Attention Before and After Feature Extraction for Action Recognition

1st Zichen Zhou
SEIEE
Shanghai Jiao Tong University
Shanghai, China
zzcbrink@sjtu.edu.cn

2nd Yiling Xu
SEIEE
Shanghai Jiao Tong University
Shanghai, China
yl.xu@sjtu.edu.cn

3rd Le Yang
Department of ECE
University of Canterbury
Christchurch, NZ
le.yang@canterbury.ac.nz

***Abstract*—Two challenges remain in video action recognition: (1) long-range spatial-temporal dependencies play important roles in action recognition, but they are difficult to be captured in a light-weight way; (2) frames that are irrelevant or redundant for action recognition exist in videos, especially in untrimmed videos. In this paper, we introduce a novel network architecture augmented with a frame attention module before feature extraction and a cascaded self-attention module after feature extraction. It can evaluate the importance of each sampled frame and at the same time, capture long-range spatio-temporal dependencies of feature maps in a light-weight way. In the task of 3D feature processing, the proposed cascaded self-attention module costs much less FLOPs than the non-local block. By using efficient networks, we obtain an inference speed up to 400 frames per second. Furthermore, competitive performance with respect to the state-of-the-art methods on datasets UCF101, HMDB51 and Kinetics is demonstrated.**

***Index Terms*—Action recognition, Attention mechanism, 3D convolution**

I. Introduction

Noise greatly degrades the accuracy of action recognition. A popular framework for performing human action recognition in videos is applying the pooling operation to aggregate the features extracted from RGB streams or optical flows into a summary vector. Mean and max pooling are typical choices. However, these pooling methods consider all frames equally and as a result, they are not robust to noise [1]–[4].

Noise mentioned above mainly refers to the redundant and irrelevant frames, i.e. the video frames that are uncorrelated with the target action [6], [7]. These frames not only increase the computational cost but also deteriorate the accuracy of model. In addition, videos in popular datasets, such as UCF101 [8], HMDB51 [9] and Kinetics [10], are mostly trimmed. For untrimmed videos, the interference from noise is more prominent. Temporal segment network (TSN) [5] adopts a sparse and global temporal sampling strategy to sample frames from the entire video evenly. It divides every input video into K segments and selects a frame randomly from each segment in order that it can avoid sampling redundant frames up to a point. Unavoidably, irrelevant frames still could be sampled and fed to the subsequent networks.

Long-range dependencies along spatial and temporal dimensions remain a central problem because classical convolutional neural networks (CNNs) , owing to their structures, are limited to local receptive fields and short-range contextual information. For example, although 3D-CNNs are popular frameworks for capturing temporal relationships between sampled frames and show good performance, it can only realize small windows because of the size of convolution kernel, i.e. only short-term relationships are captured in convolution operations. Existing methods also use some post-hoc integration of window-based scores, but this is suboptimal for exploiting the temporal relationships between windows [11].

Non-local neural networks [12] employ a self-attention operation, which computes the response at a position as a weighted sum of the features at all positions in the input feature maps, leading to more powerful pixel-wise representations. Here, each pixel in the feature maps is influenced by all other pixels through self-attention maps, thus enabling the exploration of long-range contextual information. However, despite the outstanding performance, the requirement for tuning a large number of parameters and high FLOPs make this method difficult to be used in real-time action recognition.

In this paper, we make the following contributions. (1) We propose an end-to-end trainable architecture augmented with a frame attention module based on frame difference to eliminate some of the non-discriminative frames sparsely sampled from the entire video. After the frame attention operation, the effectiveness of frames is enhanced, fewer frames need to be processed as well. (2) We propose a lightweight self-attention module for capturing long-range spatio-temporal relationships of 3D feature maps. Compared to the non-local block, it replaces 3D self-attention block with 3 cascaded 2D self-attention blocks and therefore it consumes much less FLOPs than non-local block. (3) We carry out ablation studies to verify the effectiveness of two proposed attention modules and validate the proposed network on video benchmark datasets UCF101, HMDB51 and Kinetics. Compared to state-of-the-art approaches, competitive performance in terms of classification accuracy is obtained when only RGB information is used, meanwhile the advantage of inference speed is prominent.

This paper is supported in part by National Natural Science Foundation of China (61971282), National Key Research and Development Project of China Science and Technology Exchange Center (2018YFE0206700) and Scientific Research Plan of the Science and Technology Commission of Shanghai Municipality (18511105402). The corresponding author is Yiling Xu.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

