Learnable Higher-Order Representation for Action Recognition

Jie Shao, Xiangyang Xue
School of Computer Science, Shanghai Key Lab of Intelligent Information Processing,
Fudan University, Shanghai, China
Email: shaojie@fudan.edu.cn, xyxue@fudan.edu.cn

Abstract—Capturing spatiotemporal dynamics is an essential topic in video recognition. In this paper, we present learnable higher-order operations as a generic family of building blocks for capturing spatiotemporal dynamics from RGB input video space. Similar to higher-order functions, the weights of higher-order operations are themselves derived from the data with learnable parameters. Classical architectures such as residual learning and network-in-network are first-order operations where weights are directly learned from the data. Higher-order operations make it easier to capture context-sensitive patterns, such as motion. Self-attention models are also higher-order operations, but the attention weights are mostly computed from an affine operation or dot product. The learnable higher-order operations can be more generic and flexible. Experimentally, we show that on the task of video recognition, our higher-order models can achieve results on par with or better than the existing state-of-the-art methods on Something-Something (V1 and V2), Kinetics and Charades datasets.

I. INTRODUCTION

Actions in videos arise from motions of objects with respect to other objects and/or the background. To understand an action, an effective architecture should recognize not only the appearance of the target object associated with the action, but also how it relates to other objects in the scene, in both space and time. Fig. 1 shows four different categories of actions. Each column shown an action where, in temporal order, the figures above occur before the figures below. Recognizing the hand and the object is not enough. To distinguish left to right motion from right to left motion, the model must know how the hand moves against the background. It is more complicated to classify pull and push since it is an XOR operation on the relative positions of the hand and the object resulting from the hand’s movements.

The key point here is the need for recognizing patterns in spatiotemporal context. Even the same hand-iron-background combination has different meanings in different spatiotemporal contexts. The number of combinations increases sharply as scenes become more complicated and the number of objects involved increases. Conventional convolution recognizes fixed patterns determined by the fixed filter parameters, so it is difficult to capture various variations that distinguish the action categories. To recognize every object-in-context pattern, the model needs to have more detailed filters, potentially leading to a blow up of the number parameters.

On the other hand, although the object-in-context patterns can vary, they are related through a higher-order structure: pushing an iron, pulling an iron, pushing a pen, and so on. We hypothesize that the structure of object-in-context patterns can be learned, i.e., the model can learn to conclude object-in-context pattern given the context, and propose a corresponding feature extractor.

Explicitly, let \( X \) and \( Y \) respectively represent the input and output of a convolution. Let \( y_p \) and \( \{x_{p}\} \) represent a specific position of \( Y \) and the set of positions of \( X \) where \( y_p \) is computed, respectively. Denote conventional convolution operation (first-order) as \( Y = f(X; \Theta) \) where \( \Theta \) is the shared parameters at different positions. The parameters act as determined feature extractors as \( y_p = f(\{x_{p}\}; \Theta) \) for different positions.

As we analyze, the visual pattern of the target object can vary in different contexts, and feature extractors (filters) that ignore this dependence are not optimal. We replace the fixed filters with context-dependent filters \( y_p = f(\{x_{p}\}; w_p) \) where the filters \( w_p \) are in turn obtained as \( w_p = g(\{x_{p}\}; \Theta) \). The mapping \( g \) is the structure of object-in-context patterns and \( \Theta \) are the learned parameters as we hypothesize. The entire relation between \( Y \) and \( X \) can be represented through \( Y = f(X; g(X; \Theta)) \). We define this as a higher-order operation since the function \( f \) takes function \( g \) as an argument.

The proposed model is able to capture spatiotemporal contexts effectively. We test our method on four benchmark datasets for action recognition: Kinetics-400 [1], Something-Something V1 [2], Something-Something V2, and Charades datasets [3]. Specifically, we make comprehensive ablation studies on Something-Something V1 datasets and further evaluate on the other three datasets to demonstrate the generality of our proposed method. The experiments establish significant advantages of the proposed models over existing algorithms, achieving results on par with or better than the current state-of-the-art methods.

II. RELATED WORK

A. Action Recognition

Many action recognition methods are based on high-dimensional encoding of local features. For instance, Laptev et al. [4] use the local features histograms of oriented gradients (HOG) [5] and histograms of optical flow (HOF) as sparse interest points. The features are encoded into a bag of features...
representation. Wang et al. [6] and Peng et al. [7] made use of dense point trajectories which are computed using optical flow.

