# AARM: Action Attention Recalibration Module for Action Recognition

**Zhonghong Li** LIZHH43@MAIL2.SYSU.EDU.CN
**Yang Yi** ISSYY@MAIL.SYSU.EDU.CN
**Ying She** SHEY@MAIL2.SYSU.EDU.CN
**Jialun Song** SONGJL3@MAIL2.SYSU.EDU.CN
**Yukun Wu** WUYK7@MAIL2.SYSU.EDU.CN
*Guangdong Province Key Laboratory of Big Data Analysis and Processing, School of Data and Computer Science, Sun Yat-sen University*

**Editors:** Sinno Jialin Pan and Masashi Sugiyama

## Abstract

Most of Action recognition methods deploy networks pretrained on image datasets, and a common limitation is that these networks hardly capture salient features of the video clip due to their training strategies. To address this issue, we propose Action Attention Recalibration Module (AARM), a lightweight but effective module which introduces the attention mechanism to process feature maps of the network. The proposed module is composed of two novel components: 1) convolutional attention submodule that obtains inter-channel attention maps and spatial-temporal attention maps during the convolutional stage, and 2) activation attention submodule that highlights the significant activations in the fully connected process. Based on ablation studies and extensive experiments, we demonstrate that AARM enables networks to be sensitive on informative parts and gain accuracy increasements, achieving the state-of-the-art performance on UCF101 and HMDB51.

**Keywords:** Feature Modulation, Channel and Spatial-temporal Attention, Activation Attention, Action Recognition

## 1. Introduction

As a compelling topic of computer vision, human action recognition has drawn sustained attention from the research community due to its huge potential value in the industries like video surveillance and human-computer interaction. Before the introduction of AlexNet (Krizhevsky et al., 2012), handcrafted approaches led by iDT (Wang and Schmid, 2013) dominates the field of human action recognition. Since a 2D CNN (LeCun et al., 1998) pretrained on ImageNet (Russakovsky et al., 2015) achieved a state-of-the-art performance, researchers have preferred deep learning methods rather than handcrafted approaches. Although obtaining excellent performances, deep models still exist deficiencies, as they cannot benefit from the video temporal feature, a crucial feature for video action recognition. To address this, researchers develop many significant techniques. Two-stream network (Simonyan and Zisserman, 2014) introduces optical-flow as an extra input stream. 3D CNN (Tran et al., 2015) expands the 2D convolution to 3D and takes consecutive frames as input, and feature-encoding network (Diba et al., 2017) leverages temporal encoding as pooling

method. Those techniques overcome the flaws in some ways, making deep learning more suitable for action recognition.

On the other hand, the attention mechanism is one of the remarkable methods for computer vision task. The attention mechanism is to recalibrate the network to focus on important features and suppress unimportant ones. In literature, Hu et al. (2018) propose SENet using channel attention mechanism and achieve a notable performance on ImageNet classification task. Hu et al. (2019) leverage CSARNet for super resolution and reach the state-of-the-art result. It should be noted that most of the existing attention-based methods are proposed for the image-base task, which may not be optimal for the video-based action recognition task. Furthermore, how to handle spatial-temporal attention and fuse it with channel attention effectively also remain unsolved.

To tackle this issue, in this study, we propose a lightweight yet powerful module called Action Attention Recalibration Module (AARM). AARM consists of three submodules, Channel Attention Module (CAM) and Spatial-Temporal Attention Module (STAM) for the convolutional block like Resblock and Video Context Module (VCM) for the fully connected layer. CAM is leveraged to infer inter-channel relationships of convolutional feature maps and decide "what" to focus, STAM is leveraged to infer intra-channel relationships and decide "where" and "when" to focus, and VCM is leveraged to obtain attention maps among activations with less computational cost than the encoding approaches. Meanwhile, as the most discriminative part is the key to classification, highlight and dropout mechanisms are arranged in STAM to benefit the attention mechanism. Furthermore, instead of combining CAM and STAM simply in a sequential order, a parallel fusion manner by concatenation and convolution is proposed to preserve their features simultaneously. We demonstrate that AARM is suitable for any existing CNN architectures for video-base tasks by helping them understand what should be noticed or ignored through explicit feature weight maps. Finally, experimental results show that our method can achieve better performance compared with the state-of-the-art methods on two datasets UCF101 (Soomro et al., 2012) and HMDB51 (Kuehne et al., 2011).

Overall, our contributions can be summarized as follows.

- We propose a light-weight yet effective module AARM with CAM, STAM and VCM, which can recalibrate convolutional features and activation.
- We introduce the highlight and dropout mechanism for STAM and a parallel fusion strategy for CAM and STAM to further optimize AARM for video action recognition.
- We conduct extensive experiments on two human action recognition datasets to validate the effectiveness of the proposed submodules and compare AARM-based methods with the state-of-the-art to verify the performance of our module.