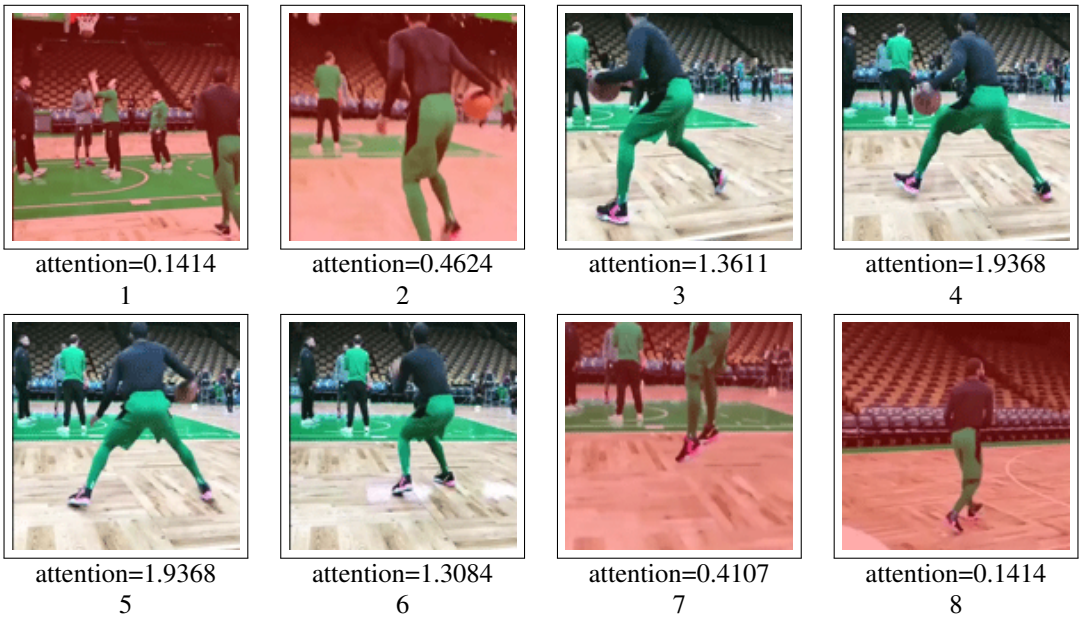


Fig. 1. Visualizations of frame attention. We sampled 8 frames from an Internet video and ran through our network trained on Kinetics. The frames that are remarked with red masks are discarded because they are considered to have low importance according to their attention. Avoiding these frames being input to the subsequent module reduces the computational cost substantially.

II. Related Works

Action classification architectures. For action recognition, many works utilized 2D architectures. [2] proposed the two-stream convolutional network with one stream for capturing spatial features (using RGB infomation) and another for temporal context (using optical flow). [13] highlighted a drawback in the two-stream network that uses a standard image CNN instead of a specialized network for processoing videos. The two-stream networks are not able to capture long-term temporal information [14]. Recurrent neural networks (RNNs) are also a popular method. [15] used a LSTM to integrate features extracted by CNNs. However, the performance of RNN on action recognition currently is inferior to recent CNN-based methods, which indicate that they may fail to model long-term dynamics well [11] .

3D convolutional network is a preferred choice. C3D (Convolutional 3D) [16] introduced a 3D architecture with 3D kernels to learn the spatial-temporal features from a sequence of frames. In a more recent work, they studied the use of the ResNet architecture with 3D convolutions and showed the improvements over their earlier C3D architecture [17]. Several works also utilized 3D architectures [18], [19]. But these approaches can only capture the short-term temporal context of the input video based on a sliding window [11], [20], [21]. The presence of Kinetics changed the prospect of action classification. [22] proposed that the Kinetics dataset has sufficient data for training deep 3D CNNs, and enables training up to 152 ResNet layers. Meanwhile, they showed that Kinetics pretrained simple 3D architectures outperform complex 2D architectures. In terms of inference speed, however, these methods are quite slow because they have to compute the average score over multiple windows, which is quite time consuming.

Different from 3D-Net approaches, TSN [5] divided the whole video into several segments and utilized a segmental consensus strategy to capture long-range relationships. In each segment, a sparse and global sampling method is used to choose frames from the entire video. However, as in its aggregation methods, frames are processed independently for inference and their class scores are aggregated only at the end. Consequently, the performance in their experiments stays the same when they change the number of samples, which indicates that their model does not really benefit from the long-term information. Similar to TSN, efficient convolutional network (ECO) [11] also sparsely samples frames from the entire video to capture long-range temporal features. In this way, the sampled frames span the entire video independent of the video length. In contrast to TSN, 3D-networks are used to learn the relationship between frames. The network is trained in an end-to-end manner to learn this relationship and directly provides video-level scores without post-hoc feature aggregation so that this model is computationally inexpensive.

Noise processing. Adaptive scan pooling (Adascan) [14] proposed a method to pool discriminative and informative frames, while discarding redundent and irrelevent frames in a single temporal scan of the video. Different from original pooling methods considering all frames equally, Adascan learns to dynamically pool video frames for action classification by producing interpretable intermediate state of each frame. In Adascan, the video is input to the network frame by frame, so the state of each frame is obtained by comparing the current pooled features and features from the incoming frame.

