Video Modeling with Correlation Networks

Heng Wang Du Tran Lorenzo Torresani Matt Feiszli
Facebook AI
{hengwang,trandu,torresani,mdf}@fb.com

Abstract

Motion is a salient cue to recognize actions in video. Modern action recognition models leverage motion information either explicitly by using optical flow as input or implicitly by means of 3D convolutional filters that simultaneously capture appearance and motion information. This paper proposes an alternative approach based on a learnable correlation operator that can be used to establish frame-to-frame matches over convolutional feature maps in the different layers of the network. The proposed architecture enables the fusion of this explicit temporal matching information with traditional appearance cues captured by 2D convolution. Our correlation network compares favorably with widely-used 3D CNNs for video modeling, and achieves competitive results over the prominent two-stream network while being much faster to train. We empirically demonstrate that correlation networks produce strong results on a variety of video datasets, and outperform the state of the art on four popular benchmarks for action recognition: Kinetics, Something-Something, Diving48 and Sports1M.

1. Introduction

After the breakthrough of AlexNet [29] on ImageNet [7], convolutional neural networks (CNNs) have become the dominant model for still-image classification [32, 46, 51, 20]. In the video domain, CNNs were initially adopted as image-based feature extractor on individual frames of the video [26]. More recently, CNNs for video analysis have been extended with the capability of capturing not only appearance information contained in individual frames but also motion information extracted from the temporal dimension of the image sequence. This is usually achieved by one of two possible mechanisms. One strategy involves the use of a two-stream network [45, 56, 15, 57, 41, 5] where one stream operates on RGB frames to model appearance information and the other stream extracts motion features from optical flow provided as input. The representations obtained from these two distinct inputs are then fused, typically in a late layer of the network. An alternative strategy is to use 3D convolutions [1, 24, 52, 49, 54, 40, 62, 9] which couple appearance and temporal modeling by means of spatiotemporal kernels.

In this paper we propose a new scheme based on a novel correlation operator inspired by the correlation layer in FlowNet [11]. While in FlowNet the correlation layer is only applied once to convert the video information from the RGB pixel space to the motion displacement space, we propose a learnable correlation operator to establish frame-to-frame matches over convolutional feature maps to capture different notions of similarity in different layers of the network. Similarly to two-stream models, our model enables the fusion of *explicit* motion cues with appearance information. However, while in two-stream models the motion and appearance subnets are disjointly learned and fused only in a late layer of the model, our network enables the efficient integration of appearance and motion information throughout the network. Compared to 3D CNNs, which extract spatiotemporal features, our model factorizes the computation of appearance and motion, and learns distinct filters capturing different measures of patch similarity. The learned filters can match pixels moving in different directions. Through our extensive experiments on four action recognition datasets (Kinetics, Something-Something, Diving48 and Sports1M), we demonstrate that our correlation network compares favorably with widely-used 3D CNNs for video modeling, and achieves competitive results over the prominent two-stream network while being much faster to train. We summarize our contributions as follows:

- A new correlation operator with learnable filters. By making use of dilation and grouping, the operator is highly efficient to compute. Compared to 3D convolution or optical flow, it provides an alternative way to model temporal information in video.
- A new correlation network which is designed to integrate motion and appearance information in every block. A rigorous study of the new architecture and comparisons with strong baselines provide insights for the different design choices.
- Our correlation network outperforms the state-of-the-art on four different video datasets without using optical

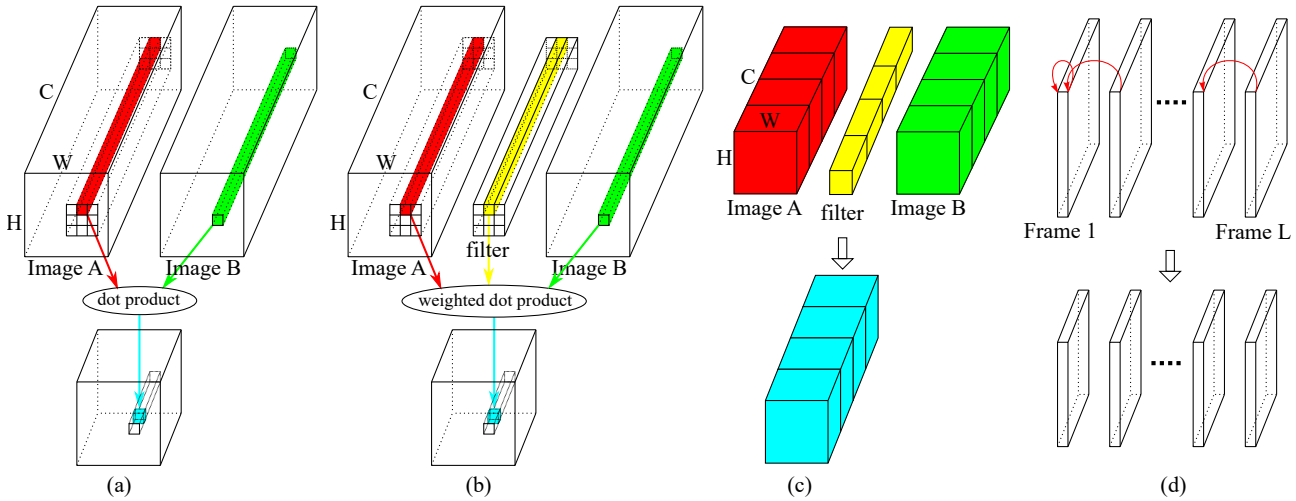


Figure 1: An illustration of the proposed correlation operator. (a) Correlation operator used for optical flow and geometric matching. (b) The introduction of filters renders the operator "learnable." (c) Groupwise correlation increases the number of output channels without adding computational cost. (d) Extending the correlation operator to work on a sequence of video frames.

flow.

