LEARNABLE HIGHER-ORDER REPRESENTATION FOR ACTION RECOGNITION

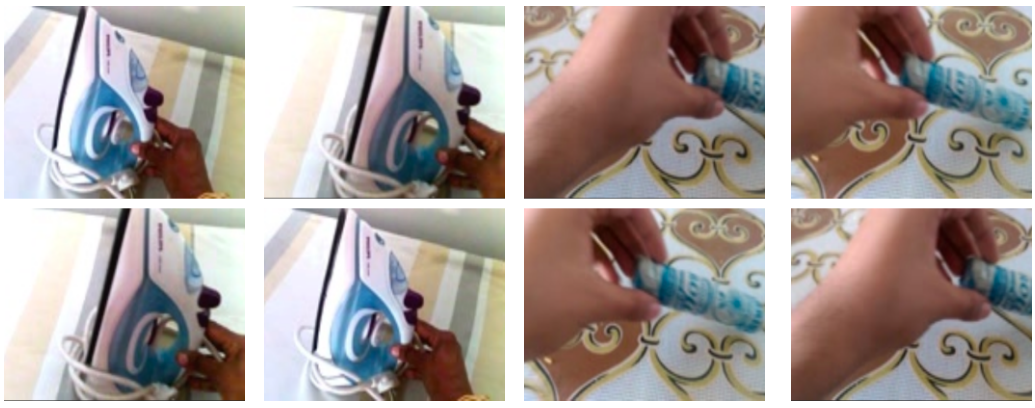
Anonymous authors
Paper under double-blind review

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 higher-order correlations from high dimensional input video space. We prove that several successful architectures for visual classification tasks are in the family of higher-order neural networks; theoretical and experimental analysis demonstrates their underlying mechanism is higher-order. 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 both Something-Something (V1 and V2) and Charades datasets.

1 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. Figure 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.



(a) pull from left to right (b) push from right to left (c) pull from right to left (d) push from left to right

Figure 1: Different contexts of the hand in four different categories. In Figure 1a, since the hand moves from left to right and the hand is on the right side of the iron, it is *pull from left to right*. Figure 1d has the same hand movement, but it is a different category since the hand is on the left of the pen. Figure 1b is a reverse action of Figure 1a, but it is not *pull from right to left*.

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. It would be difficult for conventional convolutions which recognize fixed patterns that are determined by the fixed filter parameters to capture the variety of variations that distinguish the action classes. 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, pushing a pen, pulling an iron, and so on affects the spatio-temporal relations of the involved structures to one another in similar ways. 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 $\boldsymbol{X}$ and $\boldsymbol{Y}$ respectively represent the input and output of a convolution. Let $\boldsymbol{y}_p$ and $\{\boldsymbol{x}_{p'}\}$ represent a specific position of $\boldsymbol{Y}$ and the set of positions of $\boldsymbol{X}$ from which $\boldsymbol{y}_p$ is computed, respectively. Denote conventional convolution operation as $\boldsymbol{Y} = f(\boldsymbol{X}; \Theta)$ where Θ is the shared parameters at different positions. The parameters act as determined feature extractors as $\boldsymbol{y}_p = f(\{\boldsymbol{x}_{p'}\}; \Theta)$ for different positions.

As we analyze, the visual pattern of the target object can vary in different contexts, and determined feature extractors (filters) that ignore this dependence are not optimal. We replace the fixed filters with context-dependent filters $\boldsymbol{y}_p = f(\{\boldsymbol{x}_{p'}\}; \boldsymbol{w}_p)$ where the filters $\boldsymbol{w}_p$ are in turn obtained as $\boldsymbol{w}_p = g(\{\boldsymbol{x}_{p''}\}; \Theta)$. The mapping g is the structure of object-in-context patterns and Θ are the learned parameters as we hypothesize. The entire relation between $\boldsymbol{Y}$ and $\boldsymbol{X}$ can be compactly represented through the higher-order function $\boldsymbol{Y} = f(\boldsymbol{X}; g(\boldsymbol{X}; \Theta))$.