The high performance of Convolutional Neural Network (ConvNets) in image classification tasks [8] makes it appealing to try to reuse them for video recognition tasks. Tran et al. investigated 3D ConvNets to learn spatiotemporal features end-to-end [9]. Some researchers tried to save computation by replacing 3D convolutions with separable convolutions [10], [11] or mixed convolutions [11], [12]. Meanwhile, Carreira and Zisserman introduced an inflation operation [1]. It allows for converting pre-trained 2D models into 3D. Simonyan et al. designed a two-stream architecture to capture appearance and motion information separately [13]. The spatial stream uses RGB frames as inputs, while the temporal stream learns from stacked optical flow. Wang et al. further generalized this framework to learn long-range dependencies by temporal segment [14].

Self-attention mechanisms have recently been successfully applied in visual recognition [15]–[20]. Though we do not use key and query pairs, our method can be seen as a more generalized form of self-attention that learns more structured information from the feature map.

B. Spatiotemporal Context

Contextual information is very important for action recognition. Several methods [21]–[26] exploit contextual information to facilitate action recognition. Marszalek et al. exploited the context of natural dynamic scenes for human action recognition in video [21]. Sun et al. modeled the spatiotemporal relationship between trajectories in a hierarchy of multiple levels [22]. Kovashka et al. proposed to learn the shapes of space-time feature neighborhoods that are most discriminative for a given action category [23]. Conditional Random Field models have also been exploited for object and action recognition [24]–[28].

The long-range dependency in space and time is also very important. Wang et al. investigated the non-local mean operation to captures long-range dependency by iterative utilization of local and non-local operations [18]. Cao et al. found that the global contexts modeled by non-local network are almost the same for different query positions within an image and proposed the global context (GC) block [29]. Qiu et al. proposed a two-path method to combine local and global representations [30].

Wu et al. used dynamic convolutions where the kernel is a linear output of the context window to learn from the context [31]. In our experiment, we also find a relatively non-local context is important but a more global context shows diminishing return. So we choose a context field that is not global but much larger than the typical convolutional kernel size. Some related work studied using the network to generate a network [32], [33]. The generated network is fixed after training. Ours is estimating different weights for every input.

III. LEARNABLE HIGHER-ORDER REPRESENTATION

In this section, we define our higher-order model for video analysis. Our model comprises the analysis of video feature maps by a position-dependent bank of spatio-temporal filters. The filter parameters are themselves derived through a smaller network at different positions.

The description below represents one layer or block of a larger model. We will refer to such second (or more generally, higher) order blocks as H-blocks. We note that the larger model may be composed entirely of H-blocks, or include H-blocks intermittently between conventional convolutional layers. To
allow for this more generic interpretation we will define our
blocks as working on video feature maps and producing video
feature maps, where the input map may either be the original
video itself or the output of prior blocks.

A. Notation

We denote the input video feature map of the H-block as
\[ X \in \mathbb{R}^{C_{in} \times T \times H \times W}, \]
where \( C_{in} \) is the number of channels in each frame of the video, \( T \) is the number of frames, and
the height and the width of each frame are \( H \) and \( W \). The feature/content at position \( p = (t, h, w) \), \( 1 \leq t \leq T, 1 \leq h \leq
W, 1 \leq w \leq W, \) is represented as \( x_p \), and \( x_p \in \mathbb{R}^{C_{in}} \).

We denote the output map for the H-block as \( Y \in \mathbb{R}^{C_{out} \times T' \times H' \times W'}. \) The description below assumes, for conve-
nience, that the spatio-temporal dimensions of the output map
are identical to those of the input (i.e. \( T' = T, H' = H, \) and
\( W' = W \)) although this is not essential. Similarly to the input,
we denote elements at individual spatio-temporal positions of
the output as \( y_p \), where \( y_p \in \mathbb{R}^{C_{out}} \).

In our model \( Y \) is derived from \( X \) through a second-order
relation of the form \( Y = f(X, g(X; \Theta)) \) – the relation being
second order since the function \( f \) relating the input and output
maps takes a function \( g \) as arguments to generate parameters
of function \( f \). Both \( f(\cdot) \) and \( g(\cdot) \) are convolution-like
(or actual convolution) operations; hence we can use terminology
drawn from convolutional neural networks to describe them.
As reference, we first describe the common convolutional
network structure, and subsequently build our model from it.