Attention mechanism. Attention mechanism is widely used for capturing long-range dependencies. Non-local neural networks [12] was proposed to generate attention maps by calculating the correlation matrix between each spatio-temporal pixel in the stacked feature maps. Interaction-aware spatio-temporal pyramid attention networks [23] used feature maps with different scales to construct a spatial pyramid and then utilized multi-scale information to obtain more accurate attention scores. Convolutional block attention module (CBAM) [24] proposed a lightweight approach to infer attention maps along channel and spatial dimensions separately. For channel attention, they first aggregated spatial information of a feature map by using both average pooling and max pooling operations as two different spatial context descriptors and then input them to a multi-layer perceptron to obtain channel attention. For spatial attention, they applied average-pooling and max-pooling operations along the channel axis and concatenate them to generate a feature descriptor.

III. Approach

In this section, we present the details of our networks and the specifications of the frame attention module and cascaded self-attention module for action recognition.

A. Network Architecture

We choose Efficient Convolutional Network (ECO) and its lite version ECO-Lite [11] as our baseline networks. The structures of them are shown in Table 1. The first part of BN-Inception (from con1_x to inception3c) [25] is used as 2D feature extractor, several layers of 3D-Resnet-18 (from conv3_x to the end) [17] are used as 3D feature extractor. We add a frame attention module and a cascaded self-attention module into ECO and ECO-Lite so that we can validate the performance of them. The whole networks are named BAnet (Before-and-after Attention net) and BAnet-Lite.

BAnet adopts part of the BN-Inception architecture (from the inception-4a layer to the last pooling layer) in parallel with 3D-nets as another branch for capturing features compared to BAnet-Lite. The structure of BAnet-Lite is shown in Fig. 2. First, each video is split into N segments of equal length as the input to the network. From each segment a single frame is randomly sampled. In this way, frames which are non-discriminative may still be sampled, e.g. sports videos contain some shots about audience and sports fields distractive to action recognition.

TABLE I
Architecture of ECO-Lite. In ECO, BN-Inception architecture from the inception-4a layer to the last pooling layer is used in parallel with 3D-nets, in the end, the features extracted by two branches are concated.

layer name	output size	kernel size / stride
conv1_x	$112 \times 112 \times 64$	2Dconv 7×7 / 2
pool1	$56 \times 56 \times 64$	maxpool $3 \times 3/2$
conv2_x	$56 \times 56 \times 192$	2Dconv 3×3 / 1
pool2	$28 \times 28 \times 192$	maxpool 3×3 / 2
inception(3a)	$28 \times 28 \times 256$	
inception(3b)	$28 \times 28 \times 320$	
inception(3c)	$28 \times 28 \times 96$	
conv3_x	$28 \times 28 \times C$	$\begin{bmatrix}\text{3Dconv } 3 \times 3 \times 3, 128\\ \text{3Dconv } 3 \times 3 \times 3, 128\end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times C/2$	$\begin{bmatrix}\text{3Dconv } 3 \times 3 \times 3, 256\\ \text{3Dconv } 3 \times 3 \times 3, 256\end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times C/4$	$\begin{bmatrix}\text{3Dconv } 3 \times 3 \times 3, 512\\ \text{3Dconv } 3 \times 3 \times 3, 512\end{bmatrix} \times 2$
	$1 \times 1 \times 1$	pooling, fc, softmax

Then, sampled frames are input to subsequent networks which consist of four main modules: (i) frame attention module, (ii) 2D feature extractor, (iii) cascaded self-attention module and (iv) 3D feature extractor. The purpose of the frame attention module is to measure the importance of each sampled frame and proportionally eliminate low-importance frames. The 2D feature extractor (the first part of the BN-Inception architecture, from con1_x to inception-3c) is responsible for extracting 2D features from the remaining sampled frames. The cascaded self-attention module is in charge of capturing long-range spatiotemporal dependencies by computing self-correlation of the stacked feature maps. The 3D feature extractor (several layers of 3D-Resnet-18, from conv3_x to the end) is responsible for capturing deep spatiotemporal features.

The output of 2D feature extractor is stacked respectively according to the channels and fed as a single tensor $M = [M_1, M_2, \cdots, M_C]$, $M \in \mathbb{R}^{T \times H \times W \times C}$ to the cascaded self-attention module, where $C = 96$ is the output channels of the 2D feature extractor, T is the number of sampled frames, and $H = W = 28$ is the size of 2D feature maps.

We use only one cascaded self-attention block after feature map stacking because the scale of each feature map is already small (28×28) in this stage. The self-attention operation is better applied in shallow layers because much long-range information losed in deep layers. It is also justified in [12]. We conduct no experiment about adding 2D self-attention block on former layers because the main purpose of the cascaded self-attention module is to reduce the huge FLOPs of computing 3D attention maps. Whether the 2D self-attention module provides significant improvements to our baseline model could be a future research direction.

B. Frame Attention Module

As shown in Fig. 2, the frame attention module is applied after frame random sampling. This module is designed to eliminate non-discriminative frames. It achieves this purpose by computing a score that quantifies the discriminative importance of each sampled frame based on the difference maps obtained from each pair of adjacent frames. After importance evaluation, we discard part of the low-score frames while keeping the frame elimination percentage fixed.

1) Preprocessing: To reduce the complexity of the frame attention module, two preprocessing operations are employed.