The proposed model is able to capture spatiotemporal contexts effectively. We test our method on four benchmark datasets for action recognition: Kinetics-400 (Carreira & Zisserman, 2017), Something-Something V1 (Mahdisoltani et al., 2018), Something-Something V2, and Charades datasets (Sigurdsson et al., 2016). 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.

2 Related Work

Action Recognition. Many video-action recognition methods are based on high-dimensional encodings of local features. For instance, Laptev et al. (2008) used as local features histograms of oriented gradients (HOG) (Dalal & Triggs, 2005) and histograms of optical flow (HOF) as sparse interest points. The features are encoded into a bag of features representation. Wang et al. (2011); Peng et al. (2014) made use of dense point trajectories which are computed using optical flow. The high performance of 2D ConvNets in image classification tasks (Krizhevsky et al., 2012) makes it appealing to try to reuse them for video recognition tasks. Tran et al. (2015) investigated 3D ConvNets to learn spatiotemporal features end-to-end. Some researchers tried to save computation by replacing 3D convolutions with separable convolutions (Qiu et al., 2017a; Tran et al., 2018) or mixed convolutions (Tran et al., 2018; Xie et al., 2018). Meanwhile, Carreira & Zisserman (2017) introduced an inflation operation. It allows for converting pre-trained 2D models into 3D. Simonyan & Zisserman (2014) designed a two-stream architecture to capture appearance and motion information separately. The spatial stream uses RGB frames as inputs, while the temporal stream learns from stacked optical flow. Wang et al. (2016) further generalized this framework to learn long-range dependencies by temporal segment.

Spatiotemporal Context. Contextual information is very important for action recognition. Galleguillos & Belongie (2010) review different approaches of using contextual information in the field of object recognition. Several methods (Marszałek et al., 2009; Sun et al., 2009; Kovashka & Grauman, 2010; Vail et al., 2007; Cao et al., 2015; Chen et al., 2014) exploit contextual information to facilitate action recognition. Marszałek et al. (2009) exploited the context of natural dynamic scenes for human action recognition in video. Sun et al. (2009) modeled the spatio-temporal relationship between trajectories in a hierarchy of multiple levels. Kovashka & Grauman (2010) proposed to learn the shapes of space-time feature neighborhoods that are most discriminative for a given action category. Conditional Random Field (CRF) models has also been exploited for object and action

recognition (Vail et al., 2007; Cao et al., 2015; Chen et al., 2014; Quattoni et al., 2005; Wang et al., 2018a). Quattoni et al. (2005) propose a CRF framework that incorporates hidden variables for part-based object recognition. Wang et al. (2018a) use a CRF-based approach to exploit the relationship among features from videos captured by cameras from different viewpoints.

3 OUR APPROACH

In the section below we define our second-order model for video analysis. Our model comprises the analysis of video feature maps by a position-dependent bank of spatio-temporal filters, whose filter parameter values are themselves computed through a higher-level function. We first present our notation, and subsequently describe the model itself.

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.

3.1 NOTATION

We denote the *input* video feature map of the H-block as $\boldsymbol{X} \in \mathbb{R}^{C_{in} \times T \times H \times W}$, 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 $\boldsymbol{x}_p$, and $\boldsymbol{x}_p \in \mathbb{R}^{C_{in}}$.

We denote the *output* map for the H-block as $\boldsymbol{Y} \in \mathbb{R}^{C_{out} \times T' \times H' \times W'}$. The description below assumes, for convenience, 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 $\boldsymbol{y}_p$, where $\boldsymbol{y}_p \in \mathbb{R}^{C_{out}}$.

