to-end trainable using a loss on optical flow to guide the flow component, *i.e*., learn optical flow from RGB frames, and then recognize actions with the estimated optical flow. One alternative to this loss, which requires ground-truth optical flow, is using an unsupervised loss to train on action datasets where this ground-truth is unavailable [49]. Other approaches, such as [4, 26] used an off-the-shell optical flow method ([1] in [4], EpicFlow [29] in [26]) as pseudo ground-truth flow. Fan *et al*. [7] integrate TV-L1 flow [45] in the form of a differentiable module into a CNN. Instead of learning the flow module from scratch, they initialize it with a model learned on optical flow benchmarks, and only train the network for action recognition, without any loss on flow. As expected, the output of the flow module learned in this fashion no longer corresponds to an accurate optical flow. In contrast to these works, we propose to compute a loss on the features from the motion stream, thereby allowing the network to mimic or enhance the Flow stream.

Very recent works have attempted to model appearance and motion into a single stream [21, 35], with modules designed to better exploit temporal information, leading to complex architectures for a modest gain. We show that our strategy of minimizing a feature-based loss provides an effective way to integrate temporal information into a standard 3D convolutional architecture.

Distillation. Our proposed learning approach is related to the concept of generalized distillation [24] that combines distillation [14] and privileged information [39]. Distillation was originally proposed for knowledge transfer from a complex to a simple model by using class probabilities of the complex model as 'soft target' for the smaller one [14]. In a similar spirit, our goal is to transfer knowledge from the motion stream to a network with only RGB input, without explicit flow computation. The learning under privileged information paradigm provides a model trained with additional information available only in the training phase and not at test time [39]. In our case, flow is the privileged information available for training, along with RGB, but only RGB is available at test time.

Garcia *et al*. [9] developed a distillation framework for action recognition with their four-step process that hallucinates depth features from RGB frames. They distilled depth features via logits, as well as matching feature maps of depth and RGB networks. In a similar spirit, Hoffmann *et al*. [15] hallucinate depth information for an object detector by combining different losses between mid-level features with standard object detection losses. Another recently proposed graph distillation approach [25] dynamically leverages information across different modalities. Our method differs from these works as: (a) we consider the case of RGB and Flow inputs, and (b) distill knowledge from the Flow to the RGB stream by matching high-level features, instead of matching class probabilities (logits).

3. Learning to Replace Flow

The state of the art for action recognition leverages both appearance (RGB) and motion (Flow) streams [2]. These streams have standard image architectures with 3D convolutions instead of 2D convolutions, and take clips of a fixed length as input. Given a video clip of consecutive frames

of an action $\hat{y}$, the RGB and Flow streams are trained separately to classify actions. Let s_{RGB} (resp. s_{Flow}) denote the score computed by the RGB (resp. Flow) network before softmax, and y_{RGB} (resp. y_{Flow}) the predicted class, *i.e.*, the one with the highest score. The prediction at test time is usually obtained by averaging s_{RGB} and s_{Flow}.

We now address the challenge of avoiding flow computation at test time, while achieving similar performance as two-stream network. To this end, we propose a solution based on the concept of learning under privileged information [39]. We consider the Flow stream, operating on flow clips, as a teacher network that possesses vital information needed for action recognition. Our goal is to train a second network (student) to classify actions using RGB frames as input, along with the privileged information from the teacher, *i.e.*, Flow, that is supplied only at training time. In the following, we assume that the Flow stream is already trained for action recognition and we freeze its weights. We now detail our two learning strategies: (i) to mimic flow features using RGB frames (Section 3.1), and (ii) to leverage both appearance and motion information (Section 3.2).

