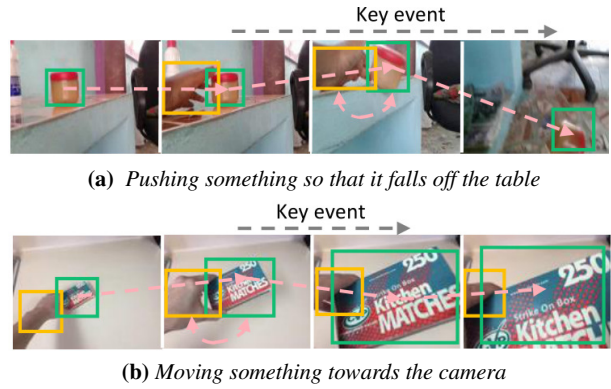# EAN: Event Adaptive Network for Enhanced Action Recognition

Yuan Tian, Yichao Yan ✉**, Guangtao Zhai** ✉**, Guodong Guo, and Zhiyong Gao**

Abstract Efficiently modeling spatial-temporal information in videos is crucial for action recognition. To achieve this goal, state-of-the-art methods typically employ the convolution operator and the dense interaction modules such as non-local blocks. However, these methods cannot accurately fit the diverse events in videos. On the one hand, the adopted convolutions are with fixed scales, thus struggling with events of various scales. On the other hand, the dense interaction modeling paradigm only achieves sub-optimal performance as action-irrelevant parts bring additional noises for the final prediction. In this paper, we propose a unified action recognition framework to investigate the dynamic nature of video content by introducing the following designs. First, when extracting local cues, we generate the spatial-temporal kernels of dynamic-scale to adaptively fit the diverse events. Second, to accurately aggregate these cues into a global video representation, we propose to mine the interactions only among a few selected foreground objects by a Transformer, which yields a sparse paradigm. We call the proposed framework as *Event Adaptive Network* (EAN) because both key designs are adaptive to the input video content. To exploit the short-term motions within local segments, we propose a novel and efficient *Latent Motion Code* (LMC) module, further improving the performance of the framework. Extensive experiments on several large-scale video datasets, *e.g.*, Something-to-Something V1&V2, Kinetics, and Diving48, verify that our models achieve state-of-the-art or competitive performances at low FLOPs. *Codes are available at:* `https://github.com/tianyuan168326/EAN-Pytorch`.

Y. Tian, G. Zhai, and Z. Gao are with the Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University, Shanghai, China. E-mail: {ee_tianyuan,zhaiguangtao,zhiyong.gao}@sjtu.edu.cn. Y. Yan is with the AI Institute, Shanghai Jiao Tong University, Shanghai, China. E-mail: yanyichao@sjtu.edu.cn. G. Guo is with Baidu. E-mail: guoguodong01@baidu.com. ✉ denotes the corresponding author.

(a) *Pushing something so that it falls off the table*

(b) *Moving something towards the camera*

Fig. 1: Two examples from the Something-Something dataset [21]. The objects (*i.e.*, can, box, and hand) and events have diverse spatial-temporal scales in different videos. Therefore, convolution kernels with adaptive scales can better fit them. Moreover, the interactions among them are naturally sparse, which can be accurately and efficiently modeled by a dedicated sparse model. The objects, object interactions, and key events are indicated by the colored boxes, pink dotted arrows, and gray dotted arrows, respectively.

Keywords Action recognition · Dynamic neural networks · Vision Transformers · Motion representation

1 Introduction

Video action recognition is an open challenge in computer vision, drawing increasing attention in both research and industrial communities, because of its fundamental role for tremendous applications, *e.g.*, human behavior monitoring [9] [7] [51], video surveillance [17], anomaly events analysis [2] [35], to name a few. It goes beyond the recognition performed on single images and depends on comprehensively

modeling both (1) the local spatial-temporal cues and (2) the global object interactions in videos.

Many previous methods [60] [32] [70] [53] [6] [54] [14] achieve promising performance by only modeling the local spatial-temporal cues. However, these networks are typically built with convolutions, whose scales are usually empirically determined and kept fixed for different input videos. Indeed, by designing multi-scale networks, such as in ResNet [23], Inception networks [48], and Res2Net [18], the models are equipped with convolution kernels of diverse scales. But, these architectures are still static, not adapting to the various events within videos. We illustrate this challenge in Fig. 1. There naturally arises a question - *can we design a dynamic architecture that adaptively fits the events in each video?*

Additionally, recognizing the actions in videos needs to reason about the interactions among the objects. Although the local interactions can be well captured by the convolutions, there are always some non-local interactions that can only be observed from a global view. For example, in Fig. 1 (a), the key interaction is "the can is moved towards the ground across several frames". Modeling interactions like this requires global reasoning capability, which is beyond the function of convolution. To model the global information, dense interaction models [61] [4] [13] calculate the paired correlations at all positions, which inevitably introduce the background noise signals. In contrast, the sparse models [62] [39] are more accurate because they only target the action-relevant regions. Nevertheless, the inefficiency and the error accumulation caused by their embedded object detector are nontrivial to resolve. Moreover, the utilized heavy detector hinders the end-to-end training of the whole system. Therefore, there arises another question - *can we model the global object interactions sparsely without relying on a heavy object detector?*

In this paper, we answer both questions with *yes*, by carefully designing several spatial-temporal modeling modules. *First*, we propose an Event Adaptive Block (EAB) to enhance the convolution operators with scale-adaptive modeling capability. Particularly, this block perceives the scale information of the key events within the input video, and then dynamically synthesizes the spatial-temporal kernel. Since the scale of the kernel is not fixed, it is unfeasible to represent it with a single trainable tensor. Instead, we reformulate it as a soft fusion of several fixed-scale spatial- or temporal-convolution kernels. Since the synthesized kernel is customized to the input video, the local event cues within the video are better modeled. Moreover, the prevalent architectures, *e.g.*, R(2+1)D CNNs [54] and Inception-Nets [48] can be viewed as a special case of the proposed EAB. *Second*, we propose a Sparse Object Interaction Transformer (SOI-Tr) to build sparse interaction graphs by adaptively selecting the most important objects involved in the actions. Concretely, given the deep video features, an embedded ob-

ject localization network first outputs several saliency maps, each of which corresponds to an object. Then, a shallow Transformer [55] is used to model the long-range interactions among this small number of objects. Thanks to the feature-level detection scheme, this module gets rid of the heavy detector and is end-to-end trainable, which is more effective and efficient than the previous models [62] [39].
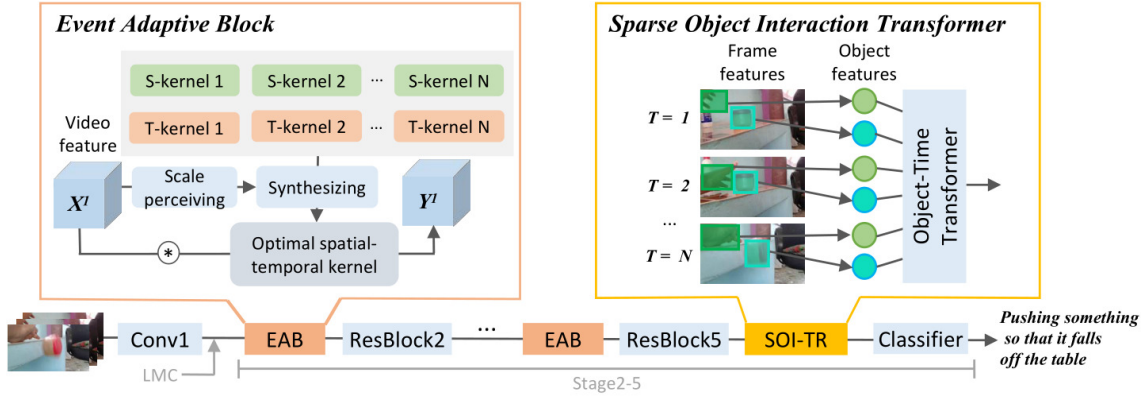
In addition to the two spatial-temporal modeling modules above, we further propose a novel Latent Motion Code (LMC) module to efficiently exploit the short-term motion information within local video segments. Specifically, the low-level motion cues within each segment, *i.e.*, RGB differences, are first encoded into a compact latent space. Then, the high-order motion information is reasoned in this space. The motion information further facilitates the discriminating capability of our method for some hard action cases.

We incorporate the proposed three modules into a unified ConvNet called Event Adaptive Network (EAN). By following a series of efficient network designs, the proposed EAN is highly efficient. The whole framework can be jointly optimized following the sparse sampling strategy proposed in TSN [60]. We emphasize our contributions as follows:

- A novel Event Adaptive Block (EAB) is proposed to generate the video-adaptive spatial-temporal convolution kernel of dynamic scale, demonstrating superior local spatial-temporal modeling capability. Moreover, our approach is the very first work to generate dynamic spatial-temporal convolution kernels for video data.
- A Sparse Object Interaction Transformer (SOI-Tr) is developed to accurately reason the global interactions among the sparse foreground objects, without relying on bounding box annotations or external object detectors.
- A novel and efficient Latent Motion Code (LMC) module is devised to capture the short-term motion information within local video segments in a latent space.
- By incorporating the proposed EAB, SOI-Tr, and LMC into the off-the-shelf 2D CNNs, *i.e.*, 2D ResNet, we build up a strong yet efficient video action recognition framework called Event Adaptive Network (EAN). Our models achieve state-of-the-art or competitive results on several large-scale video datasets, *i.e.*, Something-Something V1&V2 [21], Kinetics [6], and Diving48 [31].

2 Related work

Deep Action Recognition. Two-stream CNNs [43] [16] [15] are the earliest works on deep action recognition. Later, many methods [60] [70] [32] [34] [33] [36] [30] [58] [63] [28] [50] enhance the 2D CNNs with various temporal modules and achieve promising results. To simultaneously learn the temporal dynamics along with the spatial representations in videos, 3D networks, *e.g.*, C3D network [53], I3D [6], 3D-

Fig. 2: Event Adaptive Network. Our framework aims to simultaneously model local spatial-temporal information and global object interactions by incorporating two novel modules, *i.e.*, Event Adaptive Block (EAB) and Sparse Object Interaction Transformer (SOI-Tr), into the 2D ResNet backbone CNN. The EAB first perceives the scale of local events and then dynamically synthesizes video-adaptive spatial-temporal kernels from spatial (S) or temporal (T) kernels of fixed scales. The SOI-Tr specializes in global interactions among sparse foreground objects by leveraging a Transformer. Besides, a latent motion code (LMC) module is adopted to efficiently exploit short-term motion information within local segments. Our proposed framework is an end-to-end hybrid model that uses both convolution and self-attention.