In the rest of the paper, we introduce related work in Section 2, and detail the proposed correlation operator in Section 3. We present the correlation network in Section 4. Experimental setups are in Section 5. We discuss the experimental results in Section 6 and conclude the paper in Section 7.

2. Related Work

Architectures for video classification. Among the popular video models, there are two major categories: two-stream networks [45, 56, 15, 57, 41, 5] and 3D CNNs [1, 24, 52, 49, 54, 40, 62, 9]. Since the introduction of two-stream networks [45], further improvements have been achieved by adding connections between the two streams [15], or inflating a 2D model to 3D [4]. 3D CNNs [1, 24, 52] learn appearance and motion information simultaneously by convolving 3D filters in space and time. Successful image architectures [46, 51, 20] have been extended to video using 3D convolution [4, 52, 62]. Recent research [49, 54, 40, 41] shows that decomposing 3D convolution into 2D spatial convolution and 1D temporal convolution leads to better performance. Our correlation network goes beyond two-stream networks and 3D convolution, and we propose a new operator that can better learn the temporal dynamics of video sequences.

Motion information for action recognition. Before the popularity of deep learning, various video features [31, 44, 28, 10, 55] were hand-designed to encode motion information in video. Besides two-stream networks and 3D CNNs, ActionFlowNet [38] proposes to jointly estimate optical flow and recognize actions in one network. Fan *et al*. [12] and Piergiovanni *et al*. [39] also introduced networks to learn optical flow end-to-end for action recognition.

There is also work [50, 33, 21] seeking alternatives to optical flow. Sun *et al*. [50] extracted features guided by optical flow to capture the transformation between adjacent frames. Lee *et al*. [33] designed motion filters by computing the difference of adjacent frames. Hommos *et al*. [21] proposed to use phase instead of optical flow as the motion representation for action recognition. Our paper is along the line of designing architectures to directly learn motion information from raw RGB pixels.

Applications of correlation operation. Deep matching [60] computes the correlation of image patches to find dense correspondence to improve optical flow. Unlike deep matching using hand-crafted features, FlowNet [11] is a network, where a correlation layer performs multiplicative patch comparisons. Correlation layers were also used in other CNN-based optical flow algorithms [48, 23]. Besides optical flow, Rocco *et al*. [43] used it to estimate the geometric transformation of two images, whereas Feichtenhofer *et al*. [16] applied it to object tracking.

In the context of action recognition, Zhao *et al*. [66] utilize the correlation layer to compute a cost volume to estimate the displacement map as in optical flow. The Spatio-Temporal Channel Correlation Network [8] adapts the Squeeze-and-Excitation block [22] to a ResNeXt [61] backbone. The notion of correlation in [8] refers to the relationship among the spatial and temporal dimensions of the feature maps, which is different from the matching of ad-

Operator	Correlation	3D convolution
Input	$C_{in} \times L \times H \times W$	$C_{in} \times L \times H \times W$
Filter	$L \times C_{in} \times K \times K$	$C_{out} \times C_{in} \times K_t \times K_y \times K_x$
Output	$(G * K * K) \times L \times H \times W$	$C_{out} \times L \times H \times W$
# params	$L * C_{in} * K * K$	$C_{out} * C_{in} * K_t * K_y * K_x$
FLOPs	$C_{in} * K * K * L * H * W$	$C_{out} * C_{in} * K_t * K_y * K_x * L * H * W$

Table 1: A comparison of the correlation operator with 3D convolution. When the size K of the filter is similar (*i.e.*, $K * K \approx K_t * K_y * K_x$), the parameters of 3D convolution is about C_{out}/L times more than the correlation operator, and its FLOPs is about C_{out} times higher.

jacent frames studied in our work. We compare our results with [8] in Section 6.3.

Our paper extends this line of ideas by introducing a learnable operator based on correlation. Instead of trying to explicitly or implicitly estimate optical flow, the correlation operator is used repeatedly in combination with other operators to build a new architecture that can learn appearance and motion information simultaneously and that achieves state of the art accuracy on various video datasets.

3. Correlation Operator

This section describes the proposed correlation operator. We start by reviewing the existing correlation operator over image pairs used in optical flow [11, 48] and geometric matching [60, 43]. We then propose to inject filters into the operator to make it learnable. We discuss how to increase the number of output channels while retaining efficiency and low number of parameters by means of a groupwise variant. We finally generalize the operator to work on sequences of video frames.

Correlation operator for matching. As shown in Figure 1 (a), each image is represented by a 3D tensor of size $C \times H \times W$, where C is the number of channels and $H \times W$ is the spatial resolution. Given a feature patch $P^B(i,j)$ in image B, we compute the similarity of this patch with another patch $P^A(i^{'},j^{'})$ in image A, where (i,j) is the spatial location of the patch. To make the computation more tractable, the size of the feature patch can be reduced to a single pixel, thus $P^A(i^{'},j^{'})$ and $P^B(i,j)$ becomes C-dimensional vectors. The similarity is defined as the dot product of the two vectors:

$$S(i,j,i^{'},j^{'}) = 1/C * \sum_{c=1}^{C} (P_c^B(i,j) * P_c^A(i^{'},j^{'})), \quad (1)$$

where $1/C$ is for normalization. $(i^{'},j^{'})$ is often limited to be within a $K \times K$ neighborhood of (i,j). K is the maximal displacement for patch matching. Considering all possible locations of (i,j) and $(i^{'},j^{'})$ in Eq. 1, the output S is a tensor of size $K \times K \times H \times W$, where $K \times K$ can be flattened to play the role of channel to generate a 3D feature tensor ($K^2 \times H \times W$) like the input image.