Following [34], we use a grid \( \mathcal{R} \) over the input feature map
to specify the receptive field size and dilation for convolution
kernels. For example (all integers below),
\[
\mathcal{R} = \left\{ (t, h, w) | t \leq K_t, \ |h| \leq K_h, \ |w| \leq K_w \right\}
\]
defines a 3D kernel with kernel size \( (2K_t+1) \times (2K_h+1) \times
(2K_w+1) \) and dilation 1. The usual convolution operation can
now be written as
\[
y_p = \sum_{q \in \mathcal{R}} W_{q,p} \cdot x_{p+q}. \tag{2}
\]
where \( \{W_{q,p} \in \mathbb{R} \} \) are the weights of convolutional filters
that scan the input \( X \). Each \( W_{q,p} \) is a matrix: \( W_{q,p} \in \mathbb{R}^{C_{out} \times C_{in}} \).
The block outputs are generally further processed by an
activation function such as ReLU and \( \text{tanh} \).

Our H-block retains the same structure as above, except that
the convolution operation of Equation 2 changes to
\[
y_p = \sum_{q \in \mathcal{R}} W_{q,p} \cdot x_{p+q} + g(x_p; \Theta). \tag{3}
\]
Note that the filter \( W_{q,p} \) are now position dependent. If
the model learns these filter as parameters from the data,
Equation 3 is a local connected layer which requires nu-
merous parameters. In the higher-order model, filters \( W_{q,p} \)
are themselves derived using an upper-level function with
parameters. Representing the entire set of filter parameters as
\( \mathcal{W} = \{W_{q,p}\} \), we have
\[
W = g(X; \Theta)
\]
The actual number of parameters required to define the block
is the number of components in \( \Theta \). We propose two models
for \( g(\cdot) \) below, with different requirements for the number
of parameters.

B. Convolution-based second-order operation

In the convolution-based model for \( g(\cdot) \), we derive the filter
parameters \( \{W_{q,p}\} \) through a convolutional operation. Since
the total number of parameters in \( \{W_{q,p}\} \) can get very large,
we restrict each \( W_{q,p} \) to be a diagonal matrix, which can
equivalently be represented by the vector \( w_{q,p} \). It is similar
to a depth-wise convolution, but the weights are not shared
between different positions in the feature map. Equation 3 can
now be rewritten as
\[
y_p = \sum_{q \in \mathcal{R}} w_{q,p} \cdot x_{p+q}. \tag{4}
\]
where \( \otimes \) represents a component-wise (Schur) multiplication.
The filter parameters \( w_{q,p} \) are derived from \( X \) through a
convolutional operation as
\[
w_{p,q} = \sum_{t \in \mathcal{R}'} \Theta_{q,t} \cdot x_{p+t} \tag{5}
\]
where \( \mathcal{R}' \) (similar to \( \mathcal{R} \)) is the receptive field for the convolu-
tional filters, and represents the context field, i.e., the span
from which contexts are captured to compute the weight
\( w_{q,p} \), and \( \Theta_{q,t} \) are the convolutional operation’s parameters.
Each \( \Theta_{q,t} \) is a \( C_{in} \times C_{in} \) matrix. The complete set of parameters of \( g \)
are given by \( \Theta = \{ \Theta_{q,t} \}, \ q \in \mathcal{R}, \ t \in \mathcal{R}', \) with the total number
of parameters equal to \( C_{in}^2 \times |\mathcal{R}| \times |\mathcal{R}'| \) where \( |\mathcal{R}| \) is the
number of elements in \( \mathcal{R} \).

The shared weights \( \Theta \) capture the higher-level patterns
required to characterize spatio-temporal context. We define \( \mathcal{R}' \)
as the context field where context information is captured and
define \( \mathcal{R} \) as the kernel size where features are extracted from
\( |\mathcal{R}| \) positions.

C. ConVNet-based second-order operation

In the ConVNet-based second-order block, we use a small
convolutional neural network comprising multiple layers of
convolutions followed by activations to compute \( w_{p,q} \), i.e.
\( w_{p,q} = \text{ConvNet}_{q} \{ x_{p+t}, t \in \mathcal{R} \}; \Theta' \). We can generate all
\( w_{q,p} \) with one ConvNet. Representing \( w^p = \{ w_{q,p}, \ q \in \mathcal{R} \} \),
we can write
\[
w^p = \text{ConvNet} \{ x_{p+t}, t \in \mathcal{R} \}; \Theta', \tag{6}
\]
where \( \Theta \) are the parameters of the ConvNet.