ResNet [22] [49], R(2+1)D CNNs [54], and Slowfast networks [14], have also recently gained much attention. Our framework is built upon the 2D CNNs due to their better efficiency.

Multi-scale CNNs. Many modern image CNN architectures, *e.g.*, Inception networks [47] [48] [46], Res2Net [18], incorporate the multi-scale design for obtaining better representations. For the video tasks, Zhang *et al.* [67] proposed the various-timescale inference pooling to observe videos across various timescales. TEA [30] extends the Res2Net block with temporal modeling capability. However, all these architectures are static, while our method is dynamic and adaptive to the input video.

Dynamic Convolution. Jia *et al.* [25] first proposed the concept of dynamic filter. Latter, several works [65] [8] in image tasks attempt to dynamically generate aggregation weights and use them to combine a set of convolutional kernels. More recently, TANet [34] generalizes this idea to temporal modeling for the video recognition task. However, the generated temporal kernel is shared across all channels, demonstrating limited modeling capability and performance. In contrast, our method generates the full *spatial-temporal* kernel, whose parameters are specified for each channel.

Object Interaction models. Ma *et al.* [37] employed the LSTM to build the object-object interaction graph. Wang *et al.* [62] utilized the Graph Convolutional Networks (GCNs) to perform object relationship reasoning. Materzynska *et al.* [39] proposed a sparse semantically grounded subject-object graph representation. All these methods rely on an object detector or external bounding box annotations to determine the regions of the objects. Without leveraging any explicit object region information, Non-local Neural Net-

works [61] try to model every pairwise interaction *densely* in the feature space.

Vision Transformer. Recently, many works [12] [52] [4] [13] [20] [1] [69] [5] [10] apply the Transformer architecture [55] to image/video tasks by unfolding the visual signal or its feature map to a sequence of tokens. Although their global modeling capability is inherently superior to the convolution-based methods, these models are computationally-expensive due to the dense self-attention mechanism. Similar to us, Girdhar *et al.* [19] and Plizzari *et al.* [40] also leverage a lightweight Transformer architecture to model the interactions among the selected key regions of the input video. However, they either rely on an object region proposal network (RPN) to produce dense proposal regions or an external computationally-heavy keypoint extractor to locate the human keypoints. In contrast, our object representation is sparse and is produced by a lightweight three-layer CNN.

Short-term Motion Representation. Previous works of two-stream action recognition frameworks [43] [16] use optical flow maps as a complement to RGB inputs. Both conventional or CNN-based optical flow estimation methods [66] [24] [41] [45] can be adopted in this framework. Recent works [67] [58] propose several lightweight modules to produce task-specific short-term motion representations, which can be jointly optimized with the action recognition network. For example, PAN [67] proposes to use the difference of the low-level features between the adjacent frames as a novel motion cue named Persistence of Appearance (PA). More recently, TDN [58] uses a short-term module to map the RGB difference motion signals into compact features and fuse the features with that produced by the backbone network. In contrast, our proposed latent motion code

(LMC) module exploits high-order motion information in a latent space, which is more effective and also efficient.

3 Approach

In this work, we propose a novel video action recognition framework called Event Adaptive Network (EAN), as shown in Fig. 2. The network is built by inserting several Event Adaptive Blocks (EABs) and a Sparse Object Interaction Transformer (SOI-Tr) into different stages of the 2D ResNet backbone CNN. Moreover, a Latent Motion Code (LMC) module is adopted to exploit the short-term motion information within local video segments. All components in our framework are differentiable and the proposed EAN is end-to-end trainable.

3.1 Event Adaptive Block

Event Adaptive Block (EAB) aims to generate the spatial-temporal kernel to adaptively model the local cues within the input video, as shown in Fig. 3. We start from an approximated formulation of the optimal kernel for the video, and then implement the formulation as an efficient block.
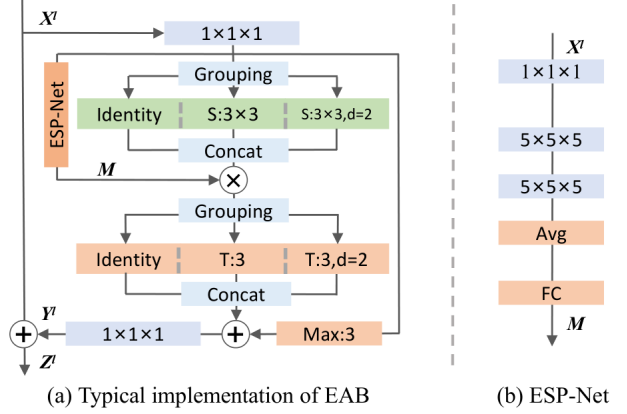
An approximated formulation for the optimal kernel. Formally, given an input video feature X^l with channel number C, which is the output of the l-th ($l \in [1, 4]$) stage of the backbone CNN. We first assume that there exists an optimal spatial-temporal kernel $\hat{F}$ that accurately fits the key elements (*i.e.*, objects and events) of the video. This kernel transforms the input X^l into an output tensor Y^l of the same shape by convolution:

$$Y^l = X^l * \hat{F}. \tag{1}$$

Both the scale and the parameters of the $\hat{F}$ can adapt to videos with different contents. Because the accurate shape of $\hat{F}$ is unknown, we cannot easily implement it as a trainable fixed-scale convolution kernel. Instead, we propose to solve the surrogate problem, *i.e.*, approximating the produced Y^l. We achieve this by leveraging a group of fixed-scale spatial or temporal convolutions:

$$Y^l = \cup_{i=1}^{G} \{ M \otimes \cup_{j=1}^{G} X_j^l * F_s^{(2j-1)} \}_i * F_t^{(2i-1)}, \tag{2}$$

where $F_s^{(2j-1)}$ and $F_t^{(2i-1)}$ represent the spatial convolution with kernel size $(2j-1) \times (2j-1)$ and the temporal convolution with kernel size $2i-1$, respectively. Each convolution is performed on a group of features for reducing the computation cost. G denotes the group number, $\cup$ denotes the channel concatenating operation, and $X_j^l = X^l[j \cdot c : (j+1) \cdot c]$, where $c = C/G$. $\otimes$ denotes the channel-wise broadcasting matrix multiplication operation. M denotes the fusion matrix that relates the spatial and temporal convolutions, and

(a) Typical implementation of EAB   (b) ESP-Net

Fig. 3: (a) **EAB** of maximum receptive filed size 5×5×5. (b) A zoom-in of ESP-Net. "S:3×3, d=2" represents a 2D spatial convolution with kernel size 3 and dilation size 2. "T" indicates a 1D temporal convolution. "Max:3" denotes the 3D max-pooling operator with kernel size 3. "Avg", $\oplus$ and $\otimes$ denote the average pooling, the element-wise addition, and the broadcasting channel-only matrix multiplication, respectively. "FC" denotes a fully connected layer.

is estimated by the Event Scale Perceiving Network (ESP-Net):

$$M = \mathsf{ESP\text{-}Net}(X^l), M \in \mathbb{R}^{C \times C}. \tag{3}$$

As formulated in Eq. (2), M dynamically gates the spatial information flowed into each temporal convolution. By choosing different M, we can mimic the previous hand-crafted video architectures. For example, by only activating the matrix elements connecting the spatial and temporal kernels with the same size, the proposed formulation degenerates to the (2+1)D convolutions. Moreover, the multi-scale spatial-only or temporal-only convolutions are also the special cases of it.

Event Scale Perceiving Network (ESP-Net). It is well known that the scale information is embodied in the spatial-temporal context, which encodes the rich semantics *w.r.t* the shapes of objects and the dynamics of events. Thus, ESP-Net is implemented as a lightweight 3D network with a small channel number but a large receptive field, as shown in Fig. 3 (b). Specifically, a 1×1×1 3D convolution layer is first adopted to reduce feature channels of the input tensor X^l by 16 times. Then, the video context features are extracted with two 3D convolutions with kernel size $5 \times 5 \times 5$ and stride size $2 \times 2 \times 2$. Subsequently, the average pooling operation is utilized to only reserve the channel dimension of the tensor, and globally aggregate the event scale information of the input video. Finally, a linear transformation layer followed by a reshaping operation is utilized to produce M.

Implementation of EAB. We wrap the above procedure into an Event Adaptive Block (EAB). This block is defined as: $Z^l = Y^l + X^l$, where Y^l is given in Eq. (2) and "$+ X^l$"

denotes a residual connection [23]. The residual connection allows us to insert the proposed block into any pre-trained model such as ResNet, without breaking its initial behavior (*e.g.*, when the weights of the last Conv layer in EAB are initialized as zeros). An example of EAB with maximum receptive field size $5 \times 5 \times 5$ is illustrated in Fig. 3 (a). Bottleneck design is introduced for reducing the computation complexity, *i.e.*, we first reduce the feature channel number by four times through $1 \times 1 \times 1$ convolutions. The spatial convolutions are followed by batch normalization (BN) and ReLU non-linearity. We also introduce a max-pooling branch as a complement for convolution. To further reduce the parameter and the computational complexity, we replace the convolution of large kernel size with dilated convolution.

Further discussion with dynamic convolution. Dynamic Convolution [8] proposes to decouple the dynamic convolution as the attentions over several static convolutions. Nevertheless, our method is dedicated to video data while they are only for image data. In addition to that, there are several other significant differences between our method and them. *First*, our method is not merely *context-adaptive* but also *scale-adaptive*. More concretely, during the kernel generation procedure, dynamic convolution uses a global average pooling (GAP) operation to extract the global context information as the first step. In contrast, we reserve the additional spatial-temporal dimensions and utilize the 3D convolutions to extract the scale information of the objects and events. *Second*, the convolutions adopted in our method are with various kernel sizes to adapt to the events of various scales, while that in dynamic convolution are with the same kernel size. *Third*, the element of M in our method is specified for each channel, while the attention weight of dynamic convolution is shared across all channels of the convolution.