Learnable correlation operator. Computer vision has achieved impressive results by moving from hand-crafted features [36, 6] to learnable deep neural networks [29, 20]. The original correlation operator [11, 48, 60, 43] does not include learnable parameters and thus it is quite limited in terms of the types of representations it can generate. We propose to endow the operator with a learnable filter as shown in Figure 1 (b). Our motivation is to learn to select informative channels during matching. To achieve this goal we introduce a weight vector W_c to Eq. 1 in the dot product computation: $W_c * P_c^B(i,j) * P_c^A(i^{'},j^{'})$. The similarity of two feature patches (*i.e.*, $P^B(i,j)$ and $P^A(i^{'},j^{'})$) is often related to how close their spatial location is. We thus apply different weight vectors W_c to different locations in the $K \times K$ neighbor to take into account the spatial distribution of the matching disparity. Thus, the size of each filter is $C \times K \times K$ as summarized in Table 1.

K indicates the maximal displacement when matching two patches. Larger valued K can cover larger regions and encode more information. The downside is that the computational cost grows quadratically w.r.t. K. Inspired by the dilated convolution [63], we propose to perform dilated correlation to handle large displacement without increasing the computational cost. We enlarge the matching region in image A by a dilation factor D. In practice, we set $K = 7$ with a dilation factor of $D = 2$ to cover a region of 13×13 pixels. Besides dilation, we also apply the operator at different spatial scales (as discussed in Section 4), which is a popular strategy to handle large displacements in optical flow [42]. From Figure 4, filters do learn to select discriminative channels as filters from certain channels are more active than the other. Having different weights in the $K \times K$ neighborhood also enables the filter to learn pixel movements in different directions.

Groupwise correlation operator. The correlation operator converts a feature map from $C \times H \times W$ to $K^2 \times H \times W$. In popular CNNs, C can be one to two orders of magnitude larger than K^2. This means that the correlation operator may cause a great reduction in the number of channels.

This is not a problem for applications such as optical flow or geometric matching, where the correlation operator is only applied once. If we want to design a network based on the correlation operator and apply it repeatedly, it will reduce the dimension of the channels dramatically, and degrade the representation power of the learned features, as shown by the results in Section 6.2.

Similar to [19], we propose a groupwise version of the correlation operator that avoids shrinking the number of channels while maintaining efficiency. Groupwise convolution [29, 61] was introduced to reduce the computational cost of convolution by constraining each kernel to span a subset of feature channels. Here we utilize this idea to increase the number of output channels without increasing the computational cost. For the groupwise correlation operator, all C channels are split into G groups for both input images and filters, and the correlation operation is computed within each group. The outputs of all groups are stacked together as shown in Figure 1 (c). This increases the number of output channels by a factor of G, to a total of K^2G channels. The size of each group is $g = C/G$. By choosing the group size properly, we can control the number of channels without additional cost.

From two images to a video clip. The original correlation operator is designed for matching a pair of images. In this paper, we apply it for video classification where the input is a sequence of L video frames. We extend the operator to video by computing correlation for every pair of adjacent frames of the input sequence. As the number of adjacent frame pairs is $L - 1$ (i.e., one fewer than the number of frames), we propose to compute self-correlation for the first frame in addition to the cross-correlation of adjacent frame pairs, shown in Figure 1 (d). It can keep the length L of the output feature map consistent with the input, and make the correlation operator easier to use when designing new architectures. The gradual change of filters within each column of Figure 4 shows filters learn to follow the motion of pixels across frames when extending the correlation operator to a video clip.

Table 1 summarizes our final proposed correlation operator and compares it with the standard 3D convolution. Intuitively, 3D convolution seeks to learn both spatial and temporal representation by convolving a 3D filter in space and time. The correlation operator however, is intentionally designed to capture matching information between adjacent frames. The correlation operator provides an alternative way to model temporal information for video classification, and it has much fewer parameters and FLOPs than the popular 3D convolution.

4. Correlation Network

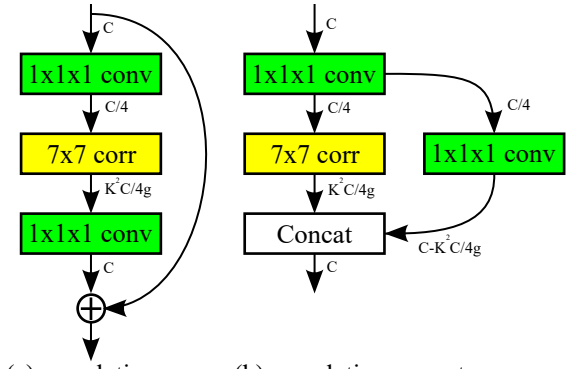
The correlation operator is designed to learn temporal information, and needs to be combined with other operators capturing appearance information in order to yield a comprehensive set of features for video classification. We first briefly introduce the backbone architecture adapted from R(2+1)D [54], then discuss how to build the correlation network to leverage the matching information by incorporating the correlation operator into the backbone.

