

# 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 three popular benchmarks for action recognition: Kinetics, Something-Something and Diving48.

## 1 Introduction

After the breakthrough of AlexNet [24] on ImageNet [6], convolutional neural networks (CNNs) have become the dominant model for still-image classification [26, 38, 42, 16]. In the video domain, CNNs were initially adopted as image-based feature extractor on individual frames of the video [21]. 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 [37, 46, 12, 47] where one stream operates on RGB frames to model appearance information while 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, 20, 43] which couple appearance and temporal modeling by means of spatiotemporal kernels.

In this paper we propose an alternative scheme based on a novel correlation operator inspired by the correlation layer in FlowNet [9]. 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. As shown in Figure 1, different filters match pixels moving in different directions. Through our extensive experiments on three action recognition



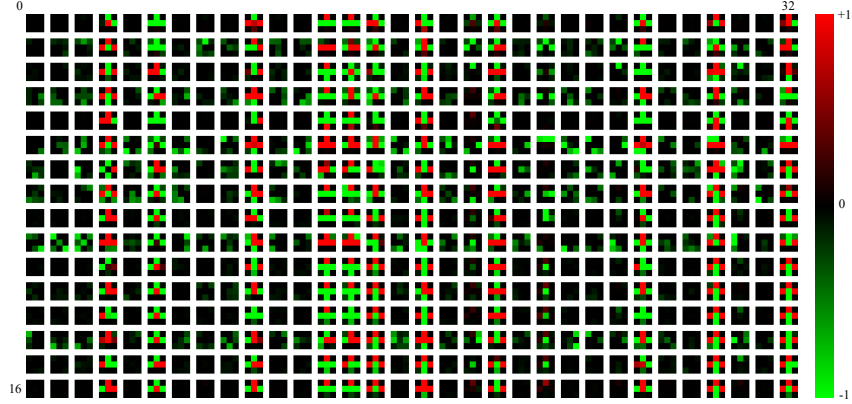


Figure 1: Visualization of correlation filters trained on Kinetics. The size of each filter is  $16 \times 32 \times 3 \times 3$ . 16 is the number of frames (shown from top to bottom); 32 is the number of channels (shown from left to right);  $3 \times 3$  is the spatial size of each filter. The similarity of filters in the same column shows that our network learns the temporal consistency of motion, *i.e.*, pixels usually move in the same direction across frames. Filters in some columns are more active than the other columns, which indicates our network learns which channels are more discriminative for matching. Zooming in to each filter, we observe that different filters learn to match pixels moving in different directions.

datasets (Kinetics, Something-Something and Diving48), 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 three different video datasets without pretraining or using optical flow.

In the rest of the paper, we introduce related work in Sec. 2, and detail the proposed correlation operator in Sec. 3. We present the correlation network in Sec. 4. Experimental setups are in Sec. 5. We discuss the experimental results in Sec. 6 and conclude the paper in Sec. 7. Code and trained models will be made publicly available at the time of publication.

## 2 Related Work

**Architectures for video classification.** Among the popular video models, there are two major categories: two-stream networks [37, 46, 12, 47] and 3D CNNs [1, 20, 43, 40, 44, 33, 53]. Since the introduction of two-stream networks [37], further improvements have been achieved by adding connections between the two streams [12], or inflating a 2D model to 3D [4]. 3D CNNs [1, 20, 43] learn appearance and motion information simultaneously by convolving 3D filters in space and time. Successful image architectures [38, 42, 16] have been extended to video using 3D convolution [4, 43, 53]. Recent research [40, 44, 33] 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 [25, 36, 23, 8, 45] were hand-designed to encode motion information in video. Besides two-stream networks and 3D CNNs, ActionFlowNet [31] proposes to jointly estimate optical flow

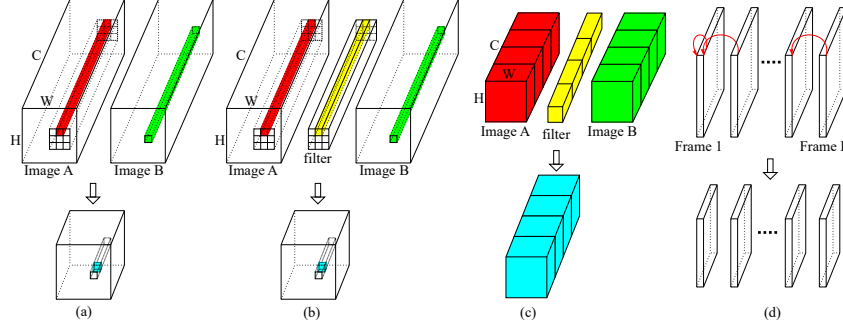


Figure 2: An illustration of the proposed correlation operator. (a) Correlation operator used for optical flow and geometric matching. (b) The introduction of filters renders the operator “learnable.” (c) Groupwise correlation increases the number of output channels without adding computational cost. (d) Extending the correlation operator to work on a sequence of video frames.

and recognize actions in one network. Fan *et al.* [10] and Piergiovanni *et al.* [32] also introduced networks to learn optical flow end-to-end for action recognition.