3.2 Sparse Object Interaction Transformer

The proposed EAB only captures the local information of the video, lacking the global modeling capability. Therefore, we propose a Sparse Object Interaction Transformer (SOI-Tr) to aggregate the local action cues into a global representation, as shown in Fig. 4. To make the modeling procedure more accurate for the specific input video, we only mine the interactions among the foreground objects in each frame, which are localized on the fly in the feature space.
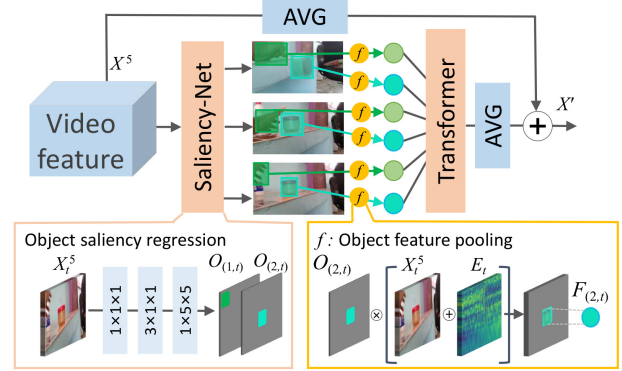
Given the output feature of the 5-th stage of the backbone CNN $X^5 \in \mathbb{R}^{C \times T \times W \times H}$, where C denotes the channel number, T denotes the temporal length, $W \times H$ represent the spatial scales, we model SOI-Tr as follows:

(1) Localizing the foreground objects. The location of each object is represented as a two-dimensional saliency map, whose spatial scale is equivalent to that of X^5, *i.e.*, $W \times H$. We use a small fully convolutional network (FCN) termed

Saliency-Net to regress the object saliency maps in parallel:

$$O = \textsf{Saliency-Net}(X^5), O \in \mathbb{R}^{N \times T \times W \times H}, \quad (4)$$

where N denotes the maximum number of the foreground objects in one frame. We empirically set $N = 4$ as most actions only involve less than four objects.

Fig. 4: The architecture of the proposed Sparse Object Interaction Transformer (**SOI-Tr**). $O_{(2,t)}$ represents the saliency map for the second object in the t-th frame. E_t denotes the positional embedding. We illustrate the feature map as the original RGB frame for better intuitive understanding.

(2) Pooling the object features. We first denote the saliency map for the n-th object in the t-th frame as $O_{(n,t)} \in \mathbb{R}^{W \times H}$. Then, the feature representation of the object is produced by spatially weighting the input video feature with the saliency map:

$$F_{(n,t)} = \textsf{SSUM}((E_t + X_t^5) \odot O_{(n,t)}), F_{(n,t)} \in \mathbb{R}^C, \quad (5)$$

where $\odot$ denotes the broadcasting element-wise multiplication, SSUM denotes the summation across spatial dimensions. E_t denotes a learnable spatially positional embedding with the same shape as $X_t^5 \in \mathbb{R}^{C \times W \times H}$.
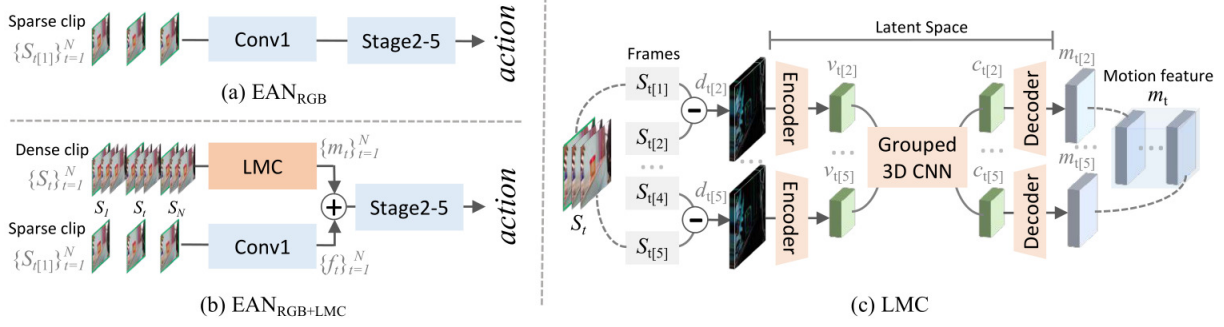
(3) Modeling the object interactions. With the object-level features, we model the global interactions among them using a Transformer:

$$F' = \textsf{Transformer}(F), F' \in \mathbb{R}^{N \times T \times C}. \quad (6)$$

The produced F' is with the same shape as F.

(4) Enhancing the global video representation. Finally, we perform average-pooling on the original video features and the interaction features, yielding the global video representation X'. It should be mentioned that the SOI-Tr module also adopts the bottleneck designing with a channel compressing factor of four.

Saliency-Net. This network is implemented as a lightweight four-layer CNN followed by a spatial Softmax layer, where the first layer reduces the the input channel number by a factor of eight. The second layer is a 3D convolution with

Fig. 5: (a) EAN$_{\text{RGB}}$ model only takes the sparsely sampled clip as the input. (b) EAN$_{\text{RGB+LMC}}$ model takes both the densely sampled clip and the sparse clip as the input. (c) Zooming-in of the Latent Motion Code (LMC) module, which transforms a local video segment S_t to a compact motion feature m_t.

kernel size $3 \times 1 \times 1$, which detects the moving objects with obvious motions. The third layer is a 2D convolution with a larger spatial kernel size 5×5, which localizes objects more accurately by considering the context information.

Transformer. The transformer architecture used in our framework is built by stacking two residual blocks, where each block includes a multi-head QKV self-attention module. Different from the vanilla Transformer, we mainly remove the classification token and replace the Layer Normalization with Batch Normalization, following the practices proposed in [44].

3.3 Latent Motion Code Module

The proposed EAB and SOI-Tr modules can already extract the action cues from the input video clip effectively. We insert several EABs and a SOI-Tr into the 2D ResNet backbone, building the EAN$_{\text{RGB}}$ model, as shown in Fig. 5 (a). EAN$_{\text{RGB}}$ already recognizes the actions from the sparsely sampled video clip effectively. Nevertheless, some subtle action cues are inevitably lost during the sampling procedure. To alleviate this issue, we sample the video clip more densely, and introduce a novel Latent Motion Code (LMC) module to efficiently mine the motion cues within the local segments of this dense clip, as shown in Fig. 5 (c). When equipping EAN$_{\text{RGB}}$ with the LMC module, we build an improved EAN$_{\text{RGB+LMC}}$ model, as shown in Fig. 5 (b). Although the EAN$_{\text{RGB+LMC}}$ takes more frames as input, it is also with high efficiency, due to the adopted latent motion modeling scheme and early feature fusion strategy.

Frame Sampling Strategy. Following the previous works [59] [67] [58], we uniformly divide the original long video into several groups and then select a segment from each group. Specifically, the input video is first divided into N groups with equal length. N is 8 or 16 for different computational budgets. During the training procedure, five adjacent frames are randomly chosen from each group as a 5-frame segment. The N segments form a dense clip $\{S_t\}_{t=1}^N$. The first frames of each segment form a sparse clip $\{S_{t[1]}\}_{t=1}^N$.

Latent Motion Code Module (LMC). This module aims to transform the short-term motion information within each local segment into a single compact motion feature, as shown in Fig. 5 (c). Given an input segment S_t, we model the motion information within it as follows:

(1) Calculating RGB difference maps. We obtain the low-level motion cue, *i.e.*, RGB difference map, by subtracting every two consecutive frames:

$$d_{t[i]} = S_{t[i]} - S_{t[i-1]}, d_{t[i]} \in \mathbb{R}^{3 \times 224 \times 224}, i \in [2, 5]. \quad (7)$$

(2) Encoding motion from RGB to latent space. Due to the high redundancy between the consecutive frames, the produced difference map is naturally sparse and contains many near-zero values. To simultaneously improve the compactness of the signal and also filter out the task-unrelated motion information, we use a learnable encoder to transform it into a high-dimensional latent space. Specifically, we divide $d_{t[i]}$ into $7 \times 7 = 49$ patches, where each patch is of shape $3 \times 32 \times 32$. Then, we compress these three-dimensional patches into 128-element latent vectors, and the vectors form a latent map of size 7×7:

$$v_{t[i]} = \mathsf{H_e}(d_{t[i]}), v_{t[i]} \in \mathbb{R}^{128 \times 7 \times 7}, \quad (8)$$

where the encoder $\mathsf{H_e}$ is implemented as a linear layer that is shared by all patches. The input and output dimensions of the layer are $3 \times 32 \times 32 = 3072$ and 128, respectively.

(3) Modeling high-order motion in latent space. The latent map is with low resolution and thus can be efficiently processed by 3D convolutions:

$$\{c_{t[i]}\}_{t=2}^5 = \mathsf{H_m}(\{v_{t[i]}\}_{t=2}^5), c_{t[i]} \in \mathbb{R}^{128 \times 7 \times 7}, \quad (9)$$

where $\mathsf{H_m}$ is implemented as two stacking 3D convolutions with kernel size 3 and group size 16. We call the produced $c_{t[i]}$ as latent motion code (LMC) because it captures the high-order motion information in the latent space.

(4) Decoding motion from latent to feature space. Through another linear transformation, LMCs can be decoded into

the feature space. Following TDN [58], we align the dimensions of the decoded features with the features from the Conv1 stage. Concretely, we decode the vector in each spatial position of LMC into a feature patch of size $16 \times 8 \times 8$, and these 7×7 patches form a motion feature map of shape $16 \times 56 \times 56$:

$$m_{t[i]} = \mathsf{H_d}(c_{t[i]}), m_{t[i]} \in \mathbb{R}^{16 \times 56 \times 56}, \qquad (10)$$

where the decoder $\mathsf{H_d}$ is implemented as a linear layer with the input dimension of 128 and the output dimension of $16 \times 8 \times 8$, respectively. Finally, the motion feature for segment S_t is constructed by stacking the motion feature maps along the channel dimension:

$$m_t = [m_{t[2]}; m_{t[3]}; m_{t[4]}; m_{t[5]}], m_t \in \mathbb{R}^{64 \times 56 \times 56}. \quad (11)$$

$\mathbf{EAN_{RGB+LMC}}$ **architecture**. As shown in Fig. 5 (b), for each segment S_t, we add the motion feature m_t produced by the LMC module to the Conv1 feature of the first frame $S_{t[1]}$:

$$f_t = m_t + \mathsf{Conv1}(S_{t[1]}). \qquad (12)$$

Then, the fused features of each segment are fed to the remained stages of EAN for predicting the action category score:

$$action = \mathsf{Stage2\text{-}5}(\{f_t\}_1^N), \qquad (13)$$

where $\mathsf{Conv1}$ and $\mathsf{Stage2\text{-}5}$ are indicated in Fig. 2.