Though ConvNets consist of multiple layers of convolu-
tions, the number of parameters in Equation 6 can be smaller
than that required by the convolutional-based model of
Equation 5. Typically we need a non-local context field, so
\( \mathcal{R}' \) is relatively bigger than the common convolutional kernel size.
For instance, we want a \( 5 \times 5 \times 5 \) context field. A ConvNet
with three layers, each computed by a \( 1 \times 3 \times 3 \), \( 3 \times 3 \times 3 \) and
\( 3 \times 1 \times 1 \) filter using only 39 parameters, whereas a single
convolution would require a \( 5 \times 5 \times 5 \) filter with 125 parameters.
Fig. 2. One example of a second-operation on 2D data with channel number $C$, width $W$, and height $H$. For every position $p$ in the feature map ($HW$ positions in total), the ConvNet derives $9$ $C$-dimensional vectors. They are concatenated into a $C' \times 3 \times 3$ filter to extract the $p$th output feature from a $3 \times 3$ region of the input feature map centered at $p$.

A. Implementation Details

To draw fair comparison with the results in [18], [38] on the same datasets, our backbone model is based on the ResNet-50 Inflated 3D architecture (Table II) and is the same as that in [38]. Note there are small differences between our backbone model with the Inflated 3D backbone in [18] where the output of the last convolutional layer is a $7/2 \times 14 \times 14$ feature map ($T$ is the number of input frames).

### Table I

| Factorization of Different Context Fields | context field | layer 1 | layer 2 | layer 3 |
|------------------------------------------|--------------|--------|--------|--------|
| $3 \times 3 \times 3$                    | $1 \times 3 \times 3$ | $3 \times 1 \times 1$ | $1 \times 1 \times 1$ |
| $3 \times 5 \times 5$                    | $1 \times 3 \times 3$ | $3 \times 3 \times 3$ | $1 \times 1 \times 1$ |
| $5 \times 5 \times 5$                    | $1 \times 3 \times 3$ | $3 \times 3 \times 3$ | $3 \times 1 \times 1$ |
| $5 \times 7 \times 7$                    | $1 \times 3 \times 3$ | $3 \times 3 \times 3$ | $3 \times 3 \times 3$ |
| $7 \times 7 \times 7$                    | $3 \times 3 \times 3$ | $3 \times 3 \times 3$ | $3 \times 3 \times 3$ |

Suppose the number of the H-block’s input channels is $C$ and the kernel size of the H-block is $|\mathcal{R}|$; the number of input channels and output channels for the three P3D convolutions are $(C, C)$, $(C, |\mathcal{R}|, C)$, and $(C, |\mathcal{R}|, C, |\mathcal{R}|)$ respectively ($\lceil\cdot\rceil\, $is integer division, for example $19\lceil/9\rceil = 2$).

After each convolution layer, we use the scaled exponential linear unit (SELU) [36] as the activation. The last convolution is always a group convolution [37] with group size $|\mathcal{R}|$ to reduce parameters. And we use softmax as the last convolution’s activation as a normalization factor.

### IV. Experiments

We perform comprehensive studies on the challenging Something-Something V1 dataset [2], and also report results on the Charades [3], Kinetics-400 [1] and Something-Something V2 dataset to show the generality of our models.
TABLE III
ABLATIONS ON SOMETHING-SOMETHING V1 ACTION CLASSIFICATION

| 1) Stages | 2) Position within Stages | 3) Number of H-Blocks |
|-----------|---------------------------|-----------------------|
| Model     | Top-1 | Top-5 | Model     | Top-1 | Top-5 | Model     | Top-1 | Top-5 |
| I3D ResNet-50 | 41.6 | 72.2 | I3D ResNet-50 | 41.6 | 72.2 | I3D ResNet-50 | 41.6 | 72.2 |
| res2      | 43.6 | 74.3 | res3-1     | 43.6 | 74.4 | 1-block    | 43.7 | 74.2 |
| res3      | 43.7 | 74.6 | res3-2     | 43.7 | 74.6 | 3-block    | 46.2 | 76.1 |
| res4      | 43.4 | 74.2 | res3-3     | 43.3 | 74.2 | 5-block    | 48.6 | 78.1 |
| res5      | 42.1 | 73.5 | res3-4     | 42.9 | 74.0 | -          | -    | -    |