In our model $\boldsymbol{Y}$ is derived from $\boldsymbol{X}$ through a second-order relation of the form $\boldsymbol{Y} = f(\boldsymbol{X}, g(\boldsymbol{X}; \Theta))$ – 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 will 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 (Dai et al., 2017), 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),

$$\mathcal{R} = \Big\{ (t, h, w) \Big| |t| \leq K_t, |h| \leq K_h, |w| \leq K_w \Big\} \tag{1}$$

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

$$\boldsymbol{y}_p = \sum_{q \in \mathcal{R}} \boldsymbol{W}_q \boldsymbol{x}_{p+q}. \tag{2}$$

where $\{\boldsymbol{W}_q, q \in \mathcal{R}\}$ are the weights of convolutional *filters* that scan the input $\boldsymbol{X}$. Each $\boldsymbol{W}_q$ is a matrix: $\boldsymbol{W}_q \in \mathbb{R}^{C_{out} \times C_{in}}$. The convolution outputs are generally further processed by an activation function such as ReLU and *tanh*.

Our H-block retains the same structure as above, except that the convolution operation of Equation 2 now changes to

$$\boldsymbol{y}_p = \sum_{q \in \mathcal{R}} \boldsymbol{W}_{p,q} \boldsymbol{x}_{p+q}. \tag{3}$$

Note that the filter parameters $\boldsymbol{W}_{p,q}$ are now position dependent. The position-dependent filter parameters $\boldsymbol{W}_{p,q}$ are themselves computed using an upper-level function. Representing the entire

set of filter parameters as $\mathcal{W} = \{\boldsymbol{W}_{p,q}\}$, we have

$$\mathcal{W} = g(\boldsymbol{X}; \Theta)$$

The *actual* number of parameters required to define the block is the number of components in Θ. We propose two models for $g(\cdot)$ below, with different requirements for the number of parameters.

3.2 CONVOLUTION-BASED SECOND-ORDER OPERATION

In the convolution-based model for $g(\cdot)$, we derive the filter parameters $\{\boldsymbol{W}_{p,q}\}$ through convolutions. Since the total number of parameters in $\{\boldsymbol{W}_{p,q}\}$ can get very large, we restrict each $\boldsymbol{W}_{p,q}$ to be a diagonal matrix, which can equivalently be represented by the vector $\boldsymbol{w}_{p,q}$. It is similar to a depth-wise convolution. Equation 3 can now be rewritten as

$$\boldsymbol{y}_p = \sum_{q \in \mathcal{R}} \boldsymbol{w}_{p,q} \otimes \boldsymbol{x}_{p+q}. \tag{4}$$

where $\otimes$ represents a component-wise (Schur) multiplication.

The filter parameters to estimate are $\boldsymbol{x}_{p,q}$ are derived from $\boldsymbol{X}$ through a convolution operation as

$$\boldsymbol{w}_{p,q} = \sum_{t \in \mathcal{R}'} \Theta_t^q \boldsymbol{x}_{p+t} \tag{5}$$

where $\mathcal{R}'$ (like $\mathcal{R}$) is the receptive field for the convolutional filters, and represents the *context field*. The size of $\mathcal{R}'$ represents the span from which context is captured to compute any single $\boldsymbol{y}_p$, and must ideally be larger than, or at least no smaller than the size of $\mathcal{R}$. Θ_t^q are the convolutional filter parameters. Each Θ_t^q is a $C_{in} \times C_{in}$ matrix. The complete set of parameters of $g()$ are given by $\Theta = \{\Theta_t^q,\ q \in \mathcal{R},\ t \in \mathcal{R}'\}$, 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 Θ 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.

3.3 CNN-BASED SECOND-ORDER OPERATION

In the CNN-based second-order block, we use a full convolutional neural network comprising multiple layers of convolutions followed by activations to compute $\boldsymbol{w}_{p,q}$. Representing $\boldsymbol{w}^q = \{\boldsymbol{w}_{p,q},\ \forall p\}$, we can write

$$\boldsymbol{w}^q = CNN\left(\boldsymbol{X}, \Theta_q\right), \tag{6}$$

where Θ_q are the parameters of the CNN. The filters and their strides in the CNN of Equation 6 are designed such that the receptive field of the CNN (representing the context field) is larger than $\mathcal{R}$.