4 Experiments

Datasets. We evaluate our method on several large-scale video datasets with different properties, requiring our models to understand different aspects of action recognition task.

Something-Something includes V1 [21] and V2 [38] versions, which are two large-scale crowd-sourcing video datasets for action recognition. There are about 110k (V1) and 220k (V2) videos covering 174 fine-grained action categories with diverse objects and scenes, focusing on humans performing pre-defined basic actions. In this dataset, the actions are performed with different objects so that models are required to understand the basic actions instead of recognizing the appearance of the objects or the background scenes. Moreover, the spatial and the temporal scales of the objects and the events vary hugely across different videos, as shown in Fig. 1, which is suitable for verifying the flexible spatial-temporal modeling ability of the proposed method.

Kinetics [6] is a challenging human action recognition dataset, which contains 400 and 600 human action classes. This dataset includes human-object interactions such as playing instruments, as well as human-human interactions such as shaking hands and hugging. Compared to the temporal

reasoning required by the actions in Something-Something, the actions in this dataset heavily rely on the appearance of the objects. We evaluate our models on the trimmed version to evaluate its capacity in modeling the appearances and the interaction among objects. The experiments are conducted on the validation set of Kinetics-400 [6] because there are many well-known baseline methods.

Diving48 [31] includes more than 18K video clips for 48 unambiguous diving classes. This proves to be a challenging task for modern action recognition systems as dives include three stages (takeoff, flight, entry) and thus require modeling of long-term temporal dynamics. This requires both multi-scale temporal modeling and the perceiving of long-range dependencies. Therefore, we conduct experiments on this dataset to verify the multi-scale spatial-temporal modeling ability of our method. We report the accuracy on the first version of the official validation split, which has been adopted by several previous methods.[1]

Implementation Detail We implement our model in Pytorch, and we adopt ResNet50 [23] pretrained on ImageNet [11] as the backbone. Following previous works [29] [30], we also insert temporal convolutions with kernel size 3 and the motion excitation (ME) module proposed in [30] before each 3×3 convolutions of bottleneck layers of the original ResNet50, aiming to enhance its basic temporal modeling ability. We also incorporate these changes into all baselines in the ablation study for a fair comparison. The parameters within the EABs and SOI-Tr are randomly initialized. For the spatial dimension of the sampled clips, the short-side of the frames are resized to 256 and then cropped to 224×224. We perform random cropping and flipping as data augmentation during training. It's worth mentioning that we do not perform horizontal flipping on the moving direction related action classes such as *"moving something from left to right"*. We train the network with a batch size of 64 and optimize it using SGD with an initial learning rate of 0.01 for 40 epochs, and decay it by a factor of 10 for every 10 epochs. The total training epochs are about 70. The dropout ratio is set to 0.5. The weight decay is set to $5e^{-4}$ and $1e^{-4}$ for Something/Diving48 and Kinetics-400, respectively.[2]

4.1 Comparison with State-of-the-Arts

Something V1 and V2. We first compare our method with the other state-of-the-art approaches on Something V1 and Something V2 datasets, as shown in Tab. 1. The previous approaches are divided into four groups: 3D CNNs, object interaction modeling enhanced 3D CNNs, 2D CNNs, and 2D CNNs enhanced with short-term motion representation. Our method outperforms all methods built with 3D convolutions and meanwhile achieves higher efficiency. For ex-

---

[1] http://www.svcl.ucsd.edu/projects/resound/Diving48_{train/test}.json

[2] We adopt the same hyper-parameter settings as the official codebase of TDN for a fair comparison.

Table 1: Comparison to state-of-the-arts on Something-Something V1&V2 datasets. Following TDN [58], we adopt the *1-clip and center-crop* inference scheme where only a center crop of 224×224 from a single clip is used for evaluation. $_{8F}$ and $_{16F}$ indicate the sampling segment number of the input video is 8 and 16, respectively. The result of $EAN_{En(RGB)}$ is produced by averaging the predicted action scores from the $EAN_{8F(RGB)}$ and $EAN_{16F(RGB)}$ models, which follows TSM [32]. − indicates the paper didn't provide the results.

Method	Backbone	Pre-train	Frames	GFLOPs	Something V1		Something V2	
					Top1 (%)	Top5 (%)	Top1 (%)	Top5 (%)
3D CNNs:								
I3D [6]	3D-ResNet50	Kinetics	32×2	306	41.6	72.2	-	-
Non-local I3D [61]	3D-ResNet50	Kinetics	32×2	336	44.4	76.0	-	-
ECO(En) [71]	BNInc+3D-ResNet18	Kinetics	92	267	46.4	-	-	-
S3D-G [64]	InceptionV1	ImageNet	64	71	48.2	78.7	-	-
3D CNNs + Object interaction:								
GCN + Non-local [62]	3D-ResNet50	Kinetics	32×2	606	46.1	76.8	-	-
I3D + STIN + OIE [39]	I3D	Kinetics	32	154	-	-	60.2	84.4
2D CNNs:								
TSN [59]	BN-Inception	ImageNet	8	16	19.5	-	33.4	-
MultiScale TRN [70]	BN-Inception	ImageNet	8	16	34.4	63.2	48.8	77.6
TSM_{8F} [32]	ResNet-50	Kinetics	8	33	45.6	74.2	58.8	85.4
TSM_{16F} [32]	ResNet-50	Kinetics	16	65	47.2	77.1	63.4	88.5
$TANet_{8F}$ [34]	ResNet-50	ImageNet	8	33	46.5	75.8	60.5	86.2
$TANet_{16F}$ [34]	ResNet-50	ImageNet	16	66	47.6	77.7	62.5	87.6
$TANet_{En}$ [34]	ResNet-50	ImageNet	8+16	99	50.6	79.3	-	-
$TEINet_{8F}$ [33]	ResNet-50	ImageNet	8	33	47.4	-	61.3	-
$TEINet_{16F}$ [33]	ResNet-50	ImageNet	16	66	49.9	-	62.1	-
$TEINet_{En}$ [33]	ResNet-50	ImageNet	8+16	99	52.5	-	65.5	89.8
STM [26]	ResNet-50	ImageNet	8×30	990	49.2	79.3	62.3	88.8
STM [26]	ResNet-50	ImageNet	16×30	2010	50.7	80.4	64.2	89.8
GST_{8F} [36]	ResNet-50	ImageNet	8	29	47.0	76.1	-	-
GST_{16F} [36]	ResNet-50	ImageNet	16	59	48.6	77.9	62.6	87.9
TEA_{8F} [30]	ResNet-50	ImageNet	8	35	48.9	78.1	-	-
TEA_{16F} [30]	ResNet-50	ImageNet	16	70	51.9	80.3	-	-
TEA [30]	ResNet-50	ImageNet	16×30	2100	52.3	81.9	65.1	89.9
$EAN_{8F(RGB)}$(Ours)	ResNet-50	ImageNet	8	36	51.9	79.5	63.5	88.2
$EAN_{16F(RGB)}$(Ours)	ResNet-50	ImageNet	16	72	53.4	81.4	64.6	89.1
$EAN_{En(RGB)}$(Ours)	ResNet-50	ImageNet	8+16	108	**55.8**	**83.1**	**66.6**	**89.9**
2D CNNs + Short-term motion:								
$TRN_{RGB+Flow}$ [70]	BN-Inception	ImageNet	8×7	-	42.0	-	55.5	83.1
$TSM_{RGB+Flow}$ [32]	ResNet-50	ImageNet	16×7	-	52.6	81.9	66.0	90.5
$PAN_{8F(RGB+PAN)}$ [67]	ResNet-50	ImageNet	8×5	68	50.5	79.2	63.8	88.6
PAN [67]	ResNet-101	ImageNet	$(8 \times 5) \times 2$	503	55.3	82.8	66.5	90.6
$TDN_{8F(RGB+SDM)}$ [58]	ResNet-50	ImageNet	8×5	36	52.3	80.6	64.0	88.8
$TDN_{16F(RGB+SDM)}$ [58]	ResNet-50	ImageNet	16×5	72	53.9	82.1	65.3	89.5
$TDN_{En(RGB+SDM)}$ [58]	ResNet-50	ImageNet	$(8+16) \times 5$	108	55.1	82.9	67.0	90.3
$EAN_{8F(RGB+LMC)}$(Ours)	ResNet-50	ImageNet	8×5	37	53.4	81.1	65.2	89.4
$EAN_{16F(RGB+LMC)}$(Ours)	ResNet-50	ImageNet	16×5	74	54.7	82.3	66.6	90.3
$EAN_{En(RGB+LMC)}$(Ours)	ResNet-50	ImageNet	$(8+16) \times 5$	111	**57.2**	**83.9**	**68.8**	**91.4**

ample, compared with Non-local I3D [61], our $EAN_{8F(RGB+LMC)}$ model achieves 8.8% higher Top1 accuracy (44.4% *vs.* 53.2% on Something V1) with only $\sim 11\%$ computational cost.

We also compare our method with the two methods [39] [62] that first detect the objects of the input frames in the RGB space and then model the object interactions. Although we do not use the pretrained object detector or extra object bounding box annotations to get the proposal regions, our method still significantly outperforms them. Specifically, our improvements over GCN + Non-local [62] and I3D + STIN + OIE [39] are **11.1%** (on Something V1) and **8.6%** (on Something V2), respectively, in terms of the Top1 recogni-

tion accuracy. This proves the superiority of the end-to-end object detection scheme and the Transformer architecture adopted in our method.