There is also work [41, 27, 17] seeking alternatives to optical flow. Sun *et al.* [41] extracted features guided by optical flow to capture the transformation between adjacent frames. Lee *et al.* [27] designed motion filters by computing the difference of adjacent frames. Hommos *et al.* [17] 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 [50] computes the correlation of image patches to find dense correspondence to improve optical flow. Unlike deep matching using hand-crafted features, FlowNet [9] is a network, where a correlation layer performs multiplicative patch comparisons. Correlation layers were also used in other CNN-based optical flow algorithms [39, 19]. Besides optical flow, Rocco *et al.* [35] used it to estimate the geometric transformation of two images, whereas Feichtenhofer *et al.* [13] applied it to object tracking.

In the context of action recognition, Zhao *et al.* [55] utilize the correlation layer to compute a cost volume to estimate the displacement map as in optical flow. The Spatio-Temporal Channel Correlation Network [7] adapts the Squeeze-and-Excitation block [18] to a ResNeXt [51] backbone. The notion of correlation in [7] refers to the relationship among the spatial and temporal dimensions of the feature maps, which is different from the matching of adjacent frames studied in our work. We compare our results with [7] in Sec. 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 [9, 39] and geometric matching [50, 35]. 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 2 (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  $P^A(i, j)$  becomes a

L is the number of frame

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.

**C-dimensional vector.** 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 [30, 5] to learnable deep neural networks [24, 16]. The original correlation operator [9, 39, 50, 35] 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 2 (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 [54], 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  $13 \times 13$  pixels. Besides dilation, we also apply the operator at different spatial scales (as discussed in Sec. 4), which is a popular strategy to handle large displacements in optical flow [34]. From Figure 1, 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 Sec. 6.2.

We propose a groupwise version of the correlation operator that avoids shrinking the number of channels while maintaining efficiency. Groupwise convolution [24, 51] 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 2 (c). This increases the number of output channels by a factor of  $G$ , to a total of  $K^2 G$  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.

Layers	R(2+1)D-26	Output size
conv <sub>1</sub>	$1 \times 7 \times 7, 64, \text{stride } 1, 2, 2$	$L \times 112 \times 112$
res <sub>2</sub>	$\begin{bmatrix} 1 \times 1 \times 1, 64 \\ 3 \times 1 \times 1, 64 \\ 1 \times 3 \times 3, 64 \\ 1 \times 1 \times 1, 256 \end{bmatrix} \times 2$	$L \times 56 \times 56$
res <sub>3</sub>	$\begin{bmatrix} 1 \times 1 \times 1, 128 \\ 3 \times 1 \times 1, 128 \\ 1 \times 3 \times 3, 128 \\ 1 \times 1 \times 1, 512 \end{bmatrix} \times 2$	$L \times 28 \times 28$
res <sub>4</sub>	$\begin{bmatrix} 1 \times 1 \times 1, 256 \\ 3 \times 1 \times 1, 256 \\ 1 \times 3 \times 3, 256 \\ 1 \times 1 \times 1, 1024 \end{bmatrix} \times 2$	$\frac{L}{2} \times 14 \times 14$
res <sub>5</sub>	$\begin{bmatrix} 1 \times 1 \times 1, 512 \\ 3 \times 1 \times 1, 512 \\ 1 \times 3 \times 3, 512 \\ 1 \times 1 \times 1, 2048 \end{bmatrix} \times 2$	$\frac{L}{4} \times 7 \times 7$
global average pool, fc		# classes

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

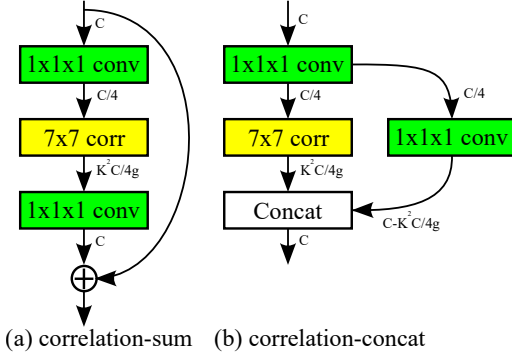


Figure 3: Two types of correlation blocks. We mark the number of channels for each operator.