| 4) Learnable Kernels | 5) Context Field | 6) Activations |
|----------------------|------------------|---------------|
| Model                | Top-1 | Top-5 | Model     | Top-1 | Top-5 | Model     | Top-1 | Top-5 |
| I3D ResNet-50        | 41.6 | 72.2 | I3D ResNet-50 | 41.6 | 72.2 | I3D ResNet-50 | 41.6 | 72.2 |
| 3 × 1 × 1            | 48.0 | 77.1 | 3 × 5 × 5  | 48.2 | 77.3 | softmax    | 48.6 | 78.1 |
| 1 × 3 × 3            | 48.1 | 77.3 | 5 × 5 × 5  | 48.6 | 78.1 | relu       | 48.3 | 77.7 |
| 3 × 3 × 3            | 48.6 | 78.1 | 5 × 7 × 7  | 48.5 | 77.6 | tanh       | 48.4 | 77.9 |
| 3 × 5 × 5            | 48.3 | 77.6 | 7 × 7 × 7  | 48.2 | 77.6 | -          | -    | -    |

1) Training: Unless specified, all the models are trained from scratch. Following [38], we first resize the input frames to the 256 × 320 dimension and then randomly crop 224 × 224 pixels for training. We first train our model with 8-frame input clips randomly sampled in 12 frames per second (FPS) on a 4-GPU machine with a batch size of 64 for 30 epochs, starting with a learning rate of 0.01 and reducing it by a factor of 10 at 15th epoch. Then we fine-tune the model with 32-frame input randomly sampled in 6FPS on an 8-GPU machine with a batch size of 32 for 45 epochs, starting with a learning rate of 0.01 and reducing by a factor of 10 at every 15 epoch.

We use mini-batch stochastic gradient descent with a momentum of 0.9 and a weight decay of 1e-4 for optimization. We use cross entropy loss function for Something-Something V1, V2 and Kinetics-400 datasets, and binary sigmoid loss for Charades datasets (multi-class and multi-label).

2) Inference: At the inference stage, we resize the input frames to the 256 × 320 dimension, randomly sample 40 clips of 32-frame inputs in 6FPS, randomly crop 224 × 224 pixels for testing. The final predictions are based on the the averaged softmax scores of 40 all clips.

B. Experiments on Something-Something V1/V2

Something-Something V1 dataset has 86K training videos, around 12K validation videos and 11K testing videos. The number of classes in this dataset is 174.

Table III shows the ablation results on the validation dataset, analyzed as follows:

1) Higher-order at different stages: We study the network performance when the H-blocks are added to different stages on the network. We add one single H-block after the first bottleneck on 1) res2, 2) res3, 3) res4 and 4) res5 in Table II. As shown in Table III, the improvement of adding one H-block on res3 is the most prominent. The improvement decreases when adding the H-block to deeper stage of the network. One possible explanation is that spatiotemporal correlation weakens as the network getting deeper, since high level features are more linearly separable so higher-order information is less important. One possible reason that higher-order on res2 cannot get the maximum improvement is that the output size of res2 is 8 times larger than the output size of res3, thus the context field is much smaller compared with the entire feature map. An evidence can be found in the following study.

2) Higher-order at different positions of the same stage: We further discuss the performance of adding a single H-block to different positions of the same stage. We add one single H-block after 1) first, 2) second, 3) third and 4) fourth bottleneck within res3. From Table III, We find that adding one H-block after the first and second bottleneck (res3-1 and res3-2) leads to a better accuracy than adding the H-block in res3-3 and res3-4. This again proves that spatiotemporal contexts weakens as the network going deeper, and our single H-block can capture more meaningful spatiotemporal contexts and lose less information than deep stack of convolution layers.

3) Going deeper with H-blocks: Table III shows the results of adding more higher-order blocks. We add 1 block, 3 blocks (2 to Res3 and 1 to Res4) and 5 blocks (3 to Res3 and 2 to Res4) in ResNet-50. More H-blocks in general lead to better results. We argue that multiple higher-order blocks can capture comprehensive contextual information. Messages in each location can be learned with its own context, which is hard to do via shared weights.

4) H-blocks within different kernel sizes: We study how the kernel size would influence the improvement by adding 5 blocks of H-blocks with different kernel sizes and same context field (5 × 5 × 5). As shown in Table III, H-blocks with a kernel size of 3 × 3 × 3 are the best. Smaller and larger will lower the classification accuracy. The reduced performance for the 3 × 5 × 5 may come from the difficulty in optimizing the network due to the larger spatial size.