As for the 2D CNN-based methods, we compare our EAN_{RGB} architecture with them for a fair comparison, where only one frame is sampled from each segment. Our models achieve the best performance under all settings of different input frame numbers. The performances of TSN and TRN are relatively inferior to other methods because both the two methods only model the temporal information upon the highest-level feature maps from the backbone CNN. TEA is superior to all other 2D CNNs because it explores multi-

Table 2: Comparison to state-of-the-arts on Kinetics-400 dataset. Following TDN [58], we adopt the *10-clip and 3-crop* inference scheme where three crops of 256×256 frames and 10 clips are used for testing. Therefore, the computational cost of the same model here is 30× heavier than that on Something datasets. − indicates the paper didn't provide the results.

Method	Backbone	Pre-train	Frames	GFLOPs	Top1 (%)	Top5 (%)
ARTNet [57]	ResNet-18	ImageNet	16×250	23.5×250	70.7	89.3
I3D [6]	Inception V1	ImageNet	64×N/A	108×N/A	72.1	90.3
I3D [6]	Inception V1	None	64×N/A	108×N/A	67.5	87.2
I3D+NL [61]	3D-ResNet-101	ImageNet	32×60	359×60	77.7	93.3
ECO(En) [71]	BNInc&3D-ResNet-18	None	92	267	70.0	-
SlowOnly [14]	3D-ResNet-50	None	8×30	41.9×30	74.8	91.6
SlowFast [14]	3D-ResNet-50	None	(4+32)×30	36.1×30	75.6	92.1
SlowFast+NL [14]	3D-ResNet-101	None	(16+64)×30	234×30	**79.8**	**93.9**
TSN [59]	BN-Inception	ImageNet	25×10	53×10	69.1	88.7
TSN [59]	Inception v3	ImageNet	25×10	80×10	72.5	90.2
R(2+1)D [54]	ResNet-34	None	32×10	152×10	72.0	90.0
TSM [32]	ResNet-50	ImageNet	8×30	33×30	74.1	-
TSM [32]	ResNet-50	ImageNet	16×30	65×30	74.7	-
STM [26]	ResNet-50	ImageNet	16×30	67×30	73.7	91.6
TEINet [33]	ResNet-50	ImageNet	8×30	33×30	74.9	91.8
TEINet [33]	ResNet-50	ImageNet	16×30	66×30	76.2	92.5
TANet [33]	ResNet-50	ImageNet	8×30	43×30	76.1	92.3
TANet [33]	ResNet-50	ImageNet	16×12	86×12	76.9	92.9
TEA [30]	ResNet-50	ImageNet	16×30	70×30	76.1	92.5
PAN [67]	ResNet-50	ImageNet	(8×5)×2	270	75.3	92.4
TDN$_{8F(RGB+SDM)}$ [58]	ResNet-50	ImageNet	(8×5)×30	36×30	76.6	92.8
TDN$_{16F(RGB+SDM)}$ [58]	ResNet-50	ImageNet	(16×5)×30	72×30	77.5	93.2
TDN$_{En(RGB+SDM)}$ [58]	ResNet-50	ImageNet	(8+16)×5×30	108×30	78.4	93.6
TDN$_{En(RGB+SDM)}$ [58]	ResNet-101	ImageNet	(8+16)×5×30	198×30	**79.4**	**94.4**
EAN$_{8F(RGB+LMC)}$(Ours)	ResNet-50	ImageNet	(8×5)×30	37×30	77.1	93.3
EAN$_{16F(RGB+LMC)}$(Ours)	ResNet-50	ImageNet	(16×5)×30	74×30	78.3	93.7
EAN$_{En(RGB+LMC)}$(Ours)	ResNet-50	ImageNet	(8+16)×5×30	111×30	**79.0**	**94.1**

scale spatial-temporal information. Compared with TEA, our method outperforms it consistently with the different input frame numbers. When using 8 and 16 input frames, the improvements are 3.0% and 1.5% on Something V1 dataset. The reason is that the multi-scale architecture of TEA is based on the hand-crafted Res2Net, which is static and not adaptive to the video. In contrast, the spatial-temporal modeling architecture of our method is dynamic and adaptive.

We further compare the improved EAN$_{RGB+LMC}$ architecture with the other recent 2D CNNs that also take advantage of the short-term motion information, where 5 adjacent frames are sampled from each segment. Compared with the optical flow-based methods, *i.e.*, TRN$_{RGB+Flow}$ and TSM$_{RGB+Flow}$, our smallest model EAN$_{8F(RGB+LMC)}$ already outperforms them by 11.2% and 0.6%, respectively. It's worth noting that the computational complexity of our LMC motion feature produced from the input video of 40 frames is only 1.1 GFLOPs, while the computational complexity of FlowNet2.0 [24] is 2006 GFLOPs for the same video. In other words, the proposed LMC module is 1823× more efficient than optical flow modality, while achieving better performance for the action recognition task.

By averaging the predictions from EAN$_{8F(RGB+LMC)}$ and EAN$_{16F(RGB+LMC)}$, the resulted model EAN$_{En(RGB+LMC)}$

boosts the action recognition performance to a new state-of-the-art level, *i.e.*, 57.2% (△ **+2.1%**) on Something V1 and 68.8% (△ **+1.8%**) on Something V2, when using the recent method TDN as the anchor. Compared with the PAN model, the improvement of our method is 1.9% on Something V1, even though that PAN adopts a much heavier backbone network, *i.e.*, 2D-ResNet101.

Furthermore, we summarize the key differences among our adopted ResNet baseline, our variant models and other recent relevant methods, as shown in Tab. 3. Particularly, TDN also uses a short-term temporal difference module (SDM) to exploit short-term motion information in the low-level feature space, and fuse the motion features into the backbone in the early stage. Nevertheless, our method outperforms TDN by 1.1% on Something V1. The consistent improvements of our method over the other methods strongly justify the superiority of the proposed event scale adaptive spatial-temporal modeling paradigm by EAB, sparse object interaction modeling scheme by SOI-Tr, and high-order motion representation in latent space by LMC.

Kinetics-400. To verify that our method also effectively captures rich object appearance cues and the interactions among them, we compare our method with other state-of-the-art results on the Kinetics-400, as shown in Tab. 2. When

Table 3: Comparison of different models in terms of input clip setting and their key modules, *i.e.*, local spatial-temporal modeling module, global aggregation module and short-term motion modeling module. PA, TSM and ME indicate the appearance of persistence module [67], the temporal shift module [32] and the motion excitation module [30], respectively. L/SDM represents the long/short-term temporal difference modules in TDN [58]. AVG indicates the spatial-temporal average pooling operation. The model performances on Something V1 are also reported.

Model	Input clip	Key components			Top1 (%)
	segment×frame	Local	Global	Motion	
TSM_{8F}	8×1	TSM	AVG	-	45.6
ResNet baseline	8×1	ME	AVG	-	48.6
$EAN_{8F(RGB)}$	8×1	EAB	SOI-Tr	-	51.9
$PAN_{8F(RGB+PAN)}$	8×5	TSM	AVG	PA	50.5
$TDN_{8F(RGB+SDM)}$	8×5	LDM	AVG	SDM	52.3
$EAN_{8F(RGB+LMC)}$	8×5	EAB	SOI-Tr	LMC	**53.4**

compared with the methods based on 2D CNNs, our method outperforms all of them when using the same backbone network, and demonstrates a better trade-off between the action recognition accuracy and the computational complexity. For example, when equipped with the same ResNet-50 backbone, our method outperforms the recent method TDN by 0.6%. When adopting the ResNet-101 backbone, TDN shows the strongest result among all 2D CNNs. Nevertheless, this also increases the computation cost of TDN, which is even close to the 3D CNN method, *i.e.*, SlowFast + NL network. Our EAN models achieve the best complexity performance trade-off among all state-of-the-art methods.

Diving48. To prove that our method can model subtle fine-grained motion cues, we test our method on Diving48. This dataset requires modeling the subtle body motions in long-short terms and includes much fewer videos compared with Something-Something and Kinetics. We input 16 frames

Table 4: Comparison to state-of-the-arts on Diving48. We adopt the *single clip or twice clips* inference schemes where a center crop of 224×224 from a single clip or twice clips is used. − indicates the paper didn't provide the results.

Method	Pre-train	Frames	Top1 (%)
TSN (from [31])	ImageNet	8	16.7
TRN (from [31])	ImageNet	8	22.8
C3D (from [31])	ImageNet	64	27.6
R(2+1)D (from [3])	Kinetics	-	28.9
P3D (from [36])	ImageNet	16	32.4
C3D (from [36])	ImageNet	16	34.5
Kanojia *et al.* [27]	ImageNet	64	35.6
TEA-ResNet50 [31]	ImageNet	16	36.0
CorrNet-101 [56]	-	32×10	38.6
GST [36]	ImageNet	16	38.8
Ours	ImageNet	16	40.4
Ours	ImageNet	16×2	**41.7**

to the network and sample two clips from the video during inference. The results are shown in Tab. 4. Our method outperforms the recent state-of-the-art GST [36] when using single clip (△ **+1.6%**) or twice clips (△ **+2.9%**) as the input videos.

4.2 Ablation Studies for EAN

We conduct extensive ablation studies on Something V1 [21] dataset to demonstrate the superiority of the proposed framework by answering the following questions. The variant models in this section are derived from the $EAN_{8F(RGB)}$ model. The input clip is always with 8 frames.
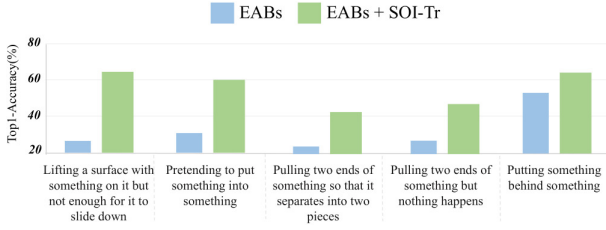
Q1: Are the proposed EAB and SOI-Tr effective and necessary? As mentioned in Sec. 3, in our framework, the EAB extracts more accurate local spatial-temporal representation and the SOI-Tr derives global object interaction representation from the video. To confirm that both two representations are effective and necessary for a high-performance action recognition framework, we conduct ablation experiments. Specifically, we equip ResNet baseline with the two proposed modules separately and analyze their impact on the performance.