**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 2 (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 1 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 [44], then discuss how to build the correlation network to leverage the matching information by incorporating the correlation operator into the backbone.

**R(2+1)D backbone.** The R(2+1)D network [44] 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 [45, 8] and 3D CNNs [44, 40, 33]. Compared with the original R(2+1)D [44], 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 3. The design of these blocks is similar in spirit to that of the bottleneck block [16]. Figure 3 (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 [16] is applied for residual learning. The *correlation-concat* block in Figure 3 (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 Sec. 6.2.

We obtain the final correlation network by inserting the correlation block into the backbone architecture. In this paper, we insert one correlation block after  $\text{res}_2$ ,  $\text{res}_3$  and  $\text{res}_4$  in Table 2. We omit  $\text{res}_5$  as its spatial resolution is low (*i.e.*,  $7 \times 7$ ). 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. Sec. 6.1 provides a more quantitative analysis.

## 5 Experimental Setups

**Video Datasets.** We evaluate our model on three video datasets that have rather different properties, emphasizing distinct aspects of action recognition. **Kinetics** [22] is among the most popular datasets for video classification. It consists of about 300K YouTube videos covering 400 categories. **Something-Something** [15] 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*<sup>2</sup> for short. **Diving48** [28] 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. For Kinetics and *Something*<sup>2</sup>, annotations on the testing set are not public available, so we report accuracy on the validation set like others. For Diving48, we report accuracy on the testing set following the setup by the authors [28].

**Training and Testing.** To train the correlation network, we sample a clip of  $L = 32$  frames with a resolution of  $224 \times 224$  from a given video. Some videos in *Something*<sup>2</sup> don't 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 [48, 38], 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 [29]. We train the model for 180 epochs in total with the first 40 epochs for warm-up [14]. For testing, we sample 10 clips uniformly spaced out in the video and average the clip-level predictions to generate the video-level results. All models are trained from scratch without pretraining on ImageNet [6] or other large-scale video datasets. We only use RGB as the input to our model, unlike two-stream networks [37, 46, 12, 47] which use both RGB and optical flow. Unless stated otherwise, we use the above setup for all the experiments on all three datasets in the paper.

## 6 Experimental Evaluation

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

### 6.1 Correlation operator vs temporal convolution

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 two variants of  $R(2+1)D-26$ :  $R2D-26$  is obtained by removing all 1D temporal convolutions (*i.e.*,  $3 \times 1 \times 1$ ), and  $R2D-26(2x)$  is produced by replacing each  $3 \times 1 \times 1$  with an additional  $1 \times 3 \times 3$ . *CorrNet-26* is obtained by inserting one *correlation-sum* block after  $\text{res}_2$ ,  $\text{res}_3$  and  $\text{res}_4$  of  $R(2+1)D-26$  as described in Sec. 4. As the correlation block adds a small overhead to the FLOPs, we create two variants of *CorrNet-26* by removing  $3 \times 1 \times 1$  in  $\text{res}_2$  and  $\text{res}_3$  to bring down the number of FLOPs to be comparable to that of the original baseline.



Models	GFLOPs	Top-1 accuracy (%)		
		Kinetics	Something <sup>2</sup>	Diving48
R2D-26	55.0	69.6	28.3	28.4
R2D-26 (2x)	75.3	72.1	34.6	30.9
R(2+1)D-26	71.9	71.7	45.2	31.1
CorrNet-26	82.8	<b>74.5</b>	<b>47.7</b>	34.3
CorrNet-26a	76.6	74.3	46.4	<b>34.4</b>
CorrNet-26b	70.5	73.7	45.6	33.6

Table 3: Correlation networks vs baselines. CorrNet-26a and CorrNet-26b denote CorrNet-26 variants with fewer FLOPs by removing  $3 \times 1 \times 1$  convolutions in  $\text{res}_2$  for CorrNet-26a and in both  $\text{res}_2$  and  $\text{res}_3$  for CorrNet-26b.

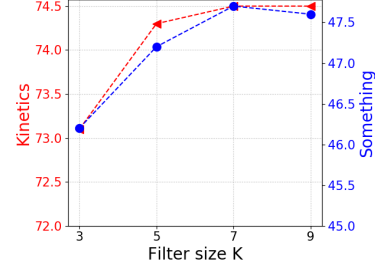


Figure 4: Effect of filter size  $K$  on classification accuracy.