The rest of the paper is organized as follows: Section 2 summarizes the related works. Section 3 introduces our modules. Section 4 reports the performance of our modules. Section 5 concludes the whole paper.

## 2. Related Works

### 2.1. Action recognition approaches

The existing action recognition methods can be summarized into two categories. The first type is the two-stream based network, as Simonyan and Zisserman (2014) first introduce a two-stream convolutional network, which takes optical-flow as input for the first time. They decompose a video clip into RGB stream and optical-flow stream to represent spatial features and temporal features respectively. Wang et al. (2016) propose Temporal Segment Network based on the two-stream network, which divides a video clip into three snippets as input. Networks share parameters among snippets, and the predictions of all snippets are fused to produce the final prediction. Diba et al. (2017) propose Deep Temporal Linear Encoding Network inheriting TSN's architecture, and it deploys a bilinear encoding layer in the fusion stage.

The other is the 3D convolutional network, Tran et al. (2015) first introduce a 3D convolutional network, expanding convolution from 2D to 3D. However, due to the huge number of parameters and the difficulty of training, the performance is unsatisfactory. To address the issue of training, Carreira and Zisserman (2017) introduce Inflated 3D convolutional Network which makes pre-training on image datasets possible in virtue of inflating a pretrained 2D convolutional network into 3D and copying an image repeatedly into a "boring video". To address the issue of excessive parameters, Tran et al. (2019) propose channel-seperated convolutional networks to balance parameter cost and channel interaction by decomposing a 3D convolutional operation into $1 \times 1 \times 1$ convolution or $3 \times 3 \times 3$ depth-wise convolution. STM proposed by Jiang et al. (2019) capture temporal features by a combination of a 2D sematic module and a motion module, successfully replaceing 3D convolution by 2D. The above-mentioned architectures have already achieved state-of-the-art performance, while we believe that the attention mechanism can be a complementary way observed from our experiments.

### 2.2. Attention mechanism

Attention can be dated back to the human visual system (Corbetta and Shulman, 2002). The reason why a man can catch the point instantly is that the system does not handle the whole scene but selectively focuses on the interesting part to maximum informativeness. Consequently, applying the attention mechanism on networks is theoretically sound. Hu et al. (2018) propose Squeeze and Excitation module for image classification. The module exploits the relationship between convolutional channels and reweights features by element-wise multiplication. Kim et al. (2018) introduce a residual attention module RAM to capture inter-spatial attention. Moreover, Park et al. (2018) introduce spatial attention mechanism and propose Bottleneck Attention Module and Convolutional Block Attention Module that take spatial attention into account. BAM is placed after each bottleneck in a network while CBAM (Woo et al., 2018) is deployed after each convolutional block. Both two modules are effective in image classification and object detection tasks. Although BAM and CBAM are simple yet effective modules for image-based tasks, they may not be optimal for video tasks. Temporal feature is the key difference between image-base task and video-base task, and Wang et al. (2018) introduce self attention mechanism to capture temporal features and

propose Non-local Neural Network for 3D CNN. Based on these observations, our proposed AARM is closely related to CBAM, and further designed for video-base task, finding better temporal feature representation and fusion method. Furthermore, our proposed AARM is simple yet effective and can be easily injected into any existing methods including two-stream methods and 3D CNN methods.

## 3. Action Attention Recalibration Module

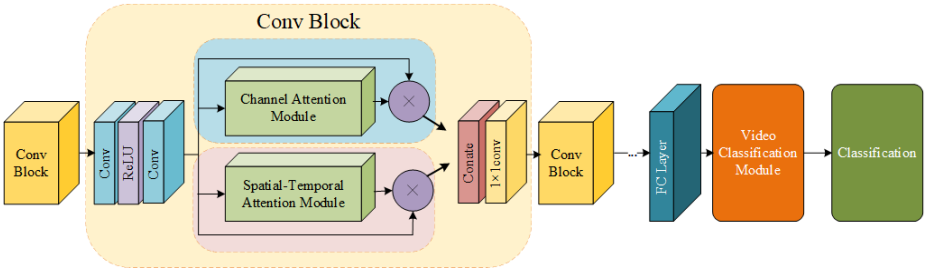


Figure 1: **The overview of AARM.** AARM has three submodules: Channel Attention Module(CAM), Spatial-Temporal Attention Module(STAM) depicted in green blocks, and Video Classification Module(VCM) depicted in orange block. CAM and STAM are deployed after convolutional layers in each convolutional block(e.g. ResBlock). They refine inter-channel attention and intra-channel attention from feature maps respectively. VCM is deployed after fully connected layers to optimize actions for classification.