3.1. MERS

Our first training strategy to hallucinate flow features from RGB input is denoted as **M**otion **E**mulating **RGB** **S**tream (MERS). We achieve this by imposing a loss function at the feature level. Initial layers of a CNN represent low-level local features, while the latter layers represent high-level global features [46], which are highly discriminative [31] for the concerned task. We thus use a loss on the output of the layer immediately before the final fully-connected layer of MERS, to mimic those of the Flow stream. We denote these features from MERS and Flow streams as fc_{MERS} and fc_{Flow}, respectively. Figure 2 illustrates the training strategy for MERS. MERS has a similar architecture and inputs as a standard RGB stream with 3D convolutions, but its target is to reduce the Mean Squared Error (MSE) loss between these features:

$$\mathcal{L}_{\text{MERS}} = \|fc_{\text{MERS}} - fc_{\text{Flow}}\|^2. \quad (1)$$

Applying this loss at the penultimate layer of the network leaves the last layer of MERS untrained. We follow a two-step training procedure, where we first train all the layers of MERS, except the last one, using the mean squared loss (1). This training provides a stream that mimics the features of the Flow stream. For performing action recognition, we train (Step 2 in the figure) only the last fully-connected layer, *i.e.*, the classifier, separately, with a cross entropy loss using these "mimicked" features.

In summary, we first train the Flow stream to classify actions using optical flow clips with cross entropy loss between the true class labels $\hat{y}$ and the predicted class labels y_{Flow}. Once the Flow stream is trained, we freeze its weights. We then train MERS to mimic the Flow stream using RGB frames by backpropagating the MSE loss between fc_{MERS} and fc_{Flow} through the first $n-1$ layers of a n layered network. These hallucinated flow features are finally used for action classification by training the n^{th} layer of MERS, with a cross-entropy loss between the true class, $\hat{y}$ and the class predicted through score s_{MERS}. Note that the cross entropy loss is backpropagated only through the last layer of MERS. At test time, MERS is independent of the Flow stream, and only RGB input is necessary.

3.2. MARS

Our second strategy goes a step further: we train a network that leverages both appearance and motion information with only RGB inputs at test time, and without explicit flow estimation. We refer to this as **M**otion-**A**ugmented **RGB** **S**tream (MARS). Recall that MERS uses the MSE loss to distill motion information into a network operating on RGB frames. To enhance this training with appearance information, we train the network by backpropagating a linear combination of MSE and cross entropy losses through the entire network. In other words, we train MARS using the following loss function:

$$\mathcal{L}_{\text{MARS}} = \text{CrossEntropy}(s_{\text{MARS}}, \hat{y}) + \alpha \|fc_{\text{MARS}} - fc_{\text{Flow}}\|^2, \quad (2)$$

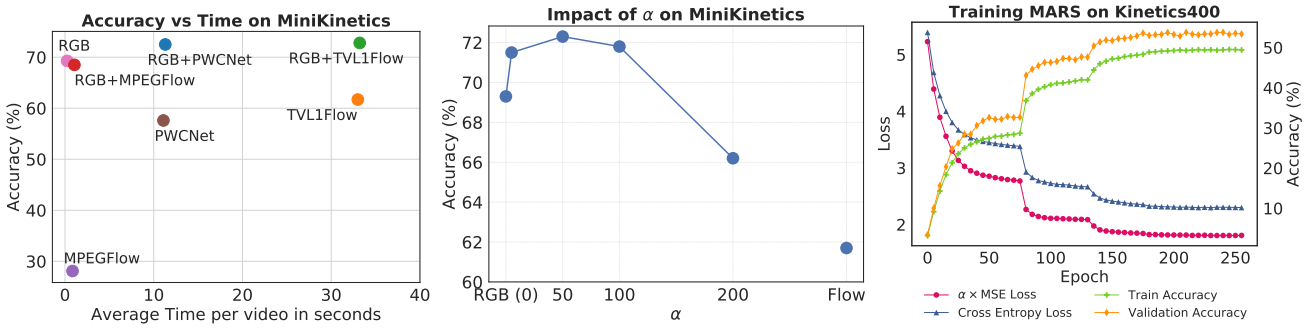
where α is a scalar weight modulating the influence of motion features. Smaller values of α makes MARS similar to a standard RGB stream, and larger ones drive it closer to MERS that mimics the Flow stream. We study the impact of α in Section 5.3. Using this combined loss ensures that a difference between the mimicked and flow features leads to a decrease in cross-entropy, *i.e.*, a higher classification accuracy. As the stream is based on RGB data, this feature difference comes from appearance. Thus, MARS effectively combines motion information distilled from the Flow stream with complementary appearance information necessary for better action classification.