First, we transform the raw RGB images to gray-scale images. Through this operation, three channels are compressed into one because it is unnecessary to use the whole information from the RGB channels to evaluate the importance of

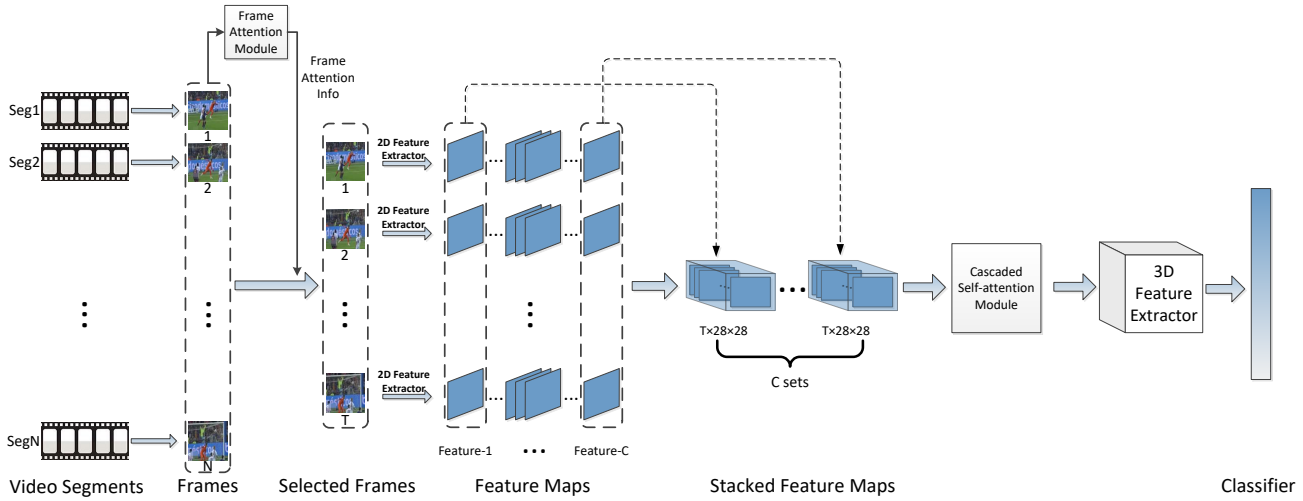


Fig. 2. Architecture of BAnet-Lite. First, We split each video into N segments averagely, from each segment a single frame is randomly sampled. Next, the sampled frames are input to the frame attention module, through this module we obtain the frame attention info which contains the score of each frame. Then, we select T high-score frames from N ($T \leq N$) as the input to the subsequent module. The 2D feature extractor extracts C feature maps from each frame and these feature-maps are stacked respectively according to the channel. After that, stacked feature maps are processed by cascaded self-attention module. Finally, they are fed into 3D feature extractor for capturing deep spatiotemporal features.

frames. We produce frame difference images from each pair of adjacent gray-scale frames.

Second, we use a max-pooling layer to downsample frame difference images. Irrelevant frames mostly come from different cameras or fast-moving lens, in other words, there are obvious differences existing in terms of views between irrelevant and normal frames. Thus, low-resolution images are sufficient to differentiate whether a frame is discriminative or not.

We illustrate these preprocessing procedures with an example. A video is uniformly divided into N segments, then we randomly sample a frame from each segment and compose them as the original input $X \in \mathbb{R}^{N\times 3\times 224\times 224}$. Compressing them into gray-scale images reduces the tensor into $X_g \in \mathbb{R}^{N\times 224\times 224}$. Next, we compute the difference between adjacent frames to obtain a frame difference tensor $X_d \in \mathbb{R}^{(N-1)\times 224\times 224}$. Finally, X_d is downsampled to $X'_d \in \mathbb{R}^{(N-1)\times 56\times 56}$ and taken as input of the multilayer perceptron with three layers for the task of frame attention. If we need to reduce more parameters, X_d could be downsampled to a smaller size.

2) Multilayer Perceptron: The frame attention module adopts a Multilayer Perceptron (MLP) with three layers. Each layer applies the ReLU function. The attention of difference maps is computed using:

$$Y_{diff} = [d_1, d_2, \cdots, d_{N-1}] = MLP(X'_d) \tag{1}$$

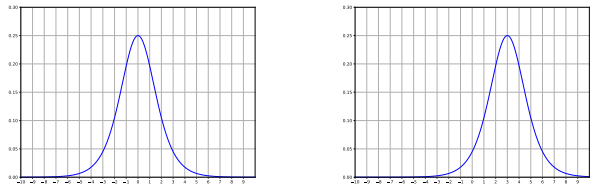
As mentioned above, $X'_d \in \mathbb{R}^{(N-1)\times 56\times 56}$ is the input of the frame attention module. The output of the MLP is $Y_{diff} \in \mathbb{R}^{(N-1)\times 1\times 1}$, of which each element $d_i, i \in [1, N-1]$, represents the attention of each frame difference map. Afterwards, the frame difference attention is converted to frame attention so that we can directly select the k most important frames. Specifically, the conversion is performed according to Eq. (2).

$$y_i = \begin{cases} F(clamp(d_i)) \times 2, & i = 1, \\ F(clamp(d_i)) + F(clamp(d_{i+1})), & 1 < i < N, \\ F(clamp(d_{i-1})) \times 2, & i = N. \end{cases} \tag{2}$$

where y_i, $i \in [1, N]$, denotes the attention of frame i, $clamp$ represents the function for clamping d_i into $[0, 7]$ in order to reduce computational cost. F is the right-translated derivative of the sigmoid function (see Fig. 3 for an illustration) used to further suppress the attention of redundant and irrelevant frames.