Illustrated in Fig 1, the overall architecture of our AARM can be divided into two parts: convolution attention part (CAM and STAM) and classification attention part (VCM). Both CAM and STAM are plugged after convolutional layers in convolutional blocks, while VCM is arranged after fully connected layers.

In a convolutional block, given convolutional feature maps $M \in R^{H\times W\times C}$ in 2D networks or $R^{T\times H\times W\times C}$ in 3D networks as input, CAM obtains channel attention map $F_{ch}(M)$ and then STAM obtains spatial-temporal attention map $F_{st}(M)$. STAM varies over the dimension of network input, which is described in the following subsection. At last, two attention maps are fused and the recalibrated feature maps $Y$ is obtained. In summary, the process of convolutional block attention refinement can be shown as follows:

$$\begin{cases} M_{ch} = F_{ch}(M) \otimes M \\ M_{st} = F_{st}(M) \otimes M \\ Y = Fuse(M_{ch}, M_{st}) \end{cases} \tag{1}$$

where $\otimes$ represents element-wise multiplication, $F_{ch}(X) \in R^{C\times 1\times 1}$ or $R^{C\times 1\times 1\times 1}$ denotes the channel attention map, $F_{st}(X) \in R^{1\times H\times W}$ or $R^{1\times H\times W\times T}$ denotes the spatial-temporal attention map, and $Fuse(\cdot,\cdot)$ denotes the fusion strategy which is described in the following subsection.

VCM is deployed after fully connected layer before classification to recalibrate activation value, which emphasizes the discriminative activations through giving high weights and vice versa. For instance, in a "golf" video clip, the activations of "golf club" and "human" are significant while the activations of "grassland" and "audience" are not. VCM is learnt to highlight "golf" and "human" and conceal "grassland" and "audience". In this way, the deep network is more sensitive to the discriminative feature and more robust to ambiguous video clips. The overall process can be formulated as

$$M' = F_{vc}(M) \otimes M \tag{2}$$

where $F_{vc}(M)$ denotes the attention vector with the length $u$ and $u$ is the number of units in the fully connected layer $M$.

The following subsections describe the details of each submodule.

### 3.1. CAM

CAM focuses on exploiting the relationships among channels of convolutional maps and quantizing them into attention value. In most cases, channels of convolutional layer focus on different parts of the feature, where some of them involve discriminative features, whereas others are insignificant. But common deep networks handle all the channels equally, which may ignore those global clues in forward propagation. CAM is invented to address the shortcoming by modelling the channel relationships explicitly. Thus, CAM enables deep networks to capture inter-channel relationships and recognize "what" is important in the early stage.

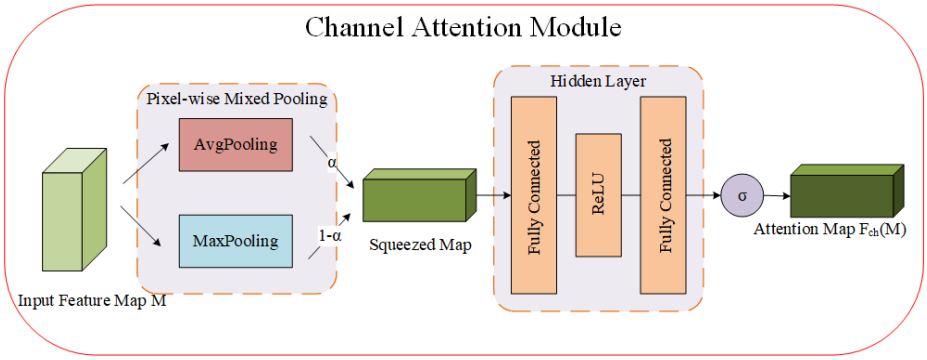


Figure 2: **The overview of Channel Attention Module.** Mixed pooling is first deployed to squeeze feature map to $C \times 1 \times 1$, where $\alpha \in [0, 1]$ is the weight of average pooling in mixed pooling. After that, the subsequent hidden preception layer is arranged to extract attention map. Then the attention map is normalized to [0,1] by a sigmoid function. Notice that the dimension of the hidden layer is reduced to $C/r \times 1 \times 1$ to restrict the number of parameters where $r$ is the reduction ratio, which is set to 16 in this paper.