Since CNNs require multiple layers of convolutions, the number of parameters in Equation 6 is apparently larger than that required by the simple convolutional model of . However, by appropriately structuring the CNNs we can, in fact, arrive at a model that requires far fewer parameters than the simple convolution model. For instance, a CNN with two layers, each computed by a 3×3 (height $\times$ width) filter provides a context field of 5×5 using only 18 parameters, whereas a single convolution would require a 5×5 filter with 25 parameters to provide the same context field. Furthermore, by appropriately *sharing* parameters across the CNNs for the different $q \in \mathcal{R}$, the actual number of parameters required can be greatly reduced.

In our implementations we implement the shared computation through a single multi-output CNN. The CNN comprises a series of M (e.g. 3) shared convolutional layers (with activations). The last shared layer is operated on by a bank of $|\mathcal{R}|$ 1×1 convolutions to derive $\boldsymbol{w}^q,\ q \in \mathcal{R}$. Figure 2 illustrates this structure.

4 EXPERIMENTS

We perform comprehensive studies on the challenging Something-Something V1 dataset (Mahdisoltani et al., 2018), and also report results on the Charades (Sigurdsson et al., 2016), Kinetics-400 (Carreira & Zisserman, 2017) and Something-Something V2 dataset to show the generality of our models.

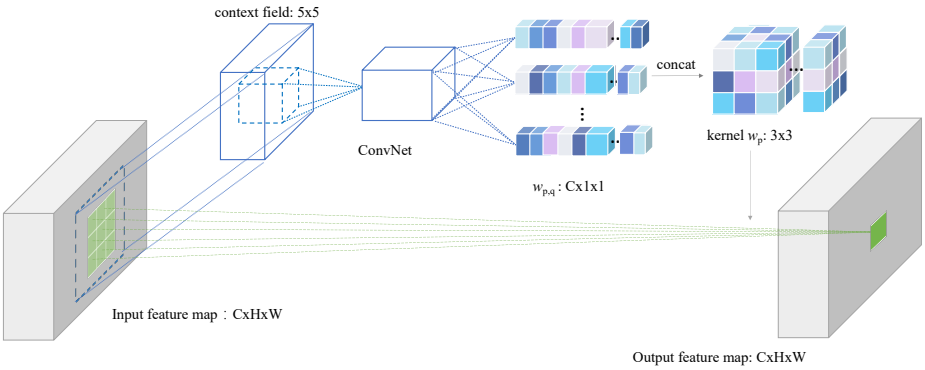


Figure 2: One example of a second-operation on 2D data with channel number C, width W, and height H. For every position p in the feature map (HW positions in total), the ConvNet derives 9 C-dimensional vectors. They are concatenated into a $C \times 3 \times 3$ filter to extract the p^{th} output feature from a 3×3 region of the input feature map centered at p.

4.1 Implementation Details

To draw fair comparison with the results in (Wang & Gupta, 2018) on the same datasets, our backbone model is based on the ResNet-50 Inflated 3D architecture (Table 7 in Appendix) and is the same as that in (Wang & Gupta, 2018) . Note there are small differences between our backbone model with the Inflated 3D backbone in (Wang et al., 2018c) where the output of the last convolutional layer is a $^T/_2 \times 14 \times 14$ feature map (T is the number of input frames).

H-blocks. Following (Qiu et al., 2017b), we use 3 layers of Pseudo-3D (P3D) convolutions to implement the ConvNet $CNN(\cdot, \Theta_q)$ in Equation 6 for obtaining a sufficiently large context field. Table 6 in Appendix shows the kernel size of three P3D convolutions as the factorization of different context fields. 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, $C//|\mathcal{R}| \times |\mathcal{R}|$) and ($C//|\mathcal{R}| \times |\mathcal{R}|$,$C \times |\mathcal{R}|$) respectively ($//$ is integer divison, for example $19//9 = 2$). The last convolution is a group convolution with group size $|\mathcal{R}|$ to reduce parameters. After each convolution layer, we use the scaled exponential linear unit (SELU) (Klambauer et al., 2017) as the activation. The last convolution is always a group convolution (Xie et al., 2017) with group size $|\mathcal{R}|$ to reduce parameters. And we use *softmax* as the last convolution's activation as a normalization factor.