Compared with directly converting difference attention to frame attention (without using clamping and nonlinear processing via applying function F), the method shown in Eq. (2) has better performance owing to the following aspects:

First, for the task of eliminating the frames which differ either greatly or little from their adjacent frames (they are considered as irrelevent frames and redundant frames respectively), we need a function to further suppress d_i when it is close to 0 or 7 (the meaning underlying the value 0 and 7 will be explained later). We choose the derivation of sigmoid function and translate it to the right by 3, as Fig. 3(b) shows, roughly speaking, this function increases the nonlinearity of the MLP. The selected offset of 3 represents the most appropriate frame difference level for capturing temporal dependency. We select 3 as the offset rather than 3.5 (average value of 0 and 7) for the reason that we consider it is desirable to



(a) (b)

Fig. 3. The derivative of the sigmoid function and its right-translated version. By this processing, the scores of frame difference images close to 0 or 7 are further suppressed.

retain redundant frames rather than irrelevant frames because redundant frames would not introduce interference into the model. Thus, scores close to 7 are suppressed more heavily than scores close to 0, as Fig. 3(b) shows.

Second, for the purpose of avoiding large score disparity and reducing computational cost, we use the clamp function to restrain the output range of MLP into [0,7] because the output of derivative of the sigmoid changes only marginally when the input value falls out of [0,7], as Fig. 3(a) shows.

At last, we retain partial discriminative frames according to the frame attention as the input of subsequent networks, the retention rate is adjustable.

C. Cascaded Self-Attention Module

As shown in Fig. 4, for capturing long-range dependencies over local feature representations using a lightweight structure, the main part of the cascaded self-attention module consists of 3 cascaded 2D blocks. Here, we use an example to illustrate the cascaded self-attention module in details. We denote the features extracted by former network as $Z_{ori} \in \mathbb{R}^{T\times H\times W\times C}$, an $1 \times 1 \times 1$ convolution layer is applied to it for dimension reduction. By this operation, we obtain $Z \in \mathbb{R}^{T\times H\times W\times C'}$, where C' represents the channels of 3D feature maps, which is less than C. Then, Z is input to the HW block, the structure of HW block is illustrated in Fig. 6. In HW block, 3D feature maps are first split by dimension T into T sets. The FLOPs of correlation matrix computation of each set (excluding softmax) can be computed as:

$$F_{2D} = C_{in}(HW)^2 \times 2 \tag{3}$$

The FLOPs of HW block is therefore:

$$F_{HW} = C_{in}T(HW)^2 \times 2 \tag{4}$$

where C_{in} denotes the input channels, H, W are the size of 2D feature maps. After processing by the HW block, we can consider that the output features $Z_{HW} \in \mathbb{R}^{T\times H\times W\times C'}$ obtain the self-attention of HW view. Then Z_{HW} is input to TW and TH blocks sequentially. As shown in Fig. 5, owing to the cascaded structure, each voxel of feature map is influenced by any other voxel. It is worth mentioning that the performance of cascaded self-attention module is not sensitive to the sequence of 2D blocks, as shown by the experiment results in Table 5.

We can easily compare the FLOPs between cascaded self-attention module and non-local block. The FLOPs of correlation matrix computation using non-local block can be computed as:

$$F_{3D} = C_{in}(THW)^2 \times 2 \tag{5}$$

where T, H, W are the size of 3D feature maps.

The total FLOPs of correlation matrix computation using cascaded self-attention module can be computed as:

$$F_{CS} = C_{in}(T(HW)^2 + H(TW)^2 + W(TH)^2) \times 2 \tag{6}$$

Thus the ratio of F_{CS} to F_{3D} is:

$$R = \frac{F_{CS}}{F_{3D}} = \frac{1}{T} + \frac{1}{H} + \frac{1}{W} \tag{7}$$

According to Eq. (7), it is obvious that cascaded self-attention module costs much less FLOPs than the non-local block. In our experiments, $T = 16$, $H = W = 28$, therefore $R = 13.4\%$. With the raise of T, H, W, more computational cost is saved.

IV. Experiments

We evaluated the BAnet on three popular human action classification datasets UCF101, HMDB51 and Kinetics in order to compare its performance against the state-of-the-art methods. The comparison is restricted to approaches that only take the raw RGB information of videos as input without further pre-processing, e.g. by providing optical flow or human skeleton. The term BAnet(MF) in tables represents a network that selects M frames from N sampled frames as input to the feature extractor. The term BAnet-Lite denotes the lite version of BAnet. The ratio of M to N is 50% in our experiments.