The architecture of CAM is illustrated in Fig 2. In the previous work (Hu et al., 2018), average pooling method is adopted simply to squeeze feature maps. However, we claim that since action recognition needs salient components for classification, it is proper to take max pooling into consideration. To this end, we choose to leverage the pixel-wise mixed pooling with weight $\alpha$ rather than the average pooling method. Given a feature map $M \in R^{C\times H\times W}$, it is reduced to a weight vector with length $C$ by the pixel-wise mixed pooling operation. After aggregation, a recalibration operation composed of two fully connected layers with a hidden layer is arranged. To restrict the number of parameters, we reduce the dimension of the hidden layers to $C/r \times 1 \times 1$ where $r$ is the reduction ratio and empirically set to 16 as SENet. After recalibration, the attention maps are normalized to [0,1] by a sigmoid function and then element-wise multiplied by the input feature maps to reweight channel attention as shown in (1). Concisely, CAM can be summarized as:

$$F_{ch}(M) = \sigma(W_2\delta W_1(\alpha Pool_{Avg,Pixel}(M) + (1-\alpha)Pool_{Max,Pixel}(M))) \tag{3}$$

where $\sigma$ denotes the sigmoid function, $Pool_{\cdot,Pixel}$ denotes the pixel-wise pooling, $\delta$ denotes the ReLU function, $\alpha$ denotes the weight of average pooling in mixed pooling which is set to 0.5 in this paper, $W_1 \in R^{C/r\times C}$ and $W_2 \in R^{C\times C/r}$ represent fully connected layer and $r$ denotes the reduction ratio which is set to 16 in this paper.

CAM can be treated as a channel-wise weight optimizer. Given a feature map, CAM generates a weight vector whose length is the channel of the feature map, and redresses it by reweighting channels.

### 3.2. STAM

Different from CAM extracting inter-channel attention maps, STAM infers intra-channel attention maps, finds "where" and "when" are discriminative parts and recalibrates the input feature map. The overview of STAM is depicted in Fig 3. Considering 3D convolution having more parameters than 2D, we arrange (2+1)D STAM for 3D CNNs.

Apart from the attention map generated by the sigmoid function, we introduce an extra attention map, the highlighted map, which is generated by a threshold function, since the highlighted map further obtains the discriminative parts for better classification. Meanwhile, to prevent overemphasis resulting overfitting, we propose a dropout layer to stochastically select one as the final map for the subsequent processing. Note that the dropout layer has two hyperparameters: threshold $\eta$ and pick rate $\chi$. $\eta$ controls the size of the region to be highlighted, and $\chi$ controls the frequency of the highlighted map being selected.

Specifically, given the 2D input feature map $M \in R^{C\times H\times W}$, STAM first aggregate to a single channel map $M' \in R^{1\times H\times W}$, and then the attention map is generated by a convolutional layer. After that, the highlighted map and the normalized attention map are produced respectively through threshold and sigmoid function. For the former, we set each pixel to 1 if it is larger than the threshold and 0 if it is less, suggesting that the size increases as $\eta$ decreases and vice versa. For the latter, the attention map is normalized to [0,1] simply by a sigmoid function. While the highlighted map is emphasized the most discriminative parts for classification, the attention map focuses on the meaningful regions to avoid overfitting. Both two maps are weight maps $\in$ [0,1] and one is randomly chosen as the