Layers	R(2+1)D-26		Output size
conv_1	$1\times7\times7$, 64, stride 1,2,2		$\text{L}\times112\times112$
res_2	$\begin{bmatrix} 1\times1\times1, 64 \\ 3\times1\times1, 64 \\ 1\times3\times3, 64 \\ 1\times1\times1, 256 \end{bmatrix}$	$\times2$	$\text{L}\times56\times56$
res_3	$\begin{bmatrix} 1\times1\times1, 128 \\ 3\times1\times1, 128 \\ 1\times3\times3, 128 \\ 1\times1\times1, 512 \end{bmatrix}$	$\times2$	$\text{L}\times28\times28$
res_4	$\begin{bmatrix} 1\times1\times1, 256 \\ 3\times1\times1, 256 \\ 1\times3\times3, 256 \\ 1\times1\times1, 1024 \end{bmatrix}$	$\times2$	$\frac{L}{2}\times14\times14$
res_5	$\begin{bmatrix} 1\times1\times1, 512 \\ 3\times1\times1, 512 \\ 1\times3\times3, 512 \\ 1\times1\times1, 2048 \end{bmatrix}$	$\times2$	$\frac{L}{4}\times7\times7$
global average pool, fc			# classes

Table 2: The R(2+1)D backbone for building correlation network.

R(2+1)D backbone. The R(2+1)D network [54] was recently introduced and shown to yield state-of-the-art action recognition results on several video datasets. R(2+1)D factorizes the traditional 3D convolution (*i.e.*, $3 \times 3 \times 3$) into a 2D spatial convolution (*i.e.*, $1 \times 3 \times 3$) and an 1D temporal convolution (*i.e.*, $3 \times 1 \times 1$). Decoupling the spatial and temporal filtering is beneficial for both hand-crafted features [55, 10] and 3D CNNs [54, 49, 40]. Compared with the original R(2+1)D [54], we make a few changes to further simplify and improve its efficiency, *e.g.*, using bottleneck layers, supporting higher input resolution, keeping the number of channels consistent, less temporal striding, *etc*. Table 2 provides the details of the R(2+1)D backbone used in this paper.

Correlation network. To incorporate the correlation operator into the backbone network, we propose the two types of correlation blocks shown in Figure 2. The design of these blocks is similar in spirit to that of the bottleneck block [20]. Figure 2 (a) illustrates the *correlation-sum* block. It first uses an $1 \times 1 \times 1$ convolution to reduce the number of channels, then applies a correlation operator for feature matching. Finally another $1 \times 1 \times 1$ is used to restore the original number of channels. A shortcut connection [20] is applied



(a) correlation-sum (b) correlation-concat

Figure 2: Two types of correlation blocks. We mark the number of channels for each operator.

for residual learning. The *correlation-concat* block in Figure 2 (b) has two branches within the block: one branch with a correlation operator and another branch passing the input feature maps through an $1 \times 1 \times 1$. The output of the two branches are combined together by concatenation in the channel dimension. We compare the two different designs in Section 6.2.

We obtain the final correlation network by inserting the correlation block into the R(2+1)D backbone architecture. In this paper, we insert one correlation block after res_2, res_3 and res_4 in Table 2. We omit res_5 as its spatial resolution is low (*i.e.*, 7×7). Note that the number of FLOPs of the correlation operator is much lower than 3D convolution. The correlation network only adds a small overhead to the computational cost of the backbone network. Section 6.1 provides a more quantitative analysis.

5. Experimental Setups

Video Datasets. We evaluate our model on four video datasets that have rather different properties, emphasizing distinct aspects of action recognition. **Kinetics** [27] is among the most popular datasets for video classification. It consists of about 300K YouTube videos covering 400 categories. **Something-Something** [18] is created by crowdsourcing. This dataset focuses on humans performing predefined basic actions with everyday objects. The same action is performed with different objects ("something") so that models are forced to understand the basic actions instead of recognizing the objects. It includes about 100K videos covering 174 classes. We note this dataset as Something for short. **Diving48** [34] was recently introduced and includes videos from diving competitions. The dataset is designed to reduce the bias of scene and object context in action recognition, and force the model to focus on understanding temporal dynamics of video data. It has a fine-grained taxonomy covering 48 different types of diving with 18K videos in total. The annotations of **Sports1M** [26] are produced automatically by analyzing the text metadata surrounding the videos. As there are many long videos in Sports1M, we cut them into shorter clips to better utilize the data and end up with a training set of about 5M samples. For Kinetics and Something, annotations on the testing set are not public available, so we report accuracy on the validation set like others. For Diving48 and Sports1M, we report accuracy on the testing set following the setup by the authors [34, 26].

Training and Testing. To train the correlation network, we sample a clip of L (16 or 32) frames with a resolution of 224×224 from a given video. Some videos in Something do not have enough frames. We simply repeat each frame twice for those videos. For data augmentation, we resize the input video to have shorter side randomly sampled in [256, 320] pixels, following [58, 46], and apply temporal jittering when sampling clips for training. For the default configuration of our correlation network, we use the *correlation-sum* block, and set the filter size to $K = 7$ and group size to $g = 32$. Training is done with synchronous distributed SGD on GPU clusters using Caffe2 [3] with a cosine learning rate schedule [35]. We train the model for 250 epochs in total with the first 40 epochs for warm-up [17] on Kinetics. As Something and Diving48 are smaller datasets, we reduce the training epochs from 250 to 150 on them. For Sports1M, we train 500 epochs since it is the largest dataset. For testing, we sample 10 clips uniformly spaced out in the video and average the clip-level predictions to generate the video-level results. Except in Section 6.3, all reported results are obtained by training from scratch without pretraining on ImageNet [7] or other large-scale video datasets. We only use RGB as the input to our model, unlike two-stream networks [45, 56, 15, 57] which use both RGB and optical flow.