To sum up our strategy MARS, we first train the Flow stream, with standard cross entropy loss. We then freeze its weights and train MARS; see Figure 3. When testing with MARS, we only use RGB frames as input to compute the class scores, thus avoiding flow computation.

4. Experimental Setup

4.1. Datasets and metrics

We focus on the popular benchmarks for action recognition: Kinetics400 [18], HMDB51 [20], UCF101 [33], and SomethingSomethingv1 [10]. Kinetics400 consists of 400 classes with approximately 240k training, 20k validation and 40k test videos. As this dataset is large, we perform some of the analyses on the MiniKinetics subset containing 200 classes, 80k training and 5k validation videos introduced by [44]. HMDB51 consists of 51 action classes



(a) (b) (c)

Figure 4: (a) Accuracy *vs*. time on MiniKinetics for different optical flow approaches using 16-frame clips. Time is averaged over all videos of the MiniKinetics validation set. (b) Accuracy of MARS for different values of α on MiniKinetics with 16-frame clips. (c) Evolution of losses and accuracies while training MARS on Kinetics400 from scratch for $\alpha = 50$.

with a total of 7000 videos and three different train/test splits. UCF101 contains 101 actions classes with a total of 13,320 videos and also three train/test splits. We denote the first split for HMDB51 and UCF101 as HMDB51-1 and UCF101-1, respectively. SomethingSomethingv1 contains a total of 174 action classes with 86,017 training, 11,522 validation and 10,960 test videos.

Metrics. For all the datasets we report top-1 mean accuracy. For Kinetics400 and SomethingSomethingv1, we report performance on the validation set as the test servers are not available anymore.

4.2. Implementation details

RGB and flow inputs. We extract frames at 25 fps and resize them, such that the smallest dimension is 256 pixels, except for SomethingSomethingv1 where frames are provided at 12 fps with a frame height of 100 pixels [10]. Following recent approaches [2, 44], we use the TV-L1 method [45] to extract optical flow, with default parameter setting from OpenCV.[1] We truncate the values to lie between -20 and 20, map them to the $[0, 255]$ range, and then save them with jpeg compression. Following [11], most experiments are done with clips of 16 consecutive frames (16f-clip) to maintain reasonable training time. We also experiment with 64 frames clips (64f-clip) with the same data augmentation. At training, we randomly sample a 112×112 crop in the image from a random clip of the given length, and randomly apply horizontal flipping, which includes reverting the x-direction in the case of flow input. We subtract the ActivityNet mean for RGB inputs and 127.5 for optical flow, *i.e*., we assume flow is centered at 0. At test time, we use center crop and average scores of all non-overlapping clips. When combining multiple streams, we average the scores of each stream.

We also evaluate the performance of two computationally efficient flow approaches, MPEGFlow [17] and PWC-Net [34] on MiniKinetics. MPEGFlow corresponds to the motion vector encoded in the MPEG video compression format that can be retrieved at nearly zero computational cost. We extract MPEGFlow from videos using macroblocks of size 8×8, and resize these flow frames such that the smaller dimension is 256 pixels. PWC-Net is a recent CNN-based approach that operates about four times faster than its competitors, while maintaining a reasonable endpoint error. For PWC-Net, input frame dimensions are resized to a multiple of 64 and pixels values are normalized between $[0, 1]$. For both these approaches, we follow the same scaling and data augmentation procedure as that of TV-L1.

Architecture and training. We choose the 3D ResNeXt-101 [43] architecture due its performance on Kinetics400, UCF101, and HMDB51 [11]. Following the setting of [11], we use the SGD optimization method with a weight decay of 0.0005, momentum of 0.9, and an initial learning rate of 0.1, except when training flow with 64f-clips, where we use 0.01. While training MARS, we choose $\alpha = 50$ based on the experimental results detailed in Section 5.3. We train on Kinetics400 and MiniKinetics from scratch. For other datasets, we finetune from the model trained on Kinetics400: all the layers in the case of SomethingSomethingv1, and only the last block and the last fully-connected layer for the smaller HMDB51 and UCF101 datasets.