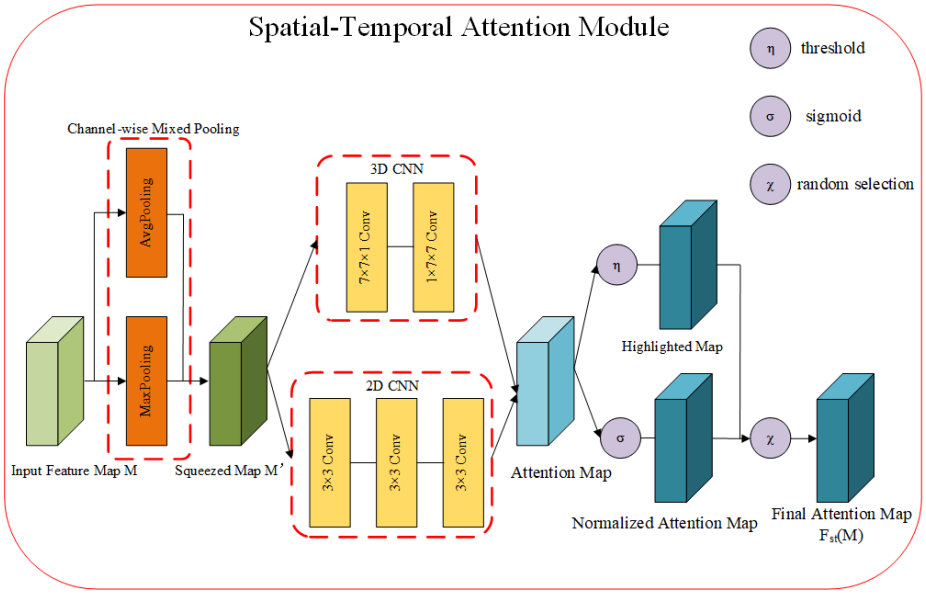


Figure 3: **The overview of Spatial-Temporal Attention Module.** For the input feature map $M \in R^{C\times H\times W}$ or $R^{C\times H\times W\times T}$, where $C$ denotes channel, $T$ denotes the temporal volume for 3D convolution and $H$ and $W$ denotes height and width respectively, channel-wise mixed pooling is leveraged to squeeze it to a single channel map. Then the convolutional layer extracts the map and constructs the highlighted map and attention map through threshold function and sigmoid function respectively. Finally the dropout mechanism works to select one of them stochastically as the final attention map for recalibration.

final attetion map. In this way, we can benefit from both attention and dropout mechanism and improve the classification performance. And the 3D feature map is processed similarly.

Note that for 2D convolutional networks, STAM does not simply perform $7 \times 7$ convolution with abundant parameters, but three consecutive $3 \times 3$ convolutional layers. Meanwhile, for 3D convolutional networks, since expanding convolutional operation from 2D to 3D leads to unbearable parameter growth and training cost, we introduce a (2+1)D convolutional layer for 3D STAM. It is claimed that factorizing 3D convolution as a sequential process of a 2D convolution and a 1D convolution reduces the number of parameters and complexity of convolutional operation (Sun et al., 2015). Based on this theory, as depicted in the bottom side of Fig 3, we design the 3D convolutional layer by a (2+1)D convolution one and expand the channel-wise mixed pooling layer to 3D. Furthermore, the dropout layer does not require any trainable parameters and is applied only during training as we need a stable classification result in testing phase. The overall process can be summarized as follows:

$$\begin{cases} F_{st}(M) = \begin{cases} threshold(F'_{ST}(M)) & \text{if } \chi \\ sigmoid(F'_{ST}(M)) & \text{others} \end{cases} \\ F'_{2DST}(M) = conv_{3\times3}(conv_{3\times3}(conv_{3\times3}(\alpha Pool_{Avg,Ch}(M)) + (1-\alpha)Pool_{Max,Ch}(M))) \\ F'_{3DST}(M) = conv_{7\times1\times1}(conv_{1\times7\times7}((\alpha Pool_{Avg,Ch}(M)) + (1-\alpha)Pool_{Max,Ch}(M))) \\ threshold(m) = 1 \qquad \text{for each } m \in M, m \geq \eta \times max(M) \\ threshold(m) = 0 \qquad \text{for each } m \in M, m < \eta \times max(M) \end{cases} \tag{4}$$

where $M$ denotes the input feature map, $Pool_{\cdot,Ch}$ denotes the channel-wise pooling, $\alpha$ denotes the weight of the mixed pooling and $threshold(\cdot)$ denotes the construction function of the highlighted map in the dropout layer.

### 3.3. Attention fusion

The proposed CAM and STAM exploit inter-channel and intra-channel relationships respectively. In this regard, there are two methods fusing two attention maps: 1) sequential fusion and 2) parallel fusion. Although many previous works (Zhang et al., 2018; Hu et al., 2018; Woo et al., 2018) combine two attention maps in a sequential way, we argue that this manner may result in the latter module being affected by the former module as the input of the latter one is the output of the former one. And the parallel manner can avoid this issue conveniently. Therefore, in our method, we choose the parallel strategy to fuse CAM and STAM simultaneously. In details, we first obtain channel attention map and spatial-temporal map respectively, then two attention maps are concatenated simply and $1 \times 1$ convolution is leveraged to combine the concatenated map. The whole process can be formulated as

$$Fuse(M_{ch}, M_{st}) = conv_{1\times1}([M_{ch}, M_{st}]) \tag{5}$$

where $[\cdot, \cdot]$ denotes concatenation.

### 3.4. VCM

While CAM and STAM refine attention maps in the convolutional blocks, VCM aims at capturing attention value in the classification stage. VCM extracts attention value before classification to improve the network's non-linearity and recalibrate activations by emphasizing informative activations and suppressing not informative ones. As VCM is arranged in the late stage, it has a simple design, which consists of a fully connected layer to obtain an attention map and a sigmoid activation for normalization.