A. Training

We train our model using mini-batch gradient descent with Nesterov accelerated gradient and employ dropout in every FC layer. In addition, we apply the data augmentation methods introduced by wang et al. [5]: we resize each sampled frames to 240×320 and adopt fixed-corner cropping and scale jittering with horizontal flipping (temporal jittering provided by sampling). Then, we run per-pixel mean subtraction and resize the cropped regions to 224×224.

We also apply the learning rate dropping strategy introduced in [5] : the initial learning rate is set to 0.001 and decreases to its 1/10 every 30 training epochs. The whole training procedure is about 90 epochs. We train the network with a momentum of 0.9, a weight decay of 0.0005, and batchsize of 16.

The weights of 2D feature extractor are initialized with the BN-Inception architecture [25] pretrained on Kinetics-400, as provided by [5]. The weights of 3D feature extractor are initialized with model of 3D-Resnet-18 pretrained on Kinetics-400, as provided by [19]. Other weights are randomly initialized. Afterwards, we train BAnet and BAnet-Lite on Kinetics.

For UCF101 and HMDB51, we finetune the BAnet and BAnet-Lite models on each one after training on Kinetics. We

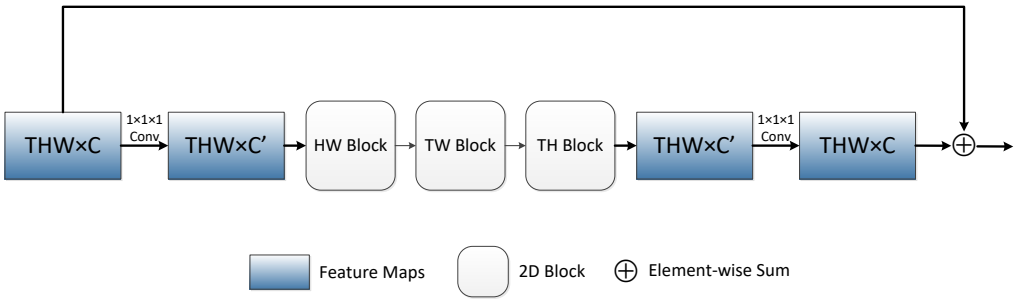


Fig. 4. The structure of cascaded self-attention module. The feature maps are shown as the size of their tensors, e.g. $THW \times C$ denotes feature maps sized THW for C channels. The structure of 2D block is shown in Fig. 6.

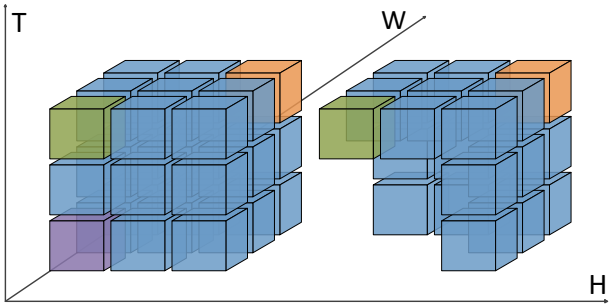


Fig. 5. We demonstrate the effect of the cascaded structure using a $3 \times 3 \times 3$ feature map. Without cascaded structure, if we split the cube by different dimensions, apply 2D self-attention operation to it respectively, and just add weights up at last, the orange voxel can only influences part of voxels, as the right cube shows, excluding the purple one. By cascaded structure, the orange voxel can influence the purple voxel through the influence to the green one.

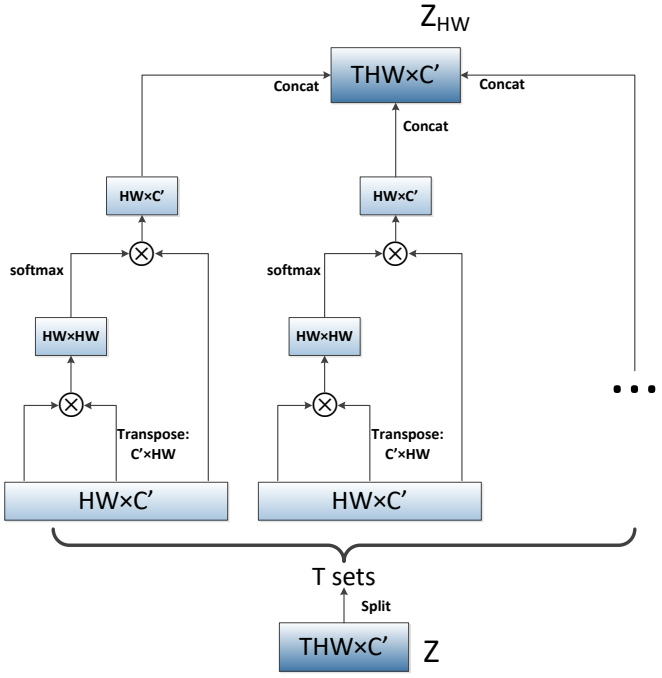


Fig. 6. Structure of HW block. In HW block, 3D feature maps are split by dimension T. "$\otimes$" denotes matrix multiplication. First, the 3D feature maps are split into T sets. For example, $Z \in \mathbb{R}^{T \times H \times W \times C'}$ is split into T sets by dimension T, each set $Z_{split} \in \mathbb{R}^{H \times W \times C'}$. At last, T sets processed features are concated into $Z_{HW} \in \mathbb{R}^{T \times H \times W \times C'}$. The softmax operation is performed on each row. TH block and TW block have the same structures with HW block, the only difference is 3D feature maps are split by different dimensions.