5. Results and Discussion

We begin with an analysis of the impact of flow in two-stream approaches in Section 5.1. Section 5.2 compares the performance of MERS and MARS training strategies and shows that MARS outperforms both RGB and Flow. Next, we present an extensive study of our network with the influence of the weighing factor in the loss equation (Section 5.3), and the impact of motion (Section 5.4). We finally compare with the state of the art in Section 5.5.

5.1. Flow stream

We first evaluate the performance of three optical flow approaches, namely TV-L1, PWC-Net, and MPEGFlow,

[1] https://opencv.org

Stream	MiniKinetics	Kinetics400	UCF101-1	HMDB51-1	Something Somethingv1
RGB	69.3	68.2	91.7	66.7	30.2
Flow	61.7	54.0	92.5	71.4	34.5
RGB+Flow	72.7	69.1	95.6	74.0	38.8
MERS	62.2	54.3	93.4	71.8	35.5
MERS+RGB	72.3	68.3	95.6	72.9	38.4
MERS+Flow	63.3	55.0	93.4	72.4	36.4
MERS+RGB+Flow	72.2	67.0	95.5	74.5	39.4
MARS	72.3	65.2	94.6	72.3	39.6
MARS+RGB	72.8	69.6	95.6	73.1	37.6
MARS+Flow	71.3	62.8	94.9	74.5	39.2
MARS+RGB+Flow	73.5	68.9	95.8	75.0	40.4

Table 1: Top-1 accuracy using 16f-clips. For MiniKinetics and Kinetics400, all the streams are trained from scratch. For UCF101-1, HMDB51-1 and SomethingSomethingv1, all the streams are finetuned from Kinetics400 pretrained models. Optical flows are computed using TV-L1 algorithm.

Stream	Kinetics400	UCF101-1	HMDB51-1	Something Somethingv1
RGB	72.0	95.2	73.5	46.6
Flow	65.6	95.8	75.9	43.0
RGB+Flow	74.5	97.5	79.8	51.7
MARS	72.7	97.1	80.1	48.7
MARS+RGB	74.8	97.3	80.6	51.7
MARS+Flow	72.3	97.5	80.9	50.4
MARS+RGB+Flow	74.9	97.8	81.3	53.0

Table 2: Top-1 accuracy using 64f-clips. For Kinetics400, all the streams are trained from scratch. For UCF101-1, HMDB51-1 and SomethingSomethingv1, all the streams are finetuned from Kinetics400 pretrained models. Optical flows are computed using TV-L1 algorithm.

on MiniKinetics. Figure 4a shows the accuracy *vs*. average time per video on the MiniKinetics validation set (250 frames per video on average). All the times quoted exclude data access time, and are computed on the same machine with an NVIDIA TitanX GPU. We observe that the TV-L1 Flow stream (orange point in the figure) produces the best accuracy, although with an extremely high computation cost of over 30s per video, which represents nearly 99% of the total time for recognizing actions. As seen in Figure 4a, MPEGFlow is the fastest to compute, by a large margin, requiring less than a second per video, but its performance is significantly lower than that of TV-L1, and all the other flow methods [40]. PWC-Net is about three times faster than TV-L1, but its performance is 5% lower than TV-L1. These observations motivate our effort to mimic the Flow stream with the help of an accurate flow method, like TV-L1, and to avoid estimating optical flow at test time. For the sake of completion, we illustrate the performance of two-stream networks with RGB for each of these three flow methods and observe an increase in accuracy, except for RGB+MPEGFlow (see Figure 4a). MPEGFlow has poor quality and consequently, its accuracy is 40% lower than RGB, which leads to the degraded performance of RGB+MPEGFlow.