6. Experimental Evaluation

To demonstrate the advantages of the proposed correlation network, we first compare the correlation operator with temporal convolution in Section 6.1. We evaluate the correlation network under different settings to justify our design choices and compare with the two-stream network in Section 6.2. We show that our correlation network outperforms the state of the art on all four datasets in Section 6.3. Finally, we visualize the learned filters in Section 6.4.

6.1. Correlation network vs baseline backbones

Table 3 compares the correlation network with different baselines. We denote the backbone architecture from Table 2 as *R(2+1)D-26*. To demonstrate the importance of temporal learning on different datasets, we create *R2D-26*, which is obtained by removing all 1D temporal convolu-

Model	Length	Top-1 accuracy (%) GFLOPs	Kinetics	Something	Diving
R2D-26	16	27.5	67.8	15.8	17.5
R(2+1)D-26	16	36.0	69.9	35.4	22.7
CorrNet-26	16	37.4	73.4	38.5	27.0
R2D-26	32	55.0	70.1	28.1	29.2
R(2+1)D-26	32	71.9	72.3	45.0	32.2
CorrNet-26	32	74.8	**75.1**	**47.4**	**35.5**

Table 3: Correlation networks vs baselines. Our CorrNet significantly outperforms the two baseline architectures on three datasets, at a very small increase in FLOPs compared to R(2+1)D. Using longer clip length L leads to better accuracy on all three datasets.

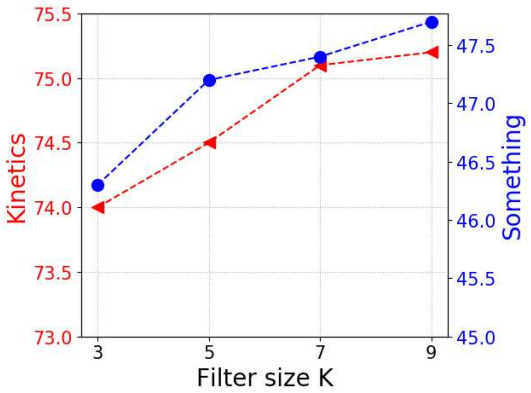


Figure 3: Effect of filter size K on classification accuracy.

tions (*i.e.*, $3 \times 1 \times 1$), and adding a $3 \times 1 \times 1$ max pooling when we need to do temporal striding. *CorrNet-26* is obtained by inserting one *correlation-sum* block after res_2, res_3 and res_4 of R(2+1)D-26 as described in Section 4. As the correlation block adds a small overhead to the FLOPs, we further reduce the number of filters for conv_1 from 64 to 32, and remove the $3 \times 1 \times 1$ temporal convolutions from res_2 for CorrNet. This reduces the accuracy of CorrNet only slightly (less than 0.5%). The resulting CorrNet-26 has similar FLOPs as R(2+1)D-26, as shown in Table 3.

R2D vs R(2+1)D. The gap between R2D and R(2+1)D varies dramatically on different datasets. On Kinetics and Diving48, R(2+1)D is only 2-5% better than R2D, but the gap widens up to 20% on Something. This is consistent with findings in [62] and is due to the design of Something where objects are not predictive of the action label. This also highlights the challenges of designing new architectures that can generalize well to different types of datasets.

R(2+1)D vs CorrNet. We observe a consistent improvement of over 3% on three datasets when comparing CorrNet with R(2+1)D in Table 3. We achieve the most significant gain on Diving48, *i.e.*, 4.3%, using 16 frames. Note that our improved R(2+1)D is a very strong baseline and its performance is already on par with the best results (listed in Table 6 and 7) reported. A significant 3% improvement on three datasets shows the power of the information learned from pixel matching and the general applicability of the correlation network to model video of different characteristics. Moreover, CorrNet only increases the GFLOPs of the network by a very small margin, from 71.9 to 74.8, comparing with R(2+1)D.

Datasets	Kinetics	Something
CorrNet-26	75.1	47.4
w/o filter	73.9	46.5
w/o grouping	74.2	46.1
correlation-concat	73.2	45.9

Table 4: Action recognition accuracy (%) for different configurations of CorrNet.

Datasets	Kinetics	Something
CorrNet-26	75.1	47.4
R(2+1)D-26 (RGB)	72.3	45.0
R(2+1)D-26 (OF)	66.5	42.5
R(2+1)D-26 (Two-stream)	74.4	47.9

Table 5: Action recognition accuracy (%) of CorrNet vs two-stream network.

Input clip length. Table 3 also compares different models using different input length L. As expected, increasing L from 16 to 32 frames can boost the performance across all datasets. Something and Diving48 benefit more from using longer inputs. It is noteworthy that the improvements of CorrNet over R(2+1)D are largely carried over when using 32 frames. To simplify, we use $L = 32$ frames in all the following experiments.

6.2. Evaluating design choices and comparison to two-stream network

To justify our design choices, we experimentally compare different configurations of CorrNet-26 in Table 4. We consider the following modifications: 1) remove filters in the correlation operator; 2) remove grouping to reduce the number of channels from C to K^2; 3) swap the *correlation-sum* block with *correlation-concat*. Note that we only change one thing at a time.

Removing filters results in an accuracy drop of 1% on both datasets, as it significantly reduces the power of the learned representations. Similarly, the aggressive channel reduction introduced by removing grouping also causes an accuracy drop of about 1%. The *correlation-concat* block performs worse than *correlation-sum*, which leverages the shortcut connection to ease optimization.