Training. Unless specified, all the models are trained from scratch. Following (Wang & Gupta, 2018), 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 15^{th} 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).

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.

4.2 Experiments on Something-Something V1

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 1: Ablations on Something-Something V1 action classification.

(a) **Stage**

model	top-1	top-5
I3D	41.6	72.2
res2-1	43.6	74.3
res3-1	43.7	74.6
res4-1	43.4	74.2
res5-1	42.1	73.5

(b) **Position within one stage**

model	top-1	top-5
I3D	41.6	72.2
res3-1	43.6	74.4
res3-2	43.7	74.6
res3-3	43.3	74.2
res3-4	42.9	74.0

(c) **Number of blocks added**

model	top-1	top-5
I3D	41.6	72.2
1-block	43.7	74.2
3-block	46.2	76.1
5-block	48.6	78.1

(d) **Kernel Size**

model	top-1	top-5
I3D	41.6	72.2
$3 \times 1 \times 1$	48.0	77.1
$1 \times 3 \times 3$	48.1	77.3
$3 \times 3 \times 3$	48.6	78.1
$3 \times 5 \times 5$	48.3	77.6

(e) **Context Field**

model	top-1	top-5
I3D	41.6	72.2
$3 \times 5 \times 5$	48.0	77.1
$1 \times 3 \times 3$	48.1	77.3
$3 \times 3 \times 3$	48.6	78.1
$3 \times 5 \times 5$	48.3	77.6

(f) **Activations**

model	top-1	top-5
I3D	41.6	72.2
softmax	48.6	78.1
ReLu	48.3	74.6
tanh	48.4	77.9

Table 1 shows the ablation results on the validation dataset, analyzed as follows:

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 7 (in Appendix). As shown in Table 1a, 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 going deeper, since high level features are more linear 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.

Higher-order at different positions of the same stage. We further discuss the performance of adding one 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 1b, 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.

Table 2: Comparisons with state-of-the-art results on Something-Something V1 dataset.

Method	Pre-train dataset	Backbone	val	test
2-stream TRN (Zhou, 2018)	Imagenet	BN-Inception	42.0	40.7
ECO (Zolfaghari et al., 2018)	ImageNet,Kinetics	BN-Inception-Res18	49.5	43.9
I3D (Wang & Gupta, 2018)	ImageNet,Kinetics	ResNet 50	41.6	-
NL I3D (Wang & Gupta, 2018)	ImageNet,Kinetics	ResNet 50	44.6	-
NL I3D + GCN (Wang & Gupta, 2018)	ImageNet,Kinetics	ResNet 50	46.1	45.0
HO I3D [ours]	None	ResNet 50	48.6	44.7
HO I3D [ours]	ImageNet,Kinetics	ResNet 50	51.2	46.7

Going deeper with H-blocks. Table 1c shows the results of adding more higher-order blocks. We add 1 block (to res3), 3 block (1 to res3 and 2 to res4), 5 blocks (3 to res4 and 2 to res3, to every other residual block) 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.

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 ($5 \times 5 \times 5$).

As shown in Table 1d, H-blocks with a kernel size of $3 \times 3 \times 3$ is the best, smaller and larger kernel lower the classification accuracy. The reduced performance for the $3 \times 5 \times 5$ may come from the optimization difficulties because of the large spatial size. **H-blocks with different context fields.**