Tables 1 and 2 show the impact of TV-L1 optical flow [45] in a two-stream framework with 16 and 64 frame clips, respectively. The first two rows in each table correspond to either RGB or Flow stream alone. Compared to these, the RGB+Flow two-stream variant shows a significant improvement in performance. For example, in the 64-clip case, we observe an increase of 2.5%, 2.3%, 6.3% and 5.1% in accuracy over RGB on Kinetics400, UCF101-1, HMDB51-1, and SomethingSomethingv1, respectively.

5.2. Recognition accuracy

MERS. We now evaluate our training strategy MERS, whose goal is to mimic the Flow stream from RGB inputs, to avoid flow computation at test time. From the results shown in Table 1, we observe that the difference in mean accuracy between MERS and Flow is less than 1% on all the datasets. This shows that MERS is a good replacement for the Flow stream. This observation is further supported by the minimal difference between MERS+RGB and RGB+Flow. Also, combining Flow with MERS, results in almost no performance improvement. In summary, MERS provides an effective alternative to the Flow stream.

MARS. Table 1 also illustrates the performance of our training strategy combining feature-based and cross entropy losses (MARS). All the results correspond to the $\alpha = 50$ setting. We observe that MARS outperforms both RGB and Flow streams alone on MiniKinetics, UCF101, HMDB51, and SomethingSomethingv1 by a substantial margin (between 1% to 9%), showing that MARS learns to leverage both appearance and motion information, effectively. However, on Kinetics400, MARS performs worse than the RGB stream. This is due to the poor performance of the Flow stream with short clips on this dataset, with a 14% difference over RGB, as many videos are mostly static in a

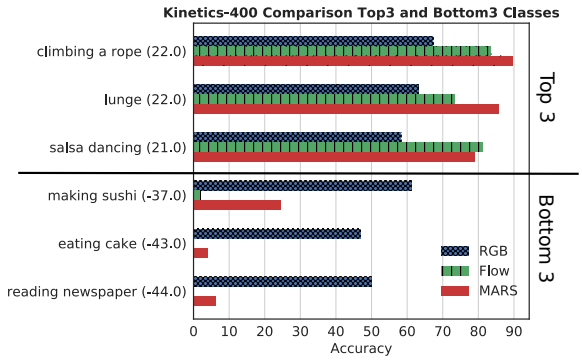


Figure 5: Class-wise performance of MARS *vs*. RGB and Flow for 6 classes of Kinetics400 dataset.

short range. Li *et al*. [22] also find that Kinetics400 is biased towards static information. When using longer clips of 64 frames, see Table 2, the difference between RGB and Flow streams is significantly lower (6.4%), and MARS performs better than both RGB and Flow streams. On all the other datasets, MARS obtains a significantly higher accuracy than RGB and Flow streams, specifically on datasets where motion is crucial, like SomethingSomethingv1 or HMDB51. In some cases, *e.g*., on HMDB51 with 64f-clips, MARS alone performs better than the two-stream RGB+Flow model. MARS+Flow and MARS perform similarly, indicating that MARS successfully leverages motion information, *i.e*., adding Flow does not further improve performance.

To further understand when MARS performs better than RGB, we compare the performance of RGB, Flow and MARS for three classes with the highest and lowest difference (shown as the number in parenthesis in the figure for each case) between MARS and RGB accuracy in Figure 5. MARS has the strongest impact for classes where Flow performs well. On the other hand, when Flow results in low performance, MARS also performs quite poorly compared to RGB, but remains better than Flow. This again shows that MARS leverages information from both appearance and motion data.

5.3. Effect of α on accuracy

MARS is trained by balancing two losses: (a) cross-entropy loss between logits and ground-truth targets, and (b) MSE loss between averaged pooling features of MARS and Flow, see (2). We report mean accuracy of MARS on MiniKinetics using values of $\alpha = \{5, 50, 100, 200\}$ in Figure 4b for 16f-clips. We also report the accuracy of RGB (*i.e*., $\alpha = 0$) and Flow (on the right). We observe that by increasing the value α, we reach a peak accuracy ($\alpha = 50$) where the values of the cross-entropy loss and $\alpha\times$MSE loss are approximately the same, see Figure 4c. This shows that MARS not only achieves a trade-off between RGB and Flow, but effectively leverages both motion and appearance. Higher values of α increase the influence of MSE over cross-entropy loss, thus causing MARS to tend towards the Flow stream accuracy, which is essentially MERS.