Figure 3 shows the performance of CorrNet-26 for $K \in \{3, 5, 7, 9\}$. As expected, a larger K can cover a larger

Methods	Pretrain	Two stream	GFLOPs × crops	Kinetics
STC-ResNext-101 [8]	✗	✗	N/A	68.7
R(2+1)D [54]	✗	✗	152×115	72.0
MARS+RGB [5]	✗	✗	N/A	74.8
ip-CSN-152 [53]	✗	✗	109×30	77.8
DynamoNet [9]	✗	✗	N/A	77.9
SlowFast-101 [14]	✗	✗	213×30	78.9
SlowFast-101+NL [14]	✗	✗	234×30	79.8
I3D [4]	ImageNet	✗	108×N/A	72.1
R(2+1)D [54]	Sports1M	✗	152×115	74.3
NL I3D-101 [58]	ImageNet	✗	359×30	77.7
ip-CSN-152 [53]	Sports1M	✗	109×30	79.2
LGD-3D-101 [41]	ImageNet	✗	N/A	79.4
R(2+1)D [54]	Sports1M	✓	304×115	75.4
I3D [4]	ImageNet	✓	216×N/A	75.7
S3D-G [62]	ImageNet	✓	142.8×N/A	77.2
LGD-3D-101 [41]	ImageNet	✓	N/A	81.2
CorrNet-50	✗	✗	115×10	77.2
CorrNet-101	✗	✗	187×10	78.5
CorrNet-101	✗	✗	224×30	79.2
CorrNet-101	Sports1M	✗	224×30	**81.0**

Table 6: Compare with the state-of-the-art on Kinetics-400.

Methods	Pretrain	Two stream	Something	Diving
R(2+1)D [54]	✗	✗		21.4
TRN [67]	✗	✗	34.4	
MFNet-C101 [33]	✗	✗	43.9	
NL I3D-50 [58]	ImageNet	✗	44.4	
R(2+1)D [54]	Sports1M	✗	45.7	28.9
NL I3D-50+GCN [59]	ImageNet	✗	46.1	
DiMoFs [2]	Kinetics	✗		31.4
Attention-LSTM [25]	ImageNet	✗		35.6
GST-50 [37]	ImageNet	✗	48.6	38.8
MARS+RGB [5]	Kinetics	✗	51.7	
S3D-G [62]	ImageNet	✓	48.2	
TRN [67]	ImageNet	✓	42.0	22.8
MARS+RGB+Flow [5]	Kinetics	✓	53.0	
CorrNet-50	✗	✗	49.3	37.9
CorrNet-101	✗	✗	50.9	38.2
CorrNet-101	✗	✗	51.7	38.6
CorrNet-101	Sports1M	✗	**53.3**	**44.7**

Table 7: Compare with the state-of-the-art on Something-Something v1 and Diving48.

neighborhood while matching pixels, thus yields a higher accuracy. But the improvements become marginal beyond $K = 7$, possibly due to the low resolution of the feature maps.

We compare CorrNet-26 with the two-stream network using the R(2+1)D backbone in Table 5. We use the Farneback [13] algorithm for computing optical flow. The two-stream network of R(2+1)D is implemented by concatenating the features after global average pooling. For R(2+1)D, the accuracy gap between RGB and optical flow is smaller on Something, as Kinetics is arguably more biased towards appearance information. Our CorrNet-26 alone is on par with R(2+1)D-26 using two streams. Note that two-stream network effectively doubles the FLOPs of the backbone and the cost of computing optical flow (not considered here) can be very high as well. This shows that our correlation network is more efficient by learning motion information from RGB pixels directly.

6.3. Comparison to the state of the art

The correlation network discussed in the previous sections is based on R(2+1)D-26 with a block configuration of [2, 2, 2, 2] for res_2, res_3, res_4 and res_5. To compare with the state-of-the-art, we simply add more layers to the backbone. Following the design of ResNet [20], CorrNet-50 uses a block configuration of [3, 4, 6, 3], whereas CorrNet-101 uses [3, 4, 23, 3]. Like in CorrNet-26, a correlation block is inserted after res_2, res_3 and res_4 for CorrNet-50. For CorrNet-101, we insert an additional correlation block in the middle of res_4, so there are 4 correlation blocks in total. Table 6, 7 and 8 compare the accuracy of CorrNet-50 and CorrNet-101 with several recently published results under different settings. For CorrNet-101 (the last two rows of Table 6 and 7) at test time, we sample more clips (30 instead of 10), as done in [58, 59] .

As expected, using deeper models or sampling more clips can further improve the accuracy. Comparing with CorrNet-26 in Table 3, CorrNet-101 is 4.1%, 4.3% and 3.1% better on Kinetics, Something and Diving48, respectively. As Diving48 is the smallest dataset among the four, increasing model capacity may lead to overfitting, thus the improvement is less significant. We also experiment with pre-training CorrNet-101 using the Sports1M dataset [26]. This time we achieve the most significant improvement on Diving48, *i.e.*, 6.1%. Smaller datasets are likely to benefit more from pre-training, as we have seen in the case of UCF101 [47] and HMDB51 [30]. On both Kinetics and Something, we observe a modest improvement of 1-2% by pre-training on Sports1M.

On Kinetics, CorrNet-101 significantly outperforms the previous models using the same setup (*i.e.*, no pretraining and only using RGB) except for the recently introduced SlowFast network [14] augmented with non-local network (NL) [58]. In fact, compared to SlowFast-101, CorrNet-101 achieves slightly higher accuracy (79.2% vs 78.9%), and it is only 0.6% lower in accuracy when SlowFast-101 is combined with NL. Comparing with results using pre-training, CorrNet-101 is 1.6% better than LGD-3D [41], *i.e.*, 81.0% vs 79.4%. The two-stream LGD-3D improves the accuracy to 81.2% by extracting the computationally expensive TV-L1 optical flow [65].