Datasets	Kinetics	Something <sup>2</sup>
CorrNet-26	74.5	47.7
w/o filter	73.6	46.5
w/o grouping	73.8	46.3
<i>correlation-concat</i>	72.2	46.1

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

Datasets	Kinetics	Something <sup>2</sup>
CorrNet-26	74.5	47.7
R(2+1)D-26 (RGB)	71.7	45.2
R(2+1)D-26 (OF)	66.4	42.9
R(2+1)D-26 (Two-stream)	74.1	48.8

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

**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 2-3% better than R2D, but the gap widens to 17% on Something<sup>2</sup>. This is consistent with findings in [52] and it is due to the design of Something<sup>2</sup> where objects are not predictive of the action label. One may argue that the superior performance of R(2+1)D over R2D is due to the larger number of FLOPs (71.9 vs. 55.0). But we note that R2D (2x) has higher FLOPs than R(2+1)D, and yet its accuracy on Something<sup>2</sup> is much lower.

**R(2+1)D vs CorrNet.** We observe a consistent improvement of about 3% on all three datasets when comparing CorrNet with R(2+1)D in Table 3. 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) reported. A significant 3% improvement on all 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. When comparing the two variants of CorrNet with R(2+1)D, the correlation network achieves a much better accuracy/FLOPs trade-offs, *i.e.*, a higher accuracy with fewer FLOPs.

## 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 4 shows the performance of CorrNet-26 for  $K \in \{3, 5, 7, 9\}$ . As expected, a larger  $K$  can cover a larger neighborhood while matching pixels, thus yields a higher accuracy. But the accuracy seems to be saturated 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 [11] 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<sup>2</sup>, 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

Methods	Pretrain	Two stream	GFLOPs × crops	Kinetics	Something <sup>2</sup>	Diving48
I3D [4]	✗	✓	216×N/A	71.6		
I3D [4]	ImageNet	✓	216×N/A	75.7		
R(2+1)D [44]	✗	✗	152×115	72.0		21.4
R(2+1)D [44]	Sports1M	✗	152×115	74.3	45.7	28.9
NL I3D-50 [48]	ImageNet	✗	282×30	76.5	44.4	
NL I3D-50+GCN [49]	ImageNet	✗	282×30		46.1	
NL I3D-101 [48]	ImageNet	✗	359×30	77.7		
MFNet-C101 [27]	✗	✗	N/A×N/A		43.9	
S3D-G [52]	ImageNet	✓	142.8×N/A	77.2	48.2	
STC-ResNext-101 [7]	✗	✗	N/A×N/A	68.7		
ECO [57]	✗	✓	N/A×N/A	70.0	49.5	
TRN [56]	ImageNet	✓	N/A×N/A		42.0	22.8
DiMoFs [2]	Kinetics	✗	N/A×N/A			31.4
CorrNet-50	✗	✗	115×10	76.5	48.5	36.0
CorrNet-101	✗	✗	187×10	77.6	50.4	37.4
CorrNet-101	✗	✗	224×30	<b>78.5</b>	<b>51.1</b>	<b>37.7</b>

Table 6: Comparing the accuracy of the correlation network with the state-of-the-art.

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  $\text{res}_2$ ,  $\text{res}_3$ ,  $\text{res}_4$  and  $\text{res}_5$ . To compare with the state-of-the-art, we simply add more layers to the backbone. Following the design of ResNet [16], 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  $\text{res}_2$ ,  $\text{res}_3$  and  $\text{res}_4$  for CorrNet-50 and CorrNet-101. Table 6 compares the accuracy of CorrNet-50 and CorrNet-101 with several recently published results on the three datasets. For CorrNet-101 (the last row of Table 6) at test time, we increase the resolution of the clip (from  $224 \times 224$  to  $256 \times 256$ ) and sampled more clips (30 instead of 10), as done in [48, 49].

As expected, using deeper models or sampling more clips can further improve the accuracy. Comparing with CorrNet-26 in Table 3, CorrNet-101 is about 3-4% better on all three datasets. On Kinetics, CorrNet-101 significantly outperforms the previous results using the same setup (*i.e.*, no pretraining and only using RGB) by a large margin of 6.5%. On Something<sup>2</sup>, we observe a similar improvement of 7.2% (51.1% for CorrNet-101 vs. 43.9% for MFNet-C101 [27]). On Diving48 [28], the improvement is even more substantial, *i.e.*, over 16%.

Pretraining on ImageNet [4, 44, 48, 52] and using optical flow [4, 44, 52] are popular ways of boosting performance on public benchmarks. Without the extra steps and the additional cost of pretraining and using optical flow, CorrNet-101 outperforms the state of the art on all three datasets by 0.8%, 1.6% and 6.3%, respectively. Thanks to the efficient design of the correlation operator and our improved R(2+1)D backbone, the FLOPs of CorrNet-101 is also lower than those of previous models, such as NL I3D [48]. FLOPs can further be significantly reduced (*i.e.*, 4x decrease) by sampling fewer clips during testing with only a small drop in accuracy, as shown in the second last row of Table 6.