We also experimented applying the feature-based loss on earlier layers or on the logits. We find that applying this loss on the earlier layers leads to a drop in accuracy, as it is more difficult to mimic lower-level flow features from RGB inputs. When this feature-based loss is applied on the logits, we obtain a similar performance on MiniKinetics, as when the loss is applied to high-level features. However, a model pretrained on Kinetics400 does not generalize well enough to other datasets.

5.4. Impact of motion

To further understand the difference in the features learned by MARS and MERS compared to those of RGB and Flow streams, we analyze their performance in the absence of motion. We replace the actual test clips of MiniKinetics by 'static' clips that are created by duplicating the middle frame of each clip, thus removing motion information. The mean accuracy, reported at the top of Figure 6, is calculated by averaging scores over such non-overlapping consecutive 'static' clips, consisting of 16 frames.

Figure 6 also shows the class activation maps [48] for each of the streams. Class activation maps help visualize discriminative regions specific to each action class. We feed the networks with 'static' 16f-clips and observe that the accuracy of RGB drops slightly with static clips and the class activations are relevant. This shows that RGB mainly focuses on appearance despite 3D convolutions. MARS shows a larger drop than RGB due to lack of motion information, but can localize the relevant regions correctly. This shows that MARS simultaneously captures appearance and motion. MERS and Flow perform close to random on the static clips (classification accuracy 0.5% and 5.1% respectively, and random visualization regions). This is expected in the absence of motion and illustrates that they behave similarly.

5.5. Comparison with the state of the art

We now compare the performance of MARS with state-of-the-art approaches in Table 3 for Kinetics400 and in Table 4 for UCF101, HMDB51 and SomethingSomethingv1. In both tables, we first compare with methods that use only RGB as input at test time, without explicit flow computation, and then with approaches that use both RGB and Flow. For Kinetics400, when using only RGB frames as input, MARS+RGB performs better than all the methods, except NL-I3D [42]. Note that NL-I3D is pretrained on ImageNet using clips of size $128 \times 224 \times 224$, *i.e*., 2 times longer and 4 times higher resolution than our clip size used to train MARS from scratch. This approach is based on a