The structure of VCM is illustrated in Fig 4. VCM can be regarded as a trainable classification optimizer. Given an activation vector $M$ from the fully connected layer, it first collects the attention value of $M$ through the fully connected layer without any pooling operation. After that, it generates the weight vector by normalizing the attention value to [0,1]. At last, the input activations $M$ are element-wise multiplied by the attention map for recalibration. The motivation behind VCM is two-fold. First, the fully connected layer is leveraged to reweight the strengths of different activations among the network

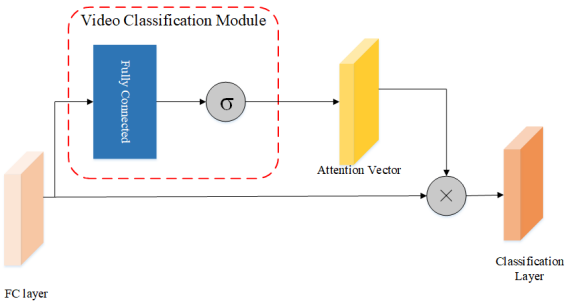


Figure 4: **The overview of Video Classification Module.** The VCM obtains the attention vector thorugh an extra fully connected and sigmoid function $\sigma$. Note that the attention vector has the same size as the input activation. And then the activations are reweighted by element-wise multiplication $\otimes$.

fully connected layer. Second, the sigmoid activation is set to improve non-linearity. The overview of the process is shown as:

$$\begin{cases} M' = F_{vc}(M) \otimes M \\ F_{vc}(M) = \sigma W M, F_{vc}(M) \in [0, 1] \end{cases} \tag{6}$$

where $W$ denotes the fully connected layer.

### 3.5. Summary

In this section, we clarify AARM and its submodules CAM, STAM and VCM. CAM and STAM are deployed after the convolutional layer in each convolutional block. Both of them obtain attention maps from convolutional feature maps while CAM refines channel-wise attention and STAM collects pixel-wise attention. VCM is deployed after the fully connected layer to obtain activation attention maps to recalibrate action for better classification. Besides, AARM has four hyperparameters: $r$, $\alpha$, *threshold* and *pick_rate*. The $r$ controls the reduction ratio in CAM which is set to 16, and the $\alpha$ indicates the weight of average pooling of mixed pooling method in CAM and STAM. And the *threshold* and *pick_rate* control the size and frequency of dropout layer in STAM. The analysis of them except $r$ are covered in details in the next section.

## 4. Experiments

To evaluate the proposed AARM, we conduct experiments on two action recognition datasets UCF101 and HMDB51, with ablation studies to verify the effectiveness of each component. Additionally, we implement well-known state-of-the-art methods using PyTorch (Ketkar, 2017) for better apple-to-apple comparisons. We provide the results of both methods and their AARM plugged architectures.

### 4.1. Experiment setting

UCF101 (Soomro et al., 2012) and HMDB51 (Kuehne et al., 2011) are two well-known action recognition benchmark datasets. The UCF101 dataset published in 2012 consists of 101 categories with 13,320 video clips collected from YouTube. The training set contains 9537 train samples and 3783 test samples, and the test set contains 3783 samples. The HMDB51 dataset published in 2011 is known as a challenging one containing 6766 video clips divided into 51 classes. Both of them providing three training/testing splits are trained by leave-one-out.

Unless stated ontherwise, we perform our AARM as follows. We use a Resnet pretrained from the ImageNet dataset as the backbone network and finetune it on the target dataset. Note that our modules are only trained during finetune phase to validate the extensibility. CAM and STAM are plugged in each convolutional block of the CNN and the output of them is the input of the next layer. Note that the dropout layer in STAM is activated in the training phase and deactivated in the testing phase. VCM is plugged after the fully connected layer and the output is used for classification. We use Top-1 classification accuracy as the metric in testing stage.

All the experiments are implemented on PyTorch with NVIDIA GeForce GTX 1080Ti GPU. We use SGD to finetune the models. We finetune our model 200 iterations for RGB branch and 250 iterations for Flow branch. The input size is croped to $224 \times 224$. The mini-batch size is set to 50, the momentum is set to 0.9 and the learning rate is set to 0.001. In comparison experiment section, all the settings follow the original approaches.

### 4.2. Ablation studies

In this subsection, to clarify each setting of the proposed AARM fairly, we utilize two-stream Resnet50 as a backbone network and measure their performances on UCF101 and HMDB51.

**Analysis of each submodule**. Table 1 shows the ablation study of AARM's submodules. For fair comparison, we leverage Resnet50 as the baseline network for the 2D network experiment and C3D(1net) for 3D. The 2D networks pretrained on ImageNet are tested on UCF101 and HMDB51, while the 3D networks trained from scratch are only tested on UCF101 as HDMB51 is too small to train. To get a closer look at effectiveness of our module, submodules are plugged into the networks one by one. Overall, each submodule makes enhancement on all networks and we can confirm the efficiency of our proposed AARM. Note that CAM and STAM are combined in a sequential way and dropout layer in STAM is not activated. The results demonstrate that CAM and STAM lead to the major performance improvement and VCM makes the minor contribution with fewer additional parameters.