Table 5: Comparison of the performance of using different spatial-temporal modeling modules.
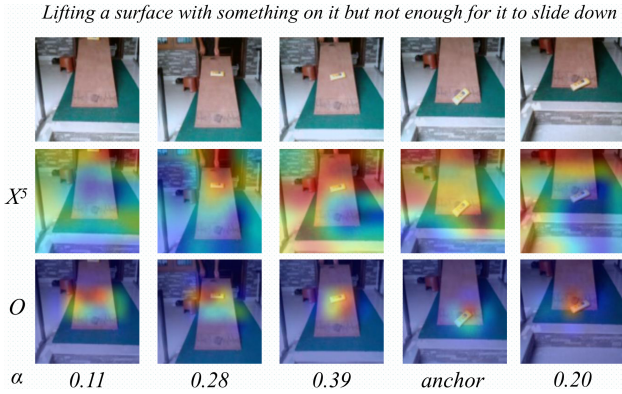
Method	Param	FLOPs	Something V1	
			Top1 (%)	Top5 (%)
ResNet baseline	24.0M	33.1G	48.6	77.5
ResNet+EABs	29.5M	35.3G	50.8	78.4
ResNet+SOI-Tr	30.3M	33.8G	49.3	77.9
ResNet+EABs+SOI-Tr	36.0M	36.1G	**51.9**	**79.5**

As shown in Tab. 5, both the two modules demonstrate strong video modeling capability. When the ResNet baseline is enhanced with the EABs, the Top1 accuracy is significantly improved by 2.2%. The reason is that the features extracted by the ResNet baseline are not accurate enough, and the proposed EABs can refine the features with the dynamic spatial-temporal kernel. For a more intuitive understanding, we will visualize the refined feature maps by our method in section 4.3. Then, we observe that the ResNet + SOI-Tr baseline also outperforms the original ResNet baseline by 0.7% in terms of Top1 accuracy, while only introducing an extra 0.7 GFLOPs computation cost. Finally, simultaneously using EAB and SOI-Tr boosts the performance to 51.9%, which proves the complementary of the two proposed modules.

We also plot the top 5 classes that are significantly improved after introducing the SOI-Tr. As shown in Fig. 6, we find that the most improved instances can be roughly divided into two groups: (a) The instances that require tracking the state of a certain object over the whole clip, such as the

Fig. 6: The top 5 action classes that are significantly improved after introducing the SOI-Tr module.

Lifting a surface with something on it but not enough for it to slide down

Fig. 7: Visualization of one video clip from the most improved category by the SOI-Tr. The input clip is first processed by EABs to obtain the spatial-temporal feature map X^5. Then, SOI-Tr calculates the saliency map O of the most concentrated object and the interactions of this object across the temporal axis. We take the 4th frame as the anchor and show the attention vector α.

videos of *"Lifting a surface ... "* and *"Pulling two ends ... "*. (b) The instances that contain multiple objects and the interactions between them, such as the videos of *"Pretending to put ... "*. This is aligned with the motivation of introducing SOI-Tr, *i.e.*, accurately modeling the long-range object interactions benefits the recognition of some complex actions.

To systematically understand how the EABs and SOI-Tr improve the recognition performance, we randomly select one video from the category *"Lifting a surface with something on it but not enough for it to slide down"* and visualize it. In Fig. 7, we can clearly see that the original feature X^5 before global modeling concentrates on the background or the board, omitting the main object, *i.e.*, the small sliding box. This makes sense because both the spatial area and the motion magnitude of the board are more obvious than the small box. After introducing the SOI-Tr, the object detector first finds the main object. Then, the Transformer model builds the long-range dependencies across the whole clip. We also notice that the board in the first frame is also detected. But, this background object will be neglected in the self-attention model because its weight is only 0.11.

Table 6: Study on the location of EAB and SOI-Tr.

EAB	SOI-Tr	Param	FLOPs	Something V1	
				Top1 (%)	Top5 (%)
Stage 1∼2	Stage 3∼5	35.6M	35.8G	49.4	78.2
Stage 1∼3	Stage 4∼5	34.8M	35.9G	50.4	79.2
Stage 1∼4	Stage 5	36.0M	36.1G	**51.9**	**79.5**
Stage 1∼5	-	65.8M	36.2G	50.8	79.1

Q2: Where to insert the proposed modules? We perform an ablation study on which stage to use local operator (EAB) and global operator (SOI-Tr). The results are shown in Tab. 6. From these results, we see that adding more EABs into the main network only slightly increases the computational cost due to the high efficiency of the bottleneck designing and group convolution. When some EABs are replaced with the SOI-Tr, the performance decreases consistently. This implies that the local spatial-temporal information is crucial for action recognition, which cannot be substituted by the high-level object interaction information. We also try to build the network only with EABs, the result is also inferior to the original hybrid model (convolution+self-attention). The setting of using EAB after stage 1∼4 and SOI-Tr after stage 5 obtains the best recognition accuracy and is with reasonable complexity.

Q3: Is the prior assumption of SOI-Tr reasonable? To prove the end-to-end foreground object detector and the sparsity assumption for object interactions are both important for SOI-Tr, we train other variant models where we replace our detected object regions with the same number of the fixed regions or the regions detected by a pre-trained Faster RCNN [42] model. When the number of the boxes output from Faster RCNN is too small, we pad it with the central region of the frames. The performances of the models are compared in Tab. 7.

Table 7: Study on various object region sampling strategies.

Regions	FLOPs	Something V1	
		Top1 (%)	Top5 (%)
None	35.3G	50.8	78.4
Fixed	35.8G	50.9	78.4
Faster RCNN	71.3G	51.1	78.7
All	36.4G	51.3	79.1
Our Det-Net	36.1G	**51.9**	**79.5**

First, we notice that building the interaction model upon the fixed regions already slightly improves the performance, proving that the interaction modeling is beneficial to the action recognition. Then we use the Faster RCNN detector to predict more accurate foreground regions. Surprisingly, the performance improvement is negligible. This may be ascribed to the fact that most frames only contain one or two objects, which cover fewer regions compared with the "fixed region" scheme. In contrast, the Saliency-Net embedded in our method always detects enough salient regions in an end-
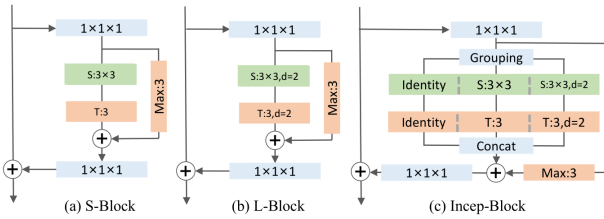
to-end manner and obtains the best performance, *i.e.*, 51.9%. Also, it is computationally efficient due to the shared feature extractor with the other parts of the framework. We emphasize that our embedded Saliency-Net outperforms Faster RCNN by 0.8% while running 118× faster.

We further try to leverage all positions to build a dense interaction model, as shown in the penultimate row of Tab. 7. However, the performance is obviously inferior to our method. This strongly supports our assumption that most regions are only background noises for the final prediction and leveraging all of them will deteriorate the final performance.

4.3 Further Studies for EAB

In this section, we make further studies on the aspects that impact the effectiveness of EAB.

Large receptive field and multi-scale modeling are important. To verify this, we introduce the following baselines:

Fig. 8: Illustrations of the baseline spatial-temporal blocks. The representation signs are the same meaning as Fig. 3.

(1) S-Block. It is implemented with a (2+1)D convolution with kernel size $3 \times 3 \times 3$ and group size 3, as shown in Fig. 8 (a), which only captures single-scale features with a small receptive field.

(2) L-Block. It is implemented with a (2+1)D convolution with kernel size $3 \times 3 \times 3$, group size 3, and dilation size $2 \times 2 \times 2$, as shown in Fig. 8 (b), which captures single-scale features with a larger receptive field.
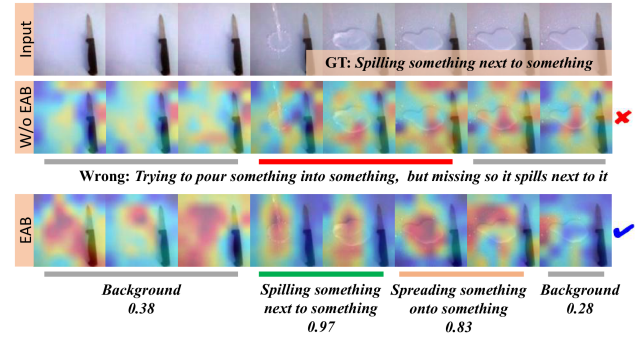
(3) Incep-Block. It is implemented with a group of (2+1)D convolutions in an Inception-style, as shown in Fig. 8 (c), which captures multi-scale features with a larger receptive field. The only difference between this baseline and EAB is that the M is replaced with an identity mapping operation.

We compare our method with the proposed baseline methods in Tab. 8. First, it can be seen that EAB outperforms the S-Block baseline by a large margin (51.9% *vs.* 49.6%). The improvement is originated from two aspects: (1) The large spatial-temporal kernel within EAB enables the larger receptive field and aggregates more local information. (2) The explicit multi-scale modeling introduces richer feature representation. It is necessary to validate the independent contribution from the two aspects. We first compare the S-Block

Table 8: Comparison of different spatial-temporal blocks. RFS denotes the receptive field size.

Models	RFS	Multi scale?	Param	FLOPs	Something V1	
					Top1 (%)	Top5 (%)
Only SOI-Tr	-	-	30.3M	33.8G	49.3	77.9
+S-Block	$3 \times 3 \times 3$	-	30.9M	36.1G	49.6	78.1
+L-Block	$5 \times 5 \times 5$	-	30.9M	36.1G	50.3	78.4
+Incep-Block	$5 \times 5 \times 5$	✓	30.9M	36.0G	50.8	78.8
+EAB	$5 \times 5 \times 5$	✓	36.0M	36.1G	**51.9**	**79.5**

baseline with the L-Block baseline. L-Block has a larger spatial-temporal receptive field size but the same number of parameters. We can see that the recognition accuracy is improved by 0.7%. Then, we build the Incep-Block baseline by enhancing the L-Block baseline with multi-scale modeling capability. This improvement further improves the recognition accuracy. From the comparisons above, we verify that both the two aspects facilitate the action task, and multi-scale architecture fully exploits the large receptive field.