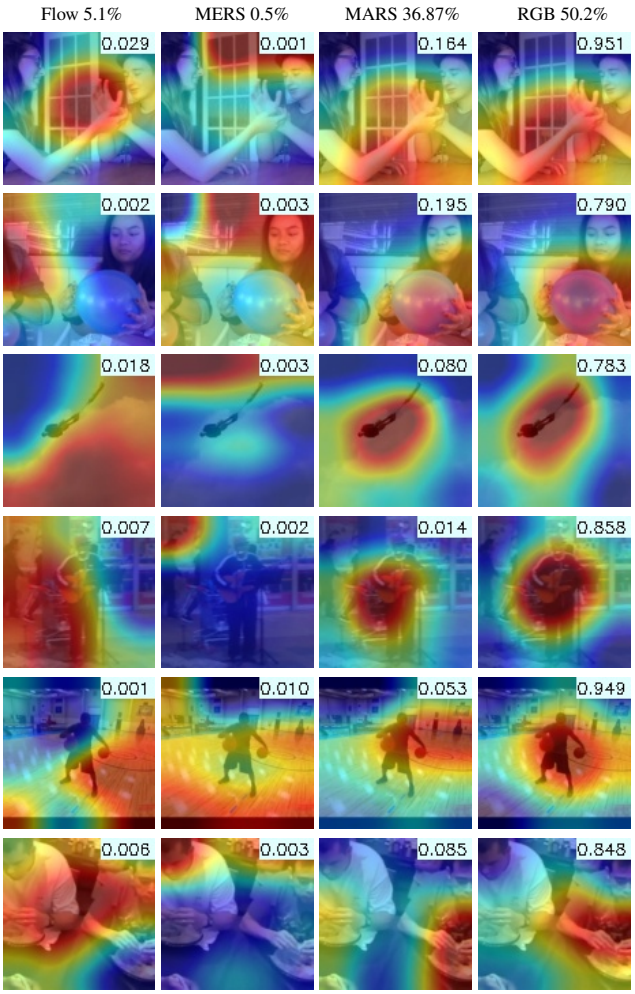


Figure 6: From left to right: class activation maps of Flow, MERS, MARS and RGB using 16f 'static' clips on MiniKinetics. Numbers right at the top indicate video classification accuracy on the validation set. Numbers on each map indicate the softmax activation scores of the ground truth class.

novel non-local module to capture long-range dependency, which can further be integrated into our architecture. In Table 4, MARS+RGB outperforms all the methods that use RGB only as inputs. In particular, the gap is quite significant for datasets where motion is important, as HMDB51 and SomethingSomethingv1, with an improvement of 3.6% and 3.5% respectively. This shows that our approach for mimicking flow features along with action recognition generalizes well to other datasets. Next, we compare the accuracy of MARS+RGB+Flow to state-of-the-art methods that use both RGB and Flow as inputs. On Kinetics400, adding Flow to MARS+RGB only has a marginal impact (increase of 0.1), as motion is not important. In contrast, on HMDB51 and SomethingSomethingv1, the gain is larger, around 1%, and we set a new state-of-the-art performance.

Method	Streams	Pretrain	Acc
I3D [2]	RGB	ImageNet	71.1*
ResNext101 [11]	RGB	none	65.1
R(2+1)D [38]	RGB	Sport-1M	74.3
S3D-G [44]	RGB	ImageNet	74.7
NL-I3D [42]	RGB	ImageNet	**77.7**
MARS	RGB	none	72.7
MARS+RGB	RGB	none	74.8
I3D [2]	RGB+Flow	ImageNet	74.2*
R(2+1)D [38]	RGB+Flow	Sports-1M	75.4
S3D-G [44]	RGB+Flow	ImageNet	77.2
MARS+RGB+Flow	RGB+Flow	none	74.9

Table 3: Comparison with state-of-the-art Top-1 accuracy results for Kinetics400 validation set. (*Calculated on the held-out test set of Kinetics400)

Method	Streams	Pretrain	UCF101	HMDB51	Something Somethingv1
TRN [47]	RGB	none	—	—	34.4
MFNet [21]	RGB	none	—	—	43.9
C3D [37]	RGB	Sports-1M	90.4	—	—
I3D [2]	RGB	ImNet+Kin	95.6	74.8	—
ResNext101 [11]	RGB	Kinetics	94.5	70.1	—-
S3D-G [44]	RGB	ImNet+Kin	96.8	75.9	48.2*
R(2+1)D [38]	RGB	Kinetics	96.8	74.5	—
MARS	RGB	Kinetics	97.4	79.3	48.7
MARS+RGB	RGB	Kinetics	**97.6**	**79.5**	**51.7**
2-stream [32]	RGB+Flow	ImageNet	88.0	59.4	—
TSN [41]	RGB+Flow	ImageNet	94.2	69.4	—
TRN [47]	RGB+Flow	none	—	—	42.0
I3D [2]	RGB+Flow	ImNet+Kin	98.0	80.7	—
R(2+1)D [38]	RGB+Flow	Kinetics	97.3	78.7	—
OFF [35]	RGB+Flow	none	96.0	74.2	—
MARS+RGB+ Flow	RGB+Flow	Kinetics	**98.1**	**80.9**	**53.0**

Table 4: Comparison with state-of-the-art results. The results of UCF101 and HMDB51 are averaged over 3 splits. For SomethingSomethingv1, the numbers represent validation set accuracy. (*pretrained on ImageNet)

6. Conclusion

In this paper, we introduced MARS, a strategy to learn a stream that takes only RGB frames as input but leverages both appearance and motion information from them. This is achieved by training a network to minimize the loss between its features and the Flow stream, along with the cross entropy loss for recognition. Our extensive evaluation showed that our single-stream MARS framework outperforms RGB and Flow streams on popular benchmarks, such as Kinetics400, UCF101, HMDB51, and SomethingSomethingv1.