**Analysis of** $\alpha$. The weight of the mixed pooling layer $\alpha$ is crucial to attention extraction. The higher $\alpha$ suggests that AARM focuses on global features, and vice versa. To clarify how $\alpha$ impacts on the performance, we conduct comparison on both UCF101 and HMDB51 with $\alpha$ from 0.1 to 0.9 and stride 0.2.

Table 2 shows the performance among *alpha*. It can be observed that $\alpha = 0.5$ achieve the best performance, while too high or too low $\alpha$ value contributes nearly nothing to performance, which implies that paying the same attention on both global features and

Table 1: **Comparison of each component on split1 of UCF101 and HMDB51.**

| Architecture | UCF101 | HMDB51 |
|---|---|---|
| Two-Stream(Resnet50) | 90.8% | 65.4% |
| Two-Stream+CAM | 91.4% | 66.0% |
| Two-Stream+CAM+STAM | 92.1% | 66.4% |
| Two-Stream+CAM+STAM+VCM | 92.4% | 66.9% |
| C3D(1net) | 44.0% | - |
| C3D+CAM | 44.3% | - |
| C3D+CAM+STAM | 45.0% | - |
| C3D+CAM+STAM+VCM | 45.4% | - |

Table 2: **Performance among aplha on UCF101 and HMDB51.** AARM can achieve the best performance while $\alpha = 0.5$.

| dataset \ $\alpha$ | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|---|---|---|---|---|---|
| UCF101 | 92.1% | 92.7% | 93.4% | 92.5% | 92.3% |
| HMDB51 | 66.5% | 67.1% | 67.6% | 66.9% | 66.5% |

silent one is proper for action recognition rather than only deploy avg pooling or max pooling.

**Analysis of dropout layer**. We further investigate the effect of the pick rate $\chi$ and the threshold $\eta$ in STAM's dropout layer on accuracy and thus activate dropout layer in STAM. First, we fix $\eta$ to 90% to find the best $\chi$ value for the dropout layer. According to Table 3, the result reports that the best performance can be achieved when the $\chi$ is 25%. Furthermore, we observe that when the highlighted map deactivated ($\chi$=0), the performance decreases lighty, implying that the highlight mechanism make a boost actually. However, we also find that the performance degrades when the highlighted map is applied more frequent than 25%. As a result, we can confirm that over emphasis on the most discriminative part leads to overfitting and regrading highlight mechanism as an auxiliary method is the best strategy .

Table 3: **Performance among $\chi$ on UCF101 and HMDB51.**

| dataset \ $\chi$(%) | 100 | 75 | 50 | 25 | 0 |
|---|---|---|---|---|---|
| UCF101 | 88.1% | 91.2% | 93.0% | 93.4% | 92.9% |
| HMDB51 | 63.6% | 65.7% | 67.0% | 67.6% | 67.1% |

Next, we explore the optimal value of $\eta$ in a similar manner and Table 4 reports the results. We can see that 90% can make the highest accuracy. And too small highlighted map ($\eta = 95\%$) causes accuracy decreases, suggesting that the network may not capture enough features for classification although they are discriminative. Meanwhile, large highlighted

map makes tiny contribution as it almost includes the whole discriminative region. We conclude that 90% keeps the balance between saliency and the map size.

In summary, the dropout plays an auxiliary role in STAM, which is activated only during the training phase. Note that both two hyperparameters are only the optimal setting for Resnet and the performance can be further improved on other architectures when the better setting is investigated.

Table 4: **Performance among $\eta$ on UCF101 and HMDB51.**

| $\eta(\%)$ / dataset | 95 | 90 | 85 | 80 | 70 |
|---|---|---|---|---|---|
| UCF101 | 92.9% | 93.4% | 93.2% | 93.1% | 92.9% |
| HMDB51 | 67.2% | 67.6% | 67.3% | 67.2% | 67.1% |

**Analysis of different CNNs**. We further analyze the generalization of our proposed AARM. To this end, we construct several network architectures including two-stream networks and 3D networks before and after our AARM. The experimental results are showed in Table 5. We have three observations: 1) The network with AARM shows higher performance than the original one. 2) Some architectures with AARM yield better performance than deeper original ones with fewer parameters. 3) Our AARM shows clear ability to boost the existing CNN architectures on action recognition task. We believe that AARM is suitable for action recognition and able to help deep architectures jump out of overfitting in some way by the novel attention mechanism.