5) H-blocks with different context fields: We study how the size of context fields influence the improvement by adding 5 blocks of H-blocks with different context fields and same kernel size (3 × 3). In Table I, we show other possible context fields and their factorization using three convolutions. As shown in Table III, The improvement of the H-block with context fields of 5 × 5 × 5 and 5 × 7 × 7 is similar, and a smaller context field of 3 × 5 × 5 as well as a larger context...
**TABLE IV**

**TEST RESULTS ON SOMETHING-SOMETHING V1 DATASET**

| Method                  | Pre-train dataset          | Input size       | Backbone            | Modality | Top1 Acc. (%) |
|-------------------------|----------------------------|------------------|---------------------|----------|---------------|
| Multiscale TRN [39]     | Imagenet                   | multi-input ensemble | Inception + 3D ResNet 18 | RGB+Flow | RGB           | 33.6       |
| ECO [40]                |                            | 16 × 224 × 224   | ResNet 50           | RGB      | 41.6          |
| I3D [1]                 | ImageNet, Kinetics         | 32 × 224 × 224   | ResNet 50           | RGB      | 44.6          |
| NL I3D [18]             | ImageNet, Kinetics         | 32 × 224 × 224   | ResNet 50           | RGB      | 46.1          |
| NL I3D + GCN [38]       | ImageNet, Kinetics         | 32 × 224 × 224   | ResNet 50           | RGB      | 46.1          |
| HO I3D [ours]           | None                       | 32 × 224 × 224   | ResNet 50           | RGB      | 48.2          |
| HO I3D [ours]           | ImageNet, Kinetics         | 32 × 224 × 224   | ResNet 50           | RGB      | 48.2          |

![Fig. 3. Visualizing learned feature map. The upper row of each sample is the feature map of I3D, the bottom row is the feature map of the higher-order network. Videos are from Something-Something V1 dataset, with labels of: (a) Moving something and something closer to each other; (b) Moving something down; (c) Touching (without moving) part of something; (d) Putting something, something and something on the table; (e) Lifting up one end of something without letting it drop down; (f) Tearing something just a little bit.](image)

field of 7 × 7 × 7 is slightly smaller. One possible explanation is that the smaller context field has a small context and it is insufficient to provide precise contextual information. And for larger context field, the context is redundant and more difficult to capture.

6) H-blocks with different activation functions: Instead of using softmax, we also use ReLU and tanh as the last activations. As shown in Table III, different activation functions versions perform similarly, illustrating that activation function of this module is not the key to the improvement in our applications; instead, it is more likely that the higher-order behavior is essential, and it is insensitive to the activation functions.

**Comparison to the state-of-the-art.** We compare the performance with the state-of-the-art approaches on the test set of the Something-Something V1 dataset. The results are summarized in Table IV (HO is short for higher-order). We use a 5 × 5 × 5 context field and a 3 × 3 × 3 kernel size with the softmax activation function. We get a top-1 accuracy of 45.7% without pre-training with other image or video datasets. When pre-trained with ImageNet and Kinetics, our model gets a top-1 accuracy of 48.2%, which is the highest single model result on the leaderboard, surpassing all the existing RGB based methods by a good margin. Note that there are a few related works unlisted such as [41]. They used a deeper backbone such as ResNet101, more input modalities and more input frames, thus are not comparable with our results.

The higher-order operation is a light and efficient module that significantly improves the network performance with limited extra computational cost. If given the 32 frames as the input, I3D model [1], non-local I3D model [18], and our HO I3D model using the ResNet50 backbone (Table II) take...
326G, 401G, and 368G FLOPs respectively.

Fig. 3 visualizes several examples of the feature maps learned by our H-blocks block as well as the I3D ResNet-50 backbone. All the feature maps are from the output of the res5 stage in Table II, resized back to the size of the original videos. In Fig. 3(a) moving something and something closer to each other, our model is focusing simultaneously on two objects and the hands, showing that our model can not only capture appearance information but also capture motion information. In Fig. 3(d) putting something, something and something on the table, we can see evident differences between I3D and H-blocks in the third frame, in which I3D is looking at the red clock, while H-block is focusing on the moving part - hand. From Fig. 3, we can conclude that our higher-order network can learn to find important relation clues instead of focusing on appearance information compared with I3D backbones.

We also investigate our models on Something-Something V2 dataset. The V2 dataset has 22K videos, which is more than twice as many videos as V1. There are 169K training videos, around 25K validation videos and 27K testing videos in the V2 dataset. The number of classes in this dataset is 174, which is the same as the V1 version. Table V shows the comparisons with the previous results on this dataset. When adding five higher-order blocks to res3 and res4 stages, our higher-order ResNet 50 achieves 62.6% Top 1 accuracy.

| Model                      | Backbone   | Top-1 Acc. (%) |
|----------------------------|------------|----------------|
| Multi-Scale TRN [39]       | Inception  | 48.8           |
| 2-Stream TRN [39]          | Inception  | 56.5           |
| HO I3D [ours]              | ResNet 50  | 62.6           |

C. Experiments on Kinetics-400

Kinetics-400 [1] contains approximately 246k training videos and 20k validation videos. It is a classification task involving 400 human action categories. We train all models on the training set and test on the validation set.