Comparing CorrNet-101 with other approaches trained from scratch in Table 7, we observe a significant improvement of 7.8% on Something (51.7% for CorrNet-101 vs. 43.9% for MFNet-C101 [33]). On Diving48 [34], the improvement is even more substantial, *i.e.*, over 17% (38.6% from CorrNet-101 vs. 21.4% from R(2+1)D). With pre-

Methods	Pretrain	Two stream	Sports1M
C3D [52]	✗	✗	61.1
P3D [40]	✗	✗	66.4
R(2+1)D [54]	✗	✗	73.0
ip-CSN-152 [53]	✗	✗	75.5
Conv Pool [64]	✗	✓	71.7
R(2+1)D [54]	✗	✓	73.3
CorrNet-101	✗	✗	77.1

Table 8: Comparison with the state-of-the-art on Sports1M.

training, CorrNet-101 is still 1.6% and 5.9% better on Something and Diving48. CorrNet-101 even slightly outperforms MARS [5] augmented with RGB and optical flow streams on Something, *i.e.*, 53.3 vs 53.0.

Table 8 provides a comparison with the state of the art on Sports1M. We only evaluate our best model CorrNet-101 to limit the training time. All the methods in Table 8 are trained from scratch since Sports1M is already a very large scale video dataset. Our CorrNet-101 established a new state of the art, *i.e.* 77.1%, outperforming the very recent ip-CSN-152 [53] by 1.6%. CorrNet-101 also significantly outperforms R(2+1)D [54] by 3.8% which uses both RGB and optical flow.

To sum up, CorrNet is a new versatile backbone that outperforms the state-of-the-art on a wide variety of video datasets. Thanks to the efficient design of the correlation operator and our improved R(2+1)D backbone, the FLOPs of CorrNet is also lower than those of previous models, such as NL I3D [58]. FLOPs can further be significantly reduced (*i.e.*, 3x decrease) by sampling fewer clips during testing with only a small drop in accuracy, as shown in the third last row of Table 6 and 7.

6.4. Visualizing Correlation Filters

In this section, we visualize the filters (*i.e.*, the yellow tensor in Fig. 1) from the correlation operator to better understand the model. We choose the CorrNet-101 trained from scratch on Kinetics from Table 6, and the correlation operator with the highest output resolution, *i.e.*, from the correlation block after res_2. The size of the filter is $L \times C \times K \times K$ as listed in Table 1, which is $32 \times 64 \times 7 \times 7$ in this case. We visualize filters for $l = 0, \ldots, 7$ and $c = 0, \ldots, 7$ in Figure 4. The color coding indicates the weights in the learned filters, and the white arrows point to the directions with largest weights.

Zooming into filters in Figure 4, we observe that each filter learns a specific motion pattern (*i.e.*, the 7×7 grid) for matching. The filters in each column are sorted in time and exhibit similar patterns. The white arrows often point to similar directions for the filters in the same column. This suggests that our network learns the temporal consistency

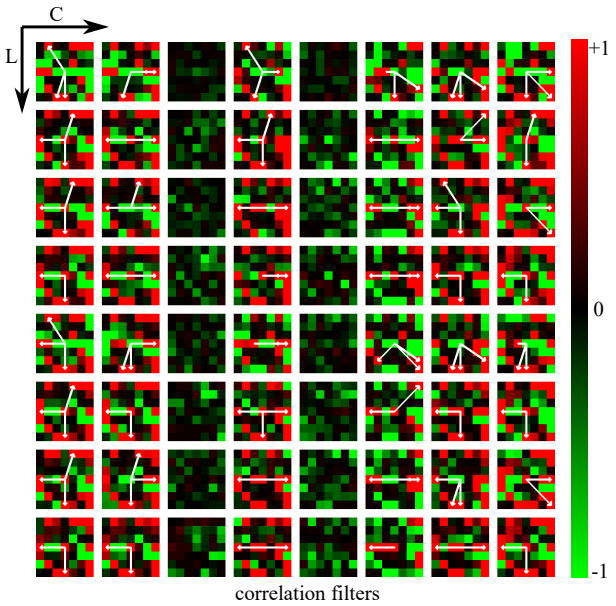


Figure 4: Visualization of CorrNet-101 trained on Kinetics. We visualize the correlation filters, which is a 4D tensor of shape $L \times C \times K \times K$. Filters in each column are aligned in time, and each column represents a different channel dimension. White arrows point to locations with highest weights, showing that different filters learn to match pixels moving in different directions.

of motion, *i.e.*, pixels usually move in the same direction across frames. Comparing filters in different columns, we observe that some columns are more active than others, which indicates that our filters learns which channels are more discriminative for matching. Filter weights for these channels can be larger than channels that are less informative for matching.

7. Conclusions

This paper explores a novel way to learn motion information from video data. Unlike previous approaches based on optical flow or 3D convolution, we propose a learnable correlation operator which establishes frame-to-frame matches over convolutional feature maps in the different layers of the network. Differently from the standard 3D convolution, the correlation operator makes the computation of motion information explicit. We design the correlation network based on this novel operator and demonstrate its superior performance on various video datasets for action recognition. Potential future work includes the application of the learnable correlation operator to other tasks, such as action localization, optical flow, and geometry matching.