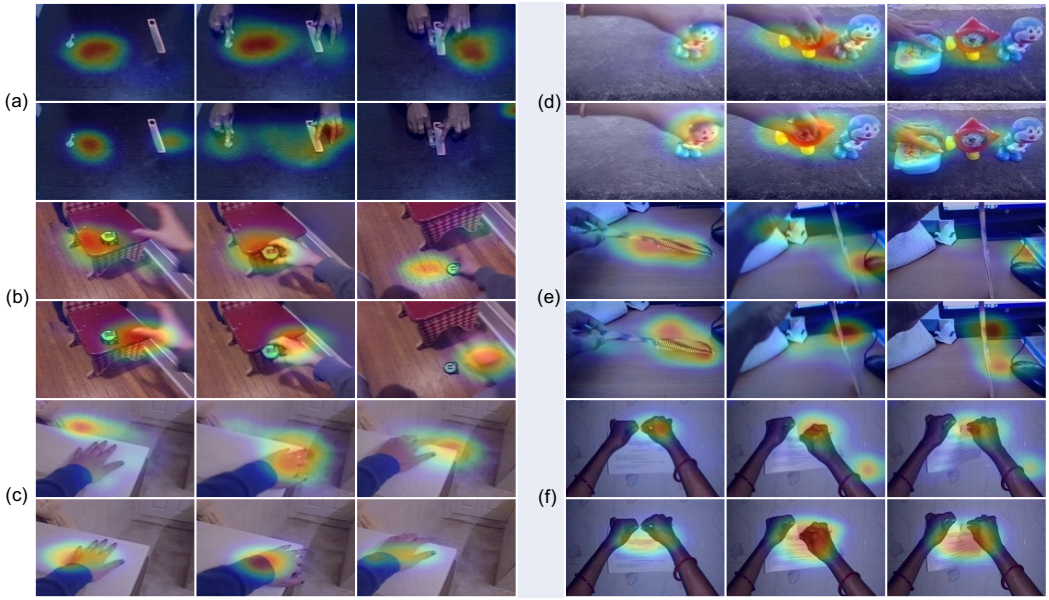


Figure 3: Visualize Learned feature map. The upper row of each sample is feature map of I3D, the bottom raw is feature map of higher-order. Videos are from Something-Something V1 dataset, with labels of: (a) Moving something and something closer to each other; (b) Moving something down; (c) Touching (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.

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 6 (in Appendix), we show other possible context fields and their factorization using three convolutions. As shown in Table 1e, The improvement of a H-blocks block with a context field of $5 \times 5 \times 5$ and $5 \times 7 \times 7$ is similar, and a smaller context field of $3 \times 5 \times 5$ as well as a larger context field of $7 \times 7 \times 7$ is slightly smaller. One possible explanation is that 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 for capturing contextual information.

H-blocks with different activation functions. Instead of use *softmax*, we also use ReLU and *tanh* as the last activations. As shown in Table 1f, 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 Something-Something V1 dataset. The results are summarized in Table 2 (HO is short for higher-order). We use the settings of $5 \times 5 \times 5$ context field and $3 \times 3 \times 3$ kernel size with the activation function of *softmax*. We get a top-1 accuracy of 44.7% without pre-training with other image or video datasets. And when pre-trained with ImageNet and Kinetics, our model gets a top-1 accuracy of 46.7%, which is the highest single model result on leaderboard, surpassing all the existing RGB or RGB + flow based methods by a good margin.

Figure 3 visualize several examples of the feature maps learned by our H-blocks block as well as I3D ResNet-50 backbone. All the feature maps are from the output of res5 stage in Table 7 (in Appendix) and resized back to the size of original videos. In Figure 3(a) *moving something and something closer to each other*, our model is focusing simultaneously on two objects and hands, showing that our model can not only capture appearance information but also capture motion information. In

Figure 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 Figure 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.

4.3 EXPERIMENTS ON KINETICS-400

Kinetics-400 (Carreira & Zisserman, 2017) 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 3 shows the comparisons with the state-of-arts on this dataset. We use the best settings from section 4.2, which is 5 H-blocks with $5 \times 5 \times 5$ context field, $3 \times 3 \times 3$ kernel size and *softmax* activation. Our model archives a top-1 accuracy of 77.8 and top-5 accuracy of 93.3. Compared with methods of using RGB and Flow, our method can learn motion information end-to-end. Our model is also better than those using RGB only for training.