Table VI shows the comparisons with the state-of-art on this dataset. We use the best settings from section 4.2, which is 5 H-blocks with 5 × 5 × 5 context field, 3 × 3 × 3 kernel size and softmax activation. Our model achieves a top-1 accuracy of 77.8 and top-5 accuracy of 93.3. Compared with methods that use RGB and Flow, our method can learn motion information end-to-end. Our model is also better than those using RGB only for training.

D. Experiments on Charades dataset

In this subsection we study the performance of higher-order neural networks on the Charades dataset. The Charades dataset is a dataset of daily indoors activities, which consists of 8K training videos and 1.8K validation videos. The average video duration is 30 seconds. There are 157 action classes in this dataset and multiple actions can happen at the same time. We report our results in Table VII. The baseline I3D ResNet 50 approach achieves 31.8% mAP. The best result NL I3D + GCN [38] in Table VII is a combination of two models. By adding 2 H-blocks to res3 and and 3 to res4 stages in the I3D Res50 backbone, our method achieves 5.1% improvements (36.9% mAP) in mAP. And we archive another 0.2% gain (37.1% mAP) by continuously adding 2 H-blocks to res2 stage. The improvement indicates the effectiveness of H-blocks.

V. Conclusion

In this paper, we have introduced higher-order networks to the task of action recognition. Higher-order networks are constructed by a general building block, termed as H-block, which aims to model position-varying contextual information. As demonstrated on the Something-Something (V1 and V2), Kinetics-400 and Charades datasets, the proposed higher-order networks are able to achieve state-of-the-art results, even using only RGB mobility inputs without fine-tuning with other image or video datasets. The good performance may be ascribed to the fact that higher-order networks are a natural for context modeling.

The actual model itself is not restricted to visual tasks, but may be applied in any task where a context governs the interpretation of an input feature, such as cross-modal or multi-modal operations. In future work, we plan to investigate the benefits of our higher-order model and its extensions, in a variety of other visual, text and cross-modal tasks.