adjust the dropout rate and the epochs according to the size of datasets.

B. Accuracy Comparison

The accuracy performance obtained with BAnet and BAnet-Lite on UCF101 and HMDB51 are shown in Table 2. Compared to the state-of-the-art methods including both 2D and 3D, BAnet outperforms most existing methods except I3D, however, our network is much more lightweight with slight accuracy disadvantages. As shown in Table 3, BAnet outperforms the other methods including I3D on larger dataset Kinetics, which shows the strong performance of our architecture.

We can also see the influence on the number of sampled frames to our model, with more frames sampled, the accuracy rises. However, this trend slows down when more than 20 frames sampled because too dense sampling leads to converse effect for simple short-term actions.

C. Inference Speed Comparison

Previous works typically measure the speed of an approach in Frames Per Second (FPS). BAnet runs at 400 FPS on a GTX 1080Ti GPU without considering I/O. But FPS can hardly reflect the inference speed of videos because of different sampling density, e.g. for the same video, sampling 8 frames costs less time than sampling 16 frames. Thus, we adopt the evaluation standard VPS (Videos Per Second) proposed by ECO, which can reflect the efficiency of our model exactly.

Benefited from comparatively shallow architecture, BAnet obtains high inference speed. Table 3 shows the inference speed comparison between BAnet and state-of-the-arts on

TABLE II
TOP-1 ACCURACIES(%) ON UCF101 AND HMDB51. ALL ACCURACIES ARE AVERAGED OVER THREE SPLITS. DIM INDICATE THE DIMENSION OF CONVOLUTION KERNEL (IF 3D KERNEL IS USED IN A NETWORK, WE REGARD ITS DIM AS 3D).

Method	Dim	UCF101 (%)	HMDB51 (%)
2-Stream CNN [2]	2D	88.0	59.4
TSN(RGB) [5]	2D	87.7	51.0
AdaScan [14]	2D	89.4	54.9
Res3D [17]	3D	85.8	54.9
ResNeXt-101 [22]	3D	90.7	63.8
I3D(RGB) [21]	3D	**95.1**	**74.3**
ECO-Lite(16F) [11]	3D	91.6	68.2
ECO(16F) [11]	3D	92.8	68.5
BAnet-Lite(4F)	3D	89.9	62.1
BAnet-Lite(8F)	3D	92.3	66.2
BAnet-Lite(16F)	3D	93.4	70.3
BAnet-Lite(20F)	3D	94.0	70.9
BAnet(8F)	3D	93.1	67.4
BAnet(16F)	3D	94.7	71.1

TABLE III
INFERENCE SPEED COMPARISON WITH STATE-OF-THE-ART APPROACHES USING GTX 1080TI GPU ON UCF101 DATASETS (SPLIT1). FOR INFERENCE SPEED, WE DO NOT TAKE TIME CONSUMPTION OF I/O INTO CONSIDERATION.

Method	Inference Speed (VPS)	Top-1 (%)
I3D(RGB) [21]	0.5	95.1
Res3D [17]	1.1	85.8
ARTNet [19]	2.9	93.5
TSN(RGB) [5]	11.8	87.7
ECO(16F) [11]	23.2	92.8
ECO-Lite(16F) [11]	33.1	91.6
BAnet-Lite(4F)	113.7	89.9
BAnet-Lite(8F)	57.0	92.3
BAnet-Lite(16F)	28.1	93.4
BAnet-Lite(20F)	21.9	94.0
BAnet(16F)	19.9	94.7

UCF101. BAnet and BAnet-Lite architecture yield competitive accuracy as other approaches at much faster inference speed.

D. Ablation Studies

To further prove the effectiveness of the frame attention module and cascaded self-attention module, we show ablation studys on the test set of Kinetics. In this section, we sampled 16 frames per video as input in all ablation studies. Obvious improvements are shown in Table 7, as the performance of the frame attention module and cascaded self-attention module. Meanwhile, we can discover that cascaded self-attention module makes more contribution. Table 5 shows that the sequence of 2D blocks has small impact on the classification accuracy. As shown in Table 6, compared with non-local block, the proposed cascaded self-attention module obtains higher VPS than the non-local block, which shows that the CS module is a more efficient way to capture long-range dependencies.