## 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] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*, pages 29–39. Springer, 2011.
- [2] G. Bertasius, C. Feichtenhofer, D. Tran, J. Shi, and L. Torresani. Learning discriminative motion features through detection. *arXiv preprint arXiv:1812.04172*, 2018.
- [3] Caffe2-Team. Caffe2: A new lightweight, modular, and scalable deep learning framework. <https://caffe2.ai/>.
- [4] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.
- [5] N. Dalal, B. Triggs, and C. Schmid. Human Detection Using Oriented Histograms of Flow and Appearance. In A. Leonardis, H. Bischof, and A. Pinz, editors, *European Conference on Computer Vision (ECCV '06)*, volume 3952 of *Lecture Notes in Computer Science (LNCS)*, pages 428–441, Graz, Austria, May 2006. Springer-Verlag.
- [6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.
- [7] A. Diba, M. Fayyaz, V. Sharma, M. M. Arzani, R. Yousefzadeh, J. Gall, and L. Van Gool. Spatio-temporal channel correlation networks for action classification. *arXiv preprint arXiv:1806.07754*, 2018.
- [8] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Proc. ICCV VS-PETS*, 2005.
- [9] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, 2015.
- [10] L. Fan, W. Huang, S. E. Chuang Gan, B. Gong, and J. Huang. End-to-end learning of motion representation for video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6016–6025, 2018.
- [11] G. Farneback. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003.
- [12] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016.
- [13] C. Feichtenhofer, A. Pinz, and A. Zisserman. Detect to track and track to detect. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3038–3046, 2017.
- [14] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.
- [15] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, et al. The? something something? video database for learning and evaluating visual common sense. In *Proc. ICCV*, 2017.
- [16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [17] O. Hommos, S. L. Pintea, P. S. Mettes, and J. C. van Gemert. Using phase instead of optical flow for action recognition. *arXiv preprint arXiv:1809.03258*, 2018.
- [18] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.
- [19] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE conference on computer vision and pattern recognition (CVPR)*, volume 2, page 6, 2017.
- [20] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE TPAMI*, 35(1):221–231, 2013.
- [21] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.
- [22] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.
- [23] A. Kläser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008.
- [24] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.
- [25] I. Laptev and T. Lindeberg. Space-time interest points. In *ICCV*, 2003.
- [26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [27] M. Lee, S. Lee, S. Son, G. Park, and N. Kwak. Motion feature network: Fixed motion filter for action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 387–403, 2018.
- [28] Y. Li, Y. Li, and N. Vasconcelos. Resound: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 513–528, 2018.
- [29] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.
- [30] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

- [31] J. Y.-H. Ng, J. Choi, J. Neumann, and L. 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.
- [32] A. Piergiovanni and M. S. Ryoo. Representation flow for action recognition. *arXiv preprint arXiv:1810.01455*, 2018.
- [33] Z. Qiu, T. Yao, , and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 2017.
- [34] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 2. IEEE, 2017.
- [35] I. Rocco, R. Arandjelovic, and J. Sivic. Convolutional neural network architecture for geometric matching. In *Proc. CVPR*, volume 2, 2017.
- [36] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM MM*, 2007.
- [37] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.
- [38] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.
- [39] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018.
- [40] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015.
- [41] S. Sun, Z. Kuang, L. Sheng, W. Ouyang, and W. Zhang. Optical flow guided feature: a fast and robust motion representation for video action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [42] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.
- [43] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.
- [44] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.
- [45] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.
- [46] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015.
- [47] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.
- [48] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. *arXiv preprint arXiv:1711.07971*, 10, 2017.
- [49] X. Wang and A. Gupta. Videos as space-time region graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 399–417, 2018.
- [50] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1385–1392, 2013.
- [51] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5987–5995. IEEE, 2017.
- [52] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning for video understanding. *arXiv preprint arXiv:1712.04851*, 2017.
- [53] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321, 2018.
- [54] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.
- [55] Y. Zhao, Y. Xiong, and D. Lin. Recognize actions by disentangling components of dynamics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6566–6575, 2018.
- [56] B. Zhou, A. Andonian, A. Oliva, and A. Torralba. Temporal relational reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 803–818, 2018.
- [57] M. Zolfaghari, K. Singh, and T. Brox. Eco: Efficient convolutional network for online video understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 695–712, 2018.