REFERENCES

[1] J. Carreira and A. Zisserman, “Quo vadis, action recognition? a new model and the kinetics dataset,” Computer Vision and Pattern Recognition (CVPR), 2017.
[2] F. Mahdisoltani, G. Berger, W. Gharbieh, D. Fleet, and R. Memisevic, “Fine-grained video classification and captioning,” 2018.
[3] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, “Hollywood in homes: Crowdsourcing data collection for activity understanding,” European Conference on Computer Vision (ECCV), 2016.
[4] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, “Learning realistic human actions from movies,” in 2008 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2008, pp. 1–8.
[5] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05). vol. 1. IEEE, 2005, pp. 88–93.
[6] H. Wang, A. Kläser, C. Schmid, and L. Cheng-Lin, “Action recognition by dense trajectories,” in CVPR 2011-IEEE Conference on Computer Vision & Pattern Recognition. IEEE, 2011, pp. 3169–3176.
[7] X. Peng, C. Zou, Y. Qiao, and Q. Peng, “Action recognition with stacked browser vectors,” in European Conference on Computer Vision. Springer, 2014, pp. 581–595.
[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” Neural Information Processing Systems (NIPS), 2012.
[9] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” International Conference on Computer Vision (ICCV), 2015.
[10] Z. Qiu, T. Yao, and T. Mei, “Learning spatio-temporal representation with pseudo-3d residual networks,” International Conference on Computer Vision (ICCV), 2017.
[11] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, “A closer look at spatiotemporal convolutions for action recognition,” Computer Vision and Pattern Recognition (CVPR), 2018.
[12] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, “Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification,” European Conference on Computer Vision (ECCV), 2018.
[13] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” Neural Information Processing Systems (NIPS), 2014.
[14] 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,” European Conference on Computer Vision (ECCV), 2016.
[15] S. Sharma, R. Kiros, and R. Salakhutdinov, “Action recognition using visual attention,” 2015.
[16] R. Girshar and D. Ramanan, “Attentional pooling for action recognition,” in Advances in Neural Information Processing Systems, 2017, pp. 34–45.
[17] A. Miech, I. Laptev, and J. Sicivic, “Learnable pooling with context gating for video classification,” 2017.
[18] X. Wang, R. Girshick, A. Gupta, and K. He, “Non-local neural networks,” in CVPR, 2018.
[19] F. Baradel, C. Wolf, and J. Mille, “Human activity recognition with pose-driven attention to RGB,” in British Machine Vision Conference 2018, BMVC 2018, Newcastle, UK, p. 200.
[20] R. Girshar, J. Carrera, C. Doersch, and A. Zisserman, “Video action transformer networks,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.
[21] M. Marszalek, I. Laptev, and C. Schmid, “Actions in context,” in CVPR 2009-IEEE Conference on Computer Vision & Pattern Recognition. IEEE Computer Society, 2009, pp. 2929–2936.
[22] J. Sun, X. Wu, S. Yan, L.-F. Cheong, T.-S. Chua, and J. Li, “Hierarchical spatio-temporal context modeling for action recognition,” in 2009 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2009, pp. 2004–2011.
[23] A. Kovashka and K. Grauman, “Learning a hierarchy of discriminative space-time neighborhood features for human action recognition,” in 2010 IEEE computer society conference on computer vision and pattern recognition. IEEE, 2010, pp. 2046–2053.
[24] D. L. Vail, M. M. Veloso, and J. D. Lafferty, “Conditional random fields for activity recognition,” in Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems. ACM, 2007, p. 235.
[25] C. Cao, Y. Zhang, and H. Lu, “Spatio-temporal triangular-chain crf for activity recognition,” in Proceedings of the 23rd ACM international conference on Multimedia. ACM, 2015, pp. 1151–1154.
[26] Y. Chen, C. Xiong, R. Xu, and J. J. Corso, “Actionness ranking with lattice conditional ordinal random fields,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 748–755.
[27] A. Quattoni, M. Collins, and T. Darrell, “Conditional random fields for object recognition,” in Advances in Neural Information Processing Systems 17, L. K. Saul, Y. Weiss, and L. Bottou, Eds. MIT Press, 2005, pp. 1097–1104.
[28] D. Wang, W. Ouyang, W. Li, and D. Xu, “Dividing and aggregating network for multi-view action recognition,” in The European Conference on Computer Vision (ECCV), September 2018.
[29] Y. Cao, J. Xu, S. Lin, F. Wei, and H. Hu, “Gencr: Non-local networks meet squeeze-excitation networks and beyond,” in Proceedings of the IEEE International Conference on Computer Vision Workshops, 2019, pp. 0–0.
[30] Z. Qiu, T. Yao, C.-W. Ngo, X. Tian, and T. Mei, “Learning spatio-temporal representation with local and global diffusion,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.
[31] F. Wu, A. Fan, A. Baevski, Y. N. Dauphin, and M. Auli, “Pay less attention with lightweight and dynamic convolutions,” 2019.
[32] L. Deutsch, “Generating neural networks with neural networks,” 2018.
[33] D. Ha, A. Dai, and Q. V. Le, “Hypernetworks,” 2016.
[34] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, “Deformable convolutional networks,” in Proceedings of the IEEE international conference on computer vision, 2017, pp. 764–773.
[35] Z. Qiu, T. Yao, and T. Mei, “Learning spatio-temporal representation with pseudo-3d residual networks,” in Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017.
[36] X. Wang and A. Gupta, “Videos as space-time region graphs,” in Proceedings of the European conference on computer vision (ECCV), 2018, pp. 399–417.
[37] B. Zhou, A. Andonian, A. Oliva, and A. Torralba, “Temporal relational reasoning in videos,” in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 803–818.
[38] M. Zolfaghari, K. Singh, and T. Brox, “Eco: Efficient convolutional network for online video understanding,” in European Conference on Computer Vision (ECCV), 2018.
[39] N. Crasto, P. Weinzaepfel, K. Alahari, and C. Schmid, “Mars: Motion-augmented rgb stream for action recognition,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 7882–7891.
[40] L. Wang, W. Li, W. Li, and L. Van Gool, “Appearance-and-relation networks for video classification,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 1430–1439.
[41] C. Feichtenhofer, H. Fan, J. Malik, and K. He, “Slowfast networks for video recognition,” in Proceedings of the IEEE international conference on computer vision, 2019, pp. 6202–6211.
[42] G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta, “Asynchronous temporal fields for action recognition,” Computer Vision and Pattern Recognition (CVPR), 2017.