Fig. 9: Comparison of the features from Stage4 of EAN and the evolution of the prediction scores. The proposed EAB can discover more semantically-aligned regions for actions, and also suppress the noisy information from backgrounds to yield correct prediction. *Zoom in for better visualization.*

We randomly select one video from Something V1 dataset and visualize the $8 \times 14 \times 14$ feature map output from Stage4 (this is before the inserting of the SOI-Tr), as shown in Fig. 9. It clearly demonstrates that our method can discover more semantically consistent regions for actions, and in the meantime reduce noisy backgrounds for correct prediction. Moreover, the feature activation heatmaps of our method are better spatially aligned with the target object (see the water). We also show the state evolution process in Fig. 9. Interestingly, our method detects the start and end points of actions although only trained with classification labels.

Dynamic architecture matters. From Tab. 8, we notice that the performance gap between the Incep-Block baseline and EAB is still rather large, *i.e.*, 1.1% Top1 accuracy. We conjecture this is due to the dynamic architecture of EAB.

As mentioned in section 3.1, the inference pathway for EAB is determined by the kernel fusion matrix M. For more detailed analysis, we propose two kernel fusion strategies: (1) Channel Shuffle. We replace M with a conventional fusion method, *i.e.*, Channel Shuffle operation [68], which enables the communication of the features of different groups. (2) Static Matrix. The M is a learnable matrix during training. But, it's a fixed matrix during inference.
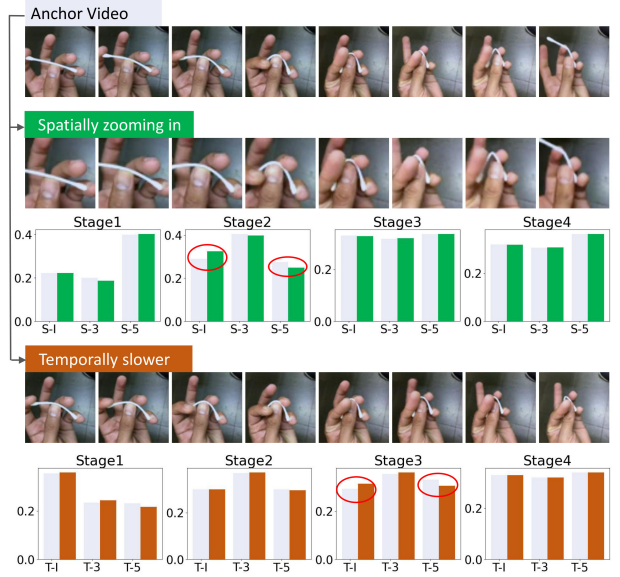
Table 9: Comparison of different feature fusion methods.

Methods	Param	FLOPs	Something V1	
			Top1 (%)	Top5 (%)
Identity (Incep-Block)	30.9M	36.0G	50.8	78.8
Channel Shuffle	30.9M	35.9G	51.1	78.9
Static Matrix	30.9M	36.0G	51.3	79.2
Dynamic Matrix	36.0M	36.1G	**51.9**	**79.5**

Both the above two baselines belong to the static architecture but they are similar to the EAB in terms of the network details, which are perfect for studying the impact of dynamic modeling. We compare their performances in Tab. 9. We observe that the performance improves consistently with a more complex kernel fusion strategy, *i.e.*, Identity →Channel Shuffle →Static Matrix →Dynamic Matrix. The Dynamic fusion matrix adopted by EAB shows the best performance (51.9%) with negligible extra cost.

Kernel visualization. To verify that the dynamic kernel fusion matrix M of EAB is indeed adaptive to the scales of the main objects and the key events within different videos, we conduct a group of experiments by augmenting one anchor video and observing the change of the weights of the fixed-scale kernels. The augmented videos and the kernel weight changing procedure are illustrated in Fig. 10. We first see, both the weight distributions along the temporal axis or the spatial axis are not sparse, *i.e.*, all kernels are activated. This supports our assumption that the optimal spatial-temporal kernel for the video is with an unknown complex shape and cannot be accurately replaced by one kernel of fixed-scale. Also, the distributions do not follow some simple distributions such as Uniform or Gaussian, indicating that the kernel weights cannot be trivially hand-crafted and are required to be learned from data. When we spatially zoom in the anchor video by 1.6×, the main objects in the video, *i.e.*, the hand and the stick, are easier to be discovered, we see that EAB is more inclined to exploit the spatial convolutions of small kernels such as that of size 1×1 instead of that of size 5×5 . Similarly, when we sample the frames with 2× higher frame-rate, the object motions become slower and the small temporal convolutions such as that with kernel size 1 are fully used.

Studies on EAB details. In this part, we conduct experiments to verify whether all the designs of EAB contribute

Fig. 10: Visualizing the dynamic kernel fusion matrix M of the proposed EAB via the kernel weights. For each spatial or temporal kernel, its weight is computed by summing the matrix values connected to it. In the first row, we give an anchor video. Below it, we show the impact to the kernel weights by changing the spatial or temporal scales of the anchor video. The kernel weights of anchor, spatially-, and temporally-augmented videos are indicated by gray, green, and brown bars, respectively. "S-3"&"T-3" denotes the fixed-size spatial&temporal kernel of size 3×3&3.

Table 10: Ablation on detailed designs of EAB.

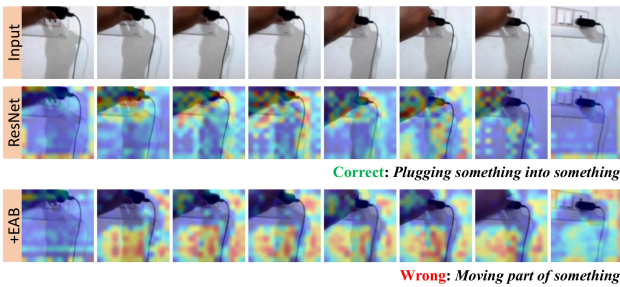| Design	Param	FLOPs	Something V1	
			Top1 (%)	Top5 (%)
Without Max Pool	36.0M	36.1G	50.6	78.4
Avg Pool	36.0M	36.1G	51.4	79.8
Without inter ReLU	36.0M	36.1G	50.9	79.0
Without dilation	37.2M	37.5G	**51.9**	**79.8**
(1+1+1)D	35.7M	35.7G	51.2	79.2
Ours	36.0M	36.1G	**51.9**	79.5

to the final performance. As shown in Tab. 10, the max pooling operation significantly improves the performance (1.3% *w.r.t* Top1 accuracy), and meanwhile our method is not sensitive to specific implementation of this operation. Both average pooling and max pooling operators achieve excellent performance. Max pooling demonstrates a slight advantage over average pooling because the regions of the key objects and frames related to action only cover a small proportion of the input video data. Also, we find that the extra non-linearity introduced by the intermediate ReLU operations between spatial- and temporal-filters also benefit the performance, which is consistent with the conclusion from previous work [54]. Besides, we demonstrate that the dilated convolution achieves comparable performance with the or-

dinary convolution while it is much more efficient. Finally, we also try to decompose the 2D spatial convolution into two stacking 1D convolutions. But this brings a slight performance drop. To summarize, the extensive experiments in this section prove the necessity of detailed designs in EAB.
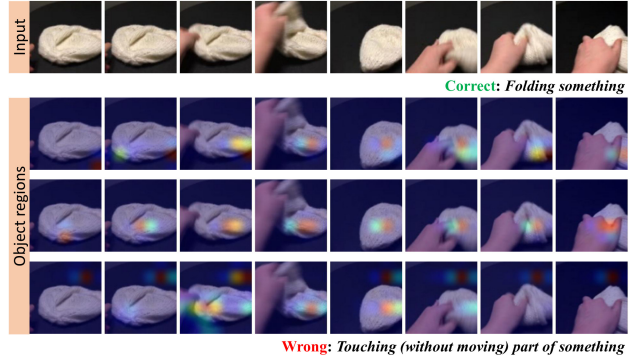
5 Erroneous Cases and Limitations

Although the quantitative results on standard benchmarks and the extensive analysis above have verified the effectiveness of the proposed framework, it inevitably has some limitations, which lead to erroneous recognition results.

One limitation is caused by the simple architecture of ESP-Net within EAB. ESP-Net is responsible for perceiving the event scales within the input video, composed of two convolution layers followed by a global average pooling operation. Although this simple "average" operation is lightweight in terms of the computational cost, it also makes the statistical results of the video biased to the large objects. As shown in Fig. 11, the feature activations are dominated by the large-area human hand shadow, neglecting the real objects (the human hand and the charger) involved in the action *plugging something into something*.

Fig. 11: Recognition error caused by EAB. The shadow of the human hand instead of the real hand and the charger is attended, causing the recognition result changing from *plugging something into something* to *moving part of something*.

Another limitation is originated from the proposed SOI-Tr. The adaptiveness of SOI-Tr lies in detecting different foreground objects for different input videos. Nevertheless, the adaptiveness may be limited by the representation of the objects, *i.e.*, points in the feature map, which correspond to fixed-size regions within the input video. Therefore, the granularity and the scale of the detected foreground objects are not flexible enough. As shown in Fig. 12, only small parts of the towel can be detected. Therefore, the global state *folding* of the towel can not be perceived. Instead, the local states of the wrongly attended objects, *i.e.*, the human hand and the partial towel, contribute to the wrong prediction *touching part of something*.

Fig. 12: Recognition error caused by SOI-Tr. We visualize the foreground object distribution maps, where only parts of the towel are detected in this case. Therefore, the global state *folding* of the towel can not be perceived, resulting the biased action recognition result *touching*.

6 Conclusion and Future Works

To model the spatial-temporal scale variances and the long-range object interactions in videos, we propose to dynamically generate the video-adaptive kernels from the input video and model the interactions among the objects with a Transformer. Moreover, we design a novel short-term motion representation to further enhance the performance of our method. We perform extensive evaluations to study the effectiveness of the proposed approach on video action recognition task, and the results demonstrate that our models achieve impressive performances on Something-Something V1/V2, Kinetics-400, and Diving48 datasets. In the future, we will explore how to better approximate the video-adaptive kernel. As for the network architecture, we will investigate more powerful backbone networks. We also plan to extend the proposed framework to more downstream video tasks such as the spatial-temporal action localization task.