Table 3: Validation results on Kinetics-400 dataset

Method	Backbone	Top-1	Top-5
ARTNet (Wang et al., 2018b)	ResNet 18	69.2	88.3
I3D (Carreira & Zisserman, 2017)	BN-Inception	71.1	89.3
2-stream I3D (Carreira & Zisserman, 2017)	BN-Inception	74.2	91.3
2-stream R(2+1)D (Tran et al., 2018)	ResNet 50	73.9	90.9
NL I3D (Wang et al., 2018c)	ResNet 50	76.5	92.6
NL I3D (Wang & Gupta, 2018)	ResNet 101	77.7	93.3
SlowFast (C. Feichtenhofer & He, 2018)	ResNet 50	77.0	92.6
NL SlowFast(C. Feichtenhofer & He, 2018)	ResNet 50	77.7	93.1
HO I3D [ours]	ResNet 50	77.8	93.3

4.4 EXPERIMENTS ON OTHER VIDEO DATASETS

In this subsection we study the performance of higher-order neural networks on 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 5. The baseline I3D ResNet 50 approach achieves 31.8% mAP. The best result NL I3D + GCN (Wang & Gupta, 2018) in Table 5 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 archives 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.

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 V1 version. Table 4 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.

Table 4: Validation results on Something Something V2 Dataset

Method	Top-1
Multi-Scale TRN (Zhou, 2018)	48.8
2-Stream TRN (Zhou, 2018)	55.5
HO I3D [ours]	62.6

Table 5: Validation results on the Charades dataset

model	mAP
I3D (Wang et al., 2018c)	31.8
NL I3D (Wang et al., 2018c)	33.5
GCN (Wang & Gupta, 2018)	36.2
NL I3D + GCN (Wang & Gupta, 2018)	37.5
HO I3D [ours]	37.1

5 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

J. Malik C. Feichtenhofer, H. Fan and K. He. Slowfast networks for video recognition. *arXiv preprint arXiv:1812.03982*, 2018.

Congqi Cao, Yifan Zhang, and Hanqing Lu. Spatio-temporal triangular-chain crf for activity recognition. In *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 1151–1154. ACM, 2015.

J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. *Computer Vision and Pattern Recognition (CVPR)*, 2017.

Wei Chen, Caiming Xiong, Ran Xu, and Jason J Corso. Actionness ranking with lattice conditional ordinal random fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 748–755, 2014.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 764–773, 2017.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, volume 1, pp. 886–893. IEEE Computer Society, 2005.

Carolina Galleguillos and Serge Belongie. Context based object categorization: A critical survey. *Computer vision and image understanding*, 114(6):712–722, 2010.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pp. 971–980, 2017.

Adriana Kovashka and Kristen 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*, pp. 2046–2053. IEEE, 2010.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems (NIPS)*, 2012.

Ivan Laptev, Marcin Marszałek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. 2008.

F. Mahdisoltani, G. Berger, W. Gharbieh, D. Fleet, and R. Memisevic. Fine-grained video classification and captioning. *arXiv preprint arXiv:1804.09235*, 2018.

Marcin Marszałek, Ivan Laptev, and Cordelia Schmid. Actions in context. In *CVPR 2009-IEEE Conference on Computer Vision & Pattern Recognition*, pp. 2929–2936. IEEE Computer Society, 2009.

Xiaojiang Peng, Changqing Zou, Yu Qiao, and Qiang Peng. Action recognition with stacked fisher vectors. In *European Conference on Computer Vision*, pp. 581–595. Springer, 2014.

Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. *International Conference on Computer Vision (ICCV)*, 2017a.

Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*, pp. 5533–5541, 2017b.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional random fields for object recognition. In L. K. Saul, Y. Weiss, and L. Bottou (eds.), *Advances in Neural Information Processing Systems 17*, pp. 1097–1104. MIT Press, 2005.

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.

K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. *Neural Information Processing Systems (NIPS)*, 2014.

Ju Sun, Xiao Wu, Shuicheng Yan, Loong-Fah Cheong, Tat-Seng Chua, and Jintao Li. Hierarchical spatio-temporal context modeling for action recognition. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2004–2011. IEEE, 2009.

D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d con- volutional networks. *International Conference on Computer Vision (ICCV)*, 2015.