Acknowledgments. This work was supported in part by ERC advanced grant Allegro and a grant from ANR (AVENUE project ANR-18-CE23-0011).

References

[1] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004. 3

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 1, 3, 5, 8

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans. PAMI*, 2018. 2

[4] Ali Diba, Ali Mohammad Pazandeh, and Luc Van Gool. Efficient two-stream motion and appearance 3D CNNs for video classification. In *ECCV workshop*, 2016. 3

[5] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 1

[6] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 3

[7] Lijie Fan, Wenbing Huang, Chuang Gan, Stefano Ermon, Boqing Gong, and Junzhou Huang. End-to-end learning of motion representation for video understanding. In *CVPR*, 2018. 3

[8] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 3

[9] Nuno Garcia, Pietro Morerio, and Vittorio Murino. Modality distillation with multiple stream networks for action recognition. In *ECCV*, 2018. 3

[10] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The something something video database for learning and evaluating visual common sense. In *ICCV*, 2017. 1, 4, 5

[11] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet. In *CVPR*, 2018. 3, 5, 8

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 2

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS workshop*, 2014. 2, 3

[15] Judy Hoffman, Saurabh Gupta, and Trevor Darrell. Learning with side information through modality hallucination. In *CVPR*, 2016. 3

[16] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017. 3

[17] Vadim Kantorov and Ivan Laptev. Efficient feature extraction, encoding and classification for action recognition. In *CVPR*, 2014. 5

[18] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 1, 3, 4

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[20] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. HMDB: A large video database for human motion recognition. In *ICCV*, 2011. 3, 4

[21] Myunggi Lee, Seungeui Lee, Sungjoon Son, Gyutae Park, and Nojun Kwak. Motion feature network: Fixed motion filter for action recognition. In *ECCV*, 2018. 3, 8

[22] Yingwei Li, Yi Li, and Nuno Vasconcelos. RESOUND: Towards action recognition without representation bias. In *ECCV*, 2018. 7

[23] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2

[24] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. In *ICLR*, 2016. 3

[25] Zelun Luo, Jun-Ting Hsieh, Lu Jiang, Juan Carlos Niebles, and Li Fei-Fei. Graph distillation for action detection with privileged modalities. In *ECCV*, 2018. 3

[26] Joe Yue-Hei Ng, Jonghyun Choi, Jan Neumann, and Larry S. Davis. ActionFlowNet: Learning motion representation for action recognition. In *WACV*, 2018. 3

[27] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *CVPR*, 2017. 3

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2

[29] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015. 3

[30] Laura Sevilla-Lara, Yiyi Liao, Fatma Guney, Varun Jampani, Andreas Geiger, and Michael J Black. On the integration of optical flow and action recognition. In *GCPR*, 2018. 2

[31] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *CVPR workshops*, 2014. 4

[32] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 1, 3, 8

[33] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 3, 4

[34] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018. 1, 5

[35] Shuyang Sun, Zhanghui Kuang, Lu Sheng, Wanli Ouyang, and Wei Zhang. Optical flow guided feature: a fast and robust motion representation for video action recognition. In *CVPR*, 2018. 3, 8

[36] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1

[37] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, 2015. 1, 3, 8

[38] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018. 1, 3, 8

[39] Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: Similarity control and knowledge transfer. *JMLR*, 2015. 2, 3, 4

[40] Gül Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. *IEEE Trans. PAMI*, 2018. 6

[41] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 8

[42] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 7, 8

[43] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 5

[44] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, 2018. 1, 2, 3, 4, 5, 8

[45] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime TV-L1 optical flow. In *Joint Pattern Recognition Symposium*, 2007. 1, 3, 5, 6

[46] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 4

[47] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *ECCV*, 2018. 8

[48] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016. 7

[49] Yi Zhu, Zhenzhong Lan, Shawn Newsam, and Alexander G Hauptmann. Hidden two-stream convolutional networks for action recognition. In *ACCV*, 2018. 3