References

1. Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., Schmid, C.: Vivit: A video vision transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6836–6846 (2021)
2. Bensch, R., Scherf, N., Huisken, J., Brox, T., Ronneberger, O.: Spatiotemporal deformable prototypes for motion anomaly detection. International Journal of Computer Vision **122**(3), 502–523 (2017)
3. Bertasius, G., Feichtenhofer, C., Tran, D., Shi, J., Torresani, L.: Learning discriminative motion features through detection. arXiv preprint arXiv:1812.04172 (2018)

4. Bertasius, G., Wang, H., Torresani, L.: Is space-time attention all you need for video understanding? arXiv preprint arXiv:2102.05095 (2021)

5. Bulat, A., Perez Rua, J.M., Sudhakaran, S., Martinez, B., Tzimiropoulos, G.: Space-time mixing attention for video transformer. Advances in Neural Information Processing Systems **34** (2021)

6. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6299–6308 (2017)

7. Chen, X., Pang, A., Yang, W., Ma, Y., Xu, L., Yu, J.: Sportscap: Monocular 3d human motion capture and fine-grained understanding in challenging sports videos. International Journal of Computer Vision **129**(10), 2846–2864 (2021)

8. Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., Liu, Z.: Dynamic convolution: Attention over convolution kernels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11,030–11,039 (2020)

9. Cherian, A., Gould, S.: Second-order temporal pooling for action recognition. International Journal of Computer Vision **127**(4), 340–362 (2019)

10. Cong, Y., Liao, W., Ackermann, H., Rosenhahn, B., Yang, M.Y.: Spatial-temporal transformer for dynamic scene graph generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 16,372–16,382 (2021)

11. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, pp. 248–255. Ieee (2009)

12. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)

13. Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., Feichtenhofer, C.: Multiscale vision transformers. arXiv preprint arXiv:2104.11227 (2021)

14. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: Proceedings of the IEEE international conference on computer vision, pp. 6202–6211 (2019)

15. Feichtenhofer, C., Pinz, A., Wildes, R.P., Zisserman, A.: Deep insights into convolutional networks for video recognition. International Journal of Computer Vision **128**(2), 420–437 (2020)

16. Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional two-stream network fusion for video action recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1933–1941 (2016)

17. Ferryman, J.M., Maybank, S.J., Worrall, A.D.: Visual surveillance for moving vehicles. International Journal of Computer Vision **37**(2), 187–197 (2000)

18. Gao, S.H., Cheng, M.M., Zhao, K., Zhang, X.Y., Yang, M.H., Torr, P.: Res2net: A new multi-scale backbone architecture. IEEE transactions on pattern analysis and machine intelligence **43**(2), 652–662 (2019)

19. Girdhar, R., Carreira, J., Doersch, C., Zisserman, A.: Video action transformer network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 244–253 (2019)

20. Girdhar, R., Grauman, K.: Anticipative video transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 13,505–13,515 (2021)

21. Goyal, R., Kahou, S.E., Michalski, V., Materzynska, J., Westphal, S., Kim, H., Haenel, V., Fruend, I., Yianilos, P., Mueller-Freitag, M., et al.: The" something something" video database for learning and evaluating visual common sense. In: Proceedings of the IEEE international conference on computer vision, vol. 1, p. 5 (2017)

22. Hara, K., Kataoka, H., Satoh, Y.: Learning spatio-temporal features with 3d residual networks for action recognition. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 3154–3160 (2017)

23. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016)

24. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2462–2470 (2017)

25. Jia, X., De Brabandere, B., Tuytelaars, T., Gool, L.V.: Dynamic filter networks. Advances in neural information processing systems **29**, 667–675 (2016)

26. Jiang, B., Wang, M., Gan, W., Wu, W., Yan, J.: Stm: Spatiotemporal and motion encoding for action recognition. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2000–2009 (2019)

27. Kanojia, G., Kumawat, S., Raman, S.: Attentive spatio-temporal representation learning for diving classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (2019)

28. Khowaja, S.A., Lee, S.L.: Semantic image networks for human action recognition. International Journal of Computer Vision (2020)

29. Kwon, H., Kim, M., Kwak, S., Cho, M.: Motionsqueeze: Neural motion feature learning for video understanding. In: European Conference on Computer Vision, pp. 345–362. Springer (2020)

30. Li, Y., Ji, B., Shi, X., Zhang, J., Kang, B., Wang, L.: Tea: Temporal excitation and aggregation for action recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 909–918 (2020)

31. Li, Y., Li, Y., Vasconcelos, N.: Resound: Towards action recognition without representation bias. In: Proceedings of the European Conference on Computer Vision, pp. 513–528 (2018)

32. Lin, J., Gan, C., Han, S.: Tsm: Temporal shift module for efficient video understanding. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 7083–7093 (2019)

33. Liu, Z., Luo, D., Wang, Y., Wang, L., Tai, Y., Wang, C., Li, J., Huang, F., Lu, T.: Teinet: Towards an efficient architecture for video recognition. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 11,669–11,676 (2020)

34. Liu, Z., Wang, L., Wu, W., Qian, C., Lu, T.: Tam: Temporal adaptive module for video recognition. arXiv preprint arXiv:2005.06803 (2020)

35. Lu, C., Shi, J., Wang, W., Jia, J.: Fast abnormal event detection. International Journal of Computer Vision **127**(8), 993–1011 (2019)

36. Luo, C., Yuille, A.L.: Grouped spatial-temporal aggregation for efficient action recognition. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5512–5521 (2019)

37. Ma, C.Y., Kadav, A., Melvin, I., Kira, Z., AlRegib, G., Peter Graf, H.: Attend and interact: Higher-order object interactions for video understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6790–6800 (2018)

38. Mahdisoltani, F., Berger, G., Gharbieh, W., Fleet, D., Memisevic, R.: Fine-grained video classification and captioning. arXiv preprint arXiv:1804.09235 **5**(6) (2018)

39. Materzynska, J., Xiao, T., Herzig, R., Xu, H., Wang, X., Darrell, T.: Something-else: Compositional action recognition with spatial-temporal interaction networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1049–1059 (2020)

40. Plizzari, C., Cannici, M., Matteucci, M.: Skeleton-based action recognition via spatial and temporal transformer networks. Computer Vision and Image Understanding **208**, 103,219 (2021)

41. Ranjan, A., Black, M.J.: Optical flow estimation using a spatial pyramid network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4161–4170 (2017)

42. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE transactions on pattern analysis and machine intelligence **39**(6), 1137–1149 (2016)

43. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Advances in neural information processing systems, pp. 568–576 (2014)

44. Srinivas, A., Lin, T.Y., Parmar, N., Shlens, J., Abbeel, P., Vaswani, A.: Bottleneck transformers for visual recognition. arXiv preprint arXiv:2101.11605 (2021)

45. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8934–8943 (2018)

46. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: Thirty-first AAAI conference on artificial intelligence (2017)

47. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9 (2015)

48. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818–2826 (2016)

49. Tian, Y., Che, Z., Bao, W., Zhai, G., Gao, Z.: Self-supervised motion representation via scattering local motion cues. In: European Conference on Computer Vision, pp. 71–89. Springer (2020)

50. Tian, Y., Lu, G., Min, X., Che, Z., Zhai, G., Guo, G., Gao, Z.: Self-conditioned probabilistic learning of video rescaling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4490–4499 (2021)

51. Tian, Y., Min, X., Zhai, G., Gao, Z.: Video-based early asd detection via temporal pyramid networks. In: 2019 IEEE International Conference on Multimedia and Expo, pp. 272–277. IEEE (2019)

52. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. arXiv preprint arXiv:2012.12877 (2020)

53. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: Proceedings of the IEEE international conference on computer vision, pp. 4489–4497 (2015)

54. Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 6450–6459 (2018)

55. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. arXiv preprint arXiv:1706.03762 (2017)

56. Wang, H., Tran, D., Torresani, L., Feiszli, M.: Video modeling with correlation networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 352–361 (2020)

57. Wang, L., Li, W., Li, W., Van Gool, L.: Appearance-and-relation networks for video classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1430–1439 (2018)

58. Wang, L., Tong, Z., Ji, B., Wu, G.: Tdn: Temporal difference networks for efficient action recognition. arXiv preprint arXiv:2012.10071 (2020)

59. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporal segment networks: Towards good practices for deep action recognition. In: European conference on computer vision, pp. 20–36. Springer (2016)

60. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporal segment networks for action recognition in videos. IEEE transactions on pattern analysis and machine intelligence **41**(11), 2740–2755 (2018)

61. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7794–7803 (2018)

62. Wang, X., Gupta, A.: Videos as space-time region graphs. In: Proceedings of the European conference on computer vision, pp. 399–417 (2018)

63. Wu, Z., Li, H., Zheng, Y., Xiong, C., Jiang, Y.G., Davis, L.S.: A coarse-to-fine framework for resource efficient video recognition. International Journal of Computer Vision (2021)

64. Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K.: Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In: Proceedings of the European Conference on Computer Vision, pp. 305–321 (2018)

65. Yang, B., Bender, G., Le, Q.V., Ngiam, J.: Condconv: Conditionally parameterized convolutions for efficient inference. arXiv preprint arXiv:1904.04971 (2019)

66. Zach, C., Pock, T., Bischof, H.: A duality based approach for real-time tv-l 1 optical flow. In: Joint pattern recognition symposium, pp. 214–223. Springer (2007)

67. Zhang, C., Zou, Y., Chen, G., Gan, L.: Pan: Towards fast action recognition via learning persistence of appearance. arXiv preprint arXiv:2008.03462 (2020)

68. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 6848–6856 (2018)

69. Zhang, Y., Li, X., Liu, C., Shuai, B., Zhu, Y., Brattoli, B., Chen, H., Marsic, I., Tighe, J.: Vidtr: Video transformer without convolutions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 13,577–13,587 (2021)

70. Zhou, B., Andonian, A., Oliva, A., Torralba, A.: Temporal relational reasoning in videos. In: Proceedings of the European Conference on Computer Vision, pp. 803–818 (2018)

71. Zolfaghari, M., Singh, K., Brox, T.: Eco: Efficient convolutional network for online video understanding. In: Proceedings of the European conference on computer vision, pp. 695–712 (2018)