D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotem- poral convolutions for action recognition. *Computer Vision and Pattern Recognition (CVPR)*, 2018.

Douglas L Vail, Manuela M Veloso, and John D Lafferty. Conditional random fields for activity recognition. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pp. 235. ACM, 2007.

Dongang Wang, Wanli Ouyang, Wen Li, and Dong Xu. Dividing and aggregating network for multi-view action recognition. In *The European Conference on Computer Vision (ECCV)*, September 2018a.

Heng Wang, Alexander Kläser, Cordelia Schmid, and Liu Cheng-Lin. Action recognition by dense trajectories. In *CVPR 2011-IEEE Conference on Computer Vision & Pattern Recognition*, pp. 3169–3176. IEEE, 2011.

L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. *European Conference on Computer Vision (ECCV)*, 2016.

Limin Wang, Wei Li, Wen Li, and Luc Van Gool. Appearance-and-relation networks for video classification. 2018b.

X. Wang and A. Gupta. Videos as space-time region graphs. 2018.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018c.

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.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

Andonian A. Torralba A. Zhou, B. Temporal relational reasoning in videos. *European Conference on Computer Vision (ECCV)*, 2018.

M. Zolfaghari, K. Singh, and T. Brox. Eco: Efficient convolutional network for online video understanding. In *European Conference on Computer Vision (ECCV)*, 2018.

A APPENDIX

Table 6 shows the factorization of different context fields. For example, we stack three convolutions with kernel size $3 \times 3 \times 3$ to get a $7 \times 7 \times 7$ context field.

Table 6: Factorization of different context fields.

context field	layer 1	layer 2	layer 3
$3 \times 3 \times 3$	$1 \times 3 \times 3$	$3 \times 1 \times 1$	$1 \times 1 \times 1$
$3 \times 5 \times 5$	$1 \times 3 \times 3$	$3 \times 3 \times 3$	$1 \times 1 \times 1$
$5 \times 5 \times 5$	$1 \times 3 \times 3$	$3 \times 3 \times 3$	$3 \times 1 \times 1$
$5 \times 7 \times 7$	$1 \times 3 \times 3$	$3 \times 3 \times 3$	$3 \times 3 \times 3$
$7 \times 7 \times 7$	$3 \times 3 \times 3$	$3 \times 3 \times 3$	$3 \times 3 \times 3$

Table 7 shows our backbone ResNet-50 I3D model. We use T×H×W to represent the dimensions of kernels and output feature maps. $T = \{8, 32\}$, and the corresponding input size is 8×224×224 and 32×224×224.

Table 7: Our backbone ResNet-50 I3D model.

layer		output size
conv_1	$5 \times 7 \times 7$, 64, stride 1,2,2	$T \times 112 \times 112$
pool_1	$1 \times 3 \times 3$, max, stride 1,2,2	$T \times 56 \times 56$
res_2	$\begin{bmatrix} 3 \times 1 \times 1, 64 \\ 1 \times 3 \times 3, 64 \\ 1 \times 1 \times 1, 256 \end{bmatrix} \times 3$	$T \times 56 \times 56$
pool_2	$3 \times 1 \times 1$, max, stride 2,1,1	$\frac{T}{2} \times 56 \times 56$
res_3	$\begin{bmatrix} 3 \times 1 \times 1, 128 \\ 1 \times 3 \times 3, 128 \\ 1 \times 1 \times 1, 512 \end{bmatrix} \times 4$	$\frac{T}{2} \times 28 \times 28$
res_4	$\begin{bmatrix} 3 \times 1 \times 1, 256 \\ 1 \times 3 \times 3, 256 \\ 1 \times 1 \times 1, 1024 \end{bmatrix} \times 6$	$\frac{T}{2} \times 14 \times 14$
res_5	$\begin{bmatrix} 3 \times 1 \times 1, 512 \\ 1 \times 3 \times 3, 512 \\ 1 \times 1 \times 1, 2048 \end{bmatrix} \times 3$	$\frac{T}{2} \times 14 \times 14$
global average pool and fc		1×1×1