TABLE IV
PERFORMANCE OF ACCURACIES COMPARED WITH SOTA METHODS ON KINETICS-400 TEST SET. FOR FAIR COMPARISON, EACH ARCHITECTURE WITHOUT IMAGENET PRETRAINING.

Method	Top-1 (%)
C3D [16]	56.1
CNN+LSTM [10]	57.0
TSN(RGB) [5]	57.1
2-stream CNN [10]	61.0
I3D(RGB) [21]	68.4
ECO-Lite(16F) [11]	64.4
ECO(16F) [11]	67.8
BAnet-Lite(16F)	65.4
BAnet(16F)	**69.7**

TABLE V
ABLATION STUDY 1, COMPARISON ON DIFFERENT 2D-BLOCK SEQUENCES OF CASCADED SELF-ATTENTION MODULE (WITHOUT FRAME ATTENTION).

Method	Top-1 (%)
ECO [11]	67.8
BAnet(HW-TW-TH)	69.0
BAnet(TW-HW-TH)	68.9
BAnet(TH-TW-HW)	68.9

TABLE VI
ABLATION STUDY 2, COMPARISON BETWEEN CASCADED SELF-ATTENTION MODULE (CS) AND NON-LOCAL BLOCK (NL). FRAME ATTENTION MODULE IS NOT ADDED.

Method	Top-1 (%)	speed (VPS)
ECO(baseline) [11]	67.8	23.2
ECO(with 1 NL)	69.0	8.1
ECO(with 1 CS)	69.0	19.9

V. CONCLUSION

We proposed a novel Before-and-After Attention Network (BAnet) for human action recognition augmented with frame attention module and cascaded self-attention module. The frame attention module works by evaluating the importance of each sampled frame based on frame diffference maps before feature extraction. The cascaded self-attention module employs 3 cascaded 2D self-attention blocks to capture long-range spatiotemporal dependencies after feature extraction. Compared to 3D self-attention computation of non-local block, cascaded self-attention module is much more efficient.

We validated our method on three popular datasets of human actions UCF101, HMDB51 and Kinetics. The results showed that the method has advantages both in the fields of classification accuracy and inference speed.

TABLE VII
ABLATION STUDY 3, THE EFFECTIVENESS OF FRAME ATTENTION MODULE AND CASCADED SELF-ATTENTION MODULE.

Method	Top-1 (%)
ECO-Lite [11]	64.4
ECO [11]	67.8
BAnet-Lite(frame attention only)	64.8
BAnet-Lite(cascaded self-attention only)	65.2
BAnet-Lite(both)	65.4
BAnet(frame attention only)	68.1
BAnet(cascaded self-attention only)	69.0
BAnet(both)	69.7

REFERENCES

[1] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *CVPR 2008-IEEE Conference on Computer Vision & Pattern Recognition*. IEEE Computer Society, 2008, pp. 1–8.

[2] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[3] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4305–4314.

[4] H. Wang, D. Oneata, J. Verbeek, and C. Schmid, "A robust and efficient video representation for action recognition," *International Journal of Computer Vision*, vol. 119, no. 3, pp. 219–238, 2016.

[5] 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," in *European conference on computer vision*. Springer, 2016, pp. 20–36.

[6] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars, "Rank pooling for action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 4, pp. 773–787, 2017.

[7] J. C. Niebles, C.-W. Chen, and L. Fei-Fei, "Modeling temporal structure of decomposable motion segments for activity classification," in *European conference on computer vision*. Springer, 2010, pp. 392–405.

[8] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[9] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: a large video database for human motion recognition," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2556–2563.

[10] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[11] M. Zolfaghari, K. Singh, and T. Brox, "Eco: Efficient convolutional network for online video understanding," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 695–712.

[12] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7794–7803.

[13] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4694–4702.

[14] A. Kar, N. Rai, K. Sikka, and G. Sharma, "Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3376–3385.

[15] Z. Li, K. Gavrilyuk, E. Gavves, M. Jain, and C. G. Snoek, "Videolstm convolves, attends and flows for action recognition," *Computer Vision and Image Understanding*, vol. 166, pp. 41–50, 2018.

[16] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "C3d: generic features for video analysis," *CoRR, abs/1412.0767*, vol. 2, no. 7, p. 8, 2014.

[17] D. Tran, J. Ray, Z. Shou, S.-F. Chang, and M. Paluri, "Convnet architecture search for spatiotemporal feature learning," *arXiv preprint arXiv:1708.05038*, 2017.

[18] M. Zolfaghari, G. L. Oliveira, N. Sedaghat, and T. Brox, "Chained multi-stream networks exploiting pose, motion, and appearance for action classification and detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2904–2913.

[19] L. Wang, W. Li, W. Li, and L. Van Gool, "Appearance-and-relation networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1430–1439.

[20] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1510–1517, 2018.

[21] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[22] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?" in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 6546–6555.

[23] Y. Du, C. Yuan, B. Li, L. Zhao, Y. Li, and W. Hu, "Interaction-aware spatio-temporal pyramid attention networks for action classification," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 373–389.

[24] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.