References

[1] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*, pages 29–39. Springer, 2011. 1, 2

[2] Gedas Bertasius, Christoph Feichtenhofer, Du Tran, Jianbo Shi, and Lorenzo Torresani. Learning discriminative motion features through detection. *arXiv preprint arXiv:1812.04172*, 2018. 7

[3] Caffe2-Team. Caffe2: A new lightweight, modular, and scalable deep learning framework. https://caffe2.ai/. 5

[4] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 2, 7

[5] Nieves Crasto, Philippe Weinzaepfel, Karteek Alahari, and Cordelia Schmid. Mars: Motion-augmented rgb stream for action recognition. In *CVPR*, pages 7882–7891, 2019. 1, 2, 7, 8

[6] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*, 2006. 3

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE, 2009. 1, 5

[8] Ali Diba, Mohsen Fayyaz, Vivek Sharma, M Mahdi Arzani, Rahman Yousefzadeh, Juergen Gall, and Luc Van Gool. Spatio-temporal channel correlation networks for action classification. *ECCV*, 2018. 2, 3, 7

[9] Ali Diba, Vivek Sharma, Luc Van Gool, and Rainer Stiefelhagen. Dynamonet: Dynamic action and motion network. *ICCV*, 2019. 1, 2, 7

[10] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Proc. ICCV VS-PETS*, 2005. 2, 4

[11] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 1, 2, 3

[12] Lijie Fan, Wenbing Huang, Stefano Ermon Chuang Gan, Boqing Gong, and Junzhou Huang. End-to-end learning of motion representation for video understanding. In *CVPR*, pages 6016–6025, 2018. 2

[13] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003. 7

[14] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, pages 6202–6211, 2019. 7

[15] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 1, 2, 5

[16] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *CVPR*, pages 3038–3046, 2017. 2

[17] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 5

[18] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The something something video database for learning and evaluating visual common sense. In *Proc. ICCV*, 2017. 5

[19] Xiaoyang Guo, Kai Yang, Wukui Yang, Xiaogang Wang, and Hongsheng Li. Group-wise correlation stereo network. In *CVPR*, pages 3273–3282, 2019. 4

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 3, 4, 7

[21] Omar Hommos, Silvia L Pintea, Pascal SM Mettes, and Jan C van Gemert. Using phase instead of optical flow for action recognition. *ECCV Workshops*, 2018. 2

[22] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, pages 7132–7141, 2018. 2

[23] 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*, volume 2, page 6, 2017. 2

[24] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE TPAMI*, 35(1):221–231, 2013. 1, 2

[25] Gagan Kanojia, Sudhakar Kumawat, and Shanmuganathan Raman. Attentive spatio-temporal representation learning for diving classification. *CVPR Workshops*, 2019. 7

[26] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 1, 5, 7

[27] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 5

[28] Alexander Kläser, Marcin Marszałek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008. 2

[29] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 3, 4

[30] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB51: a large video database for human motion recognition. In *ICCV*, 2011. 7

[31] I. Laptev and Tony Lindeberg. Space-time interest points. In *ICCV*, 2003. 2

[32] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[33] Myunggi Lee, Seungeui Lee, Sungjoon Son, Gyutae Park, and Nojun Kwak. Motion feature network: Fixed motion filter for action recognition. In *ECCV*, pages 387–403, 2018. 2, 7

[34] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *ECCV*, pages 513–528, 2018. 5, 7

[35] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *ICLR*, 2017. 5

[36] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 3

[37] Chenxu Luo and Alan L Yuille. Grouped spatial-temporal aggregation for efficient action recognition. In *ICCV*, pages 5512–5521, 2019. 7

[38] Joe Yue-Hei Ng, Jonghyun Choi, Jan Neumann, and Larry S Davis. Actionflownet: Learning motion representation for action recognition. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1616–1624. IEEE, 2018. 2

[39] AJ Piergiovanni and Michael S Ryoo. Representation flow for action recognition. *CVPR*, 2019. 2

[40] Zhaofan Qiu, Ting Yao, , and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 2017. 1, 2, 4, 8

[41] Zhaofan Qiu, Ting Yao, Chong-Wah Ngo, Xinmei Tian, and Tao Mei. Learning spatio-temporal representation with local and global diffusion. In *CVPR*, pages 12056–12065, 2019. 1, 2, 7

[42] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *CVPR*, volume 2, page 2. IEEE, 2017. 3

[43] Ignacio Rocco, Relja Arandjelovic, and Josef Sivic. Convolutional neural network architecture for geometric matching. In *Proc. CVPR*, volume 2, 2017. 2, 3

[44] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM MM*, 2007. 2

[45] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 1, 2, 5

[46] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 2, 5

[47] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human action classes from videos in the wild. In *CRCV-TR-12-01*, 2012. 7

[48] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, pages 8934–8943, 2018. 2, 3

[49] Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015. 1, 2, 4

[50] 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. 2

[51] 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, 2

[52] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 1, 2, 8

[53] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. *ICCV*, 2019. 7, 8

[54] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, pages 6450–6459, 2018. 1, 2, 4, 7, 8

[55] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 2, 4

[56] Limin Wang, Yuanjun Xiong, Zhe Wang, and Yu Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015. 1, 2, 5

[57] 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. 1, 2, 5

[58] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *CVPR*, 10, 2018. 5, 7, 8

[59] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *ECCV*, pages 399–417, 2018. 7

[60] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *ICCV*, pages 1385–1392, 2013. 2, 3

[61] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 5987–5995. IEEE, 2017. 2, 4

[62] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, pages 305–321, 2018. 1, 2, 6, 7

[63] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 3

[64] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, pages 4694–4702, 2015. 8

[65] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. *Pattern Recognition*, pages 214–223, 2007. 7

[66] Yue Zhao, Yuanjun Xiong, and Dahua Lin. Recognize actions by disentangling components of dynamics. In *CVPR*, pages 6566–6575, 2018. 2

[67] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *ECCV*, pages 803–818, 2018. 7