Table 5: **Performance of differnet CNN architecture before and after AARM on UCF101 and HMDB51.**

| Architecture | UCF101 | | HMDB51 | |
|---|---|---|---|---|
| | origin | with AARM | origin | with AARM |
| 3DResnet18(scratch) (Hara et al., 2018) | 42.4% | 44.8% | 17.1% | 19.2 % |
| VGG16 | 86.9% | 89.1% | 58.4% | 60.9% |
| Resnet18 | 87.6% | 90.4% | 61.7% | 63.2% |
| Resnet34 | 89.9% | 92.3% | 63.9% | 66.2% |
| Resnet50 | 90.8% | 93.4% | 65.4% | 67.6% |
| Resnet101 | 91.6% | 93.6% | 66.1% | 67.9% |
| Resnet152 | 92.6% | 94.1% | 66.6% | 68.3% |

### 4.3. Comparison with attention modules

In this section, to further evaluate the effectiveness and efficiency of our proposed AARM, we compare it with other well-known attention modules SE module and CBAM module on action recognition task. For fair comparison, we leverage two-stream Resnet50 as baseline architecture. Both top-1 accuracy and number of parameters are taken into consideration.

Table 6 compares the number of parameters and performance, realtive to the baseline and other attention modules. Overall, all the cases obtain lower performance than our

Table 6: **Comparison of attention modules on split1 of UCF101 and HMDB51.** Experiment result shows that Resnet50 with AARM can yield the best top-1 accuracy of all attention modules, where RGB represents the RGB branch and OF represents the optical flow branch.

| Architecture | Parameters | UCF101 | HMDB51 |
|---|---|---|---|
| Res50 | 23.71M(RGB)+23.76M(OF) | 90.8% | 65.4% |
| Res50+SE | 26.22M(RGB)+26.27M(OF) | 91.3% | 65.7% |
| Res50+CBAM | 26.23M(RGB)+26.28M(OF) | 91.7% | 66.3% |
| Res50+AARM | 26.25M(RGB)+26.3M(OF) | 93.4% | 67.6% |

method and thus we can confirm the effectiveness and efficiency our presented AARM. In details, AARM achieves the best top-1 accuracy 92.4% on UCF101 and 66.9% on HMDB51. On the other hand, on UCF101 dataset, the SE module obtains a 0.5% increasement as well as the CBAM module makes a 0.9% accuracy gain by leveraging max pooling. Moreover, by comparing parameters of different modules, AARM has nearly the same parameters as the others, especially including the novel VCM. The observation reveals that STAM actually save parameters yet preserve enhancement. In brief, our AARM is effective than other modules with bearable computational cost growth.

### 4.4. Comparison with the state-of-the-art

We finally compare our proposed AARM with various state-of-the-art action recognition methods including the attention-based method: TSN (Wang et al., 2016), Hidden Two-stream (Zhu et al., 2018), ISPAN (Du et al., 2018) and ISTA (Meng et al., 2019). We reimplement some methods and plug AARM to evaluate its extensibility. Meanwhile, we leverage two-stream Resnet50 as the backbone architecture to construct AARMNet and use list the result based on TSN method.

In Table 7, we list the recent state-of-the-art and comparable methods. Our two-stream "AARM+" methods outerperform their original methods by about 2% on both UCF101 and HDMB51, implying that AARM is of wide applicability. In addition, our TSN+AARMNet achieves 95.8% on UCF101 and 70.2% on HMDB51. Compared with other attention-base method ISTA and ISPAN, our method obtain competitive performance. Moreover, we also observe that our method gain an improvement on C3D architecture, which shows the suitability of AARM for both two-stream networks methods and 3D CNN methods.

## 5. Conclusion

In this paper, we have introduced a novel module AARM for action recognition through attention of convolution and activation. Also, we propose the highlight and dropout layer in STAM to further obtain attention in space and time, and the parallel fusion strategy for convolutional attention module to tackle the influence of order. Meanwhile, AARM can be theoretically plugged into any existing CNNs to obtain better performance without extra modality or training samples. Furthermore, the data experiments of our proposed AARM

Table 7: **Performance on split 1 of UCF101 and HMDB51**. The results indicate that methods with AARM gain improvement about 2%.

| Architecture | UCF101 | HMDB51 |
|---|---|---|
| iDT+FV (Wang and Schmid, 2013) | 85.9% | 57.2% |
| Two-Stream(VGG16) (Simonyan and Zisserman, 2014) | 86.9% | 58.4% |
| C3D (Tran et al., 2015) | 85.2% | - |
| TSN (Wang et al., 2016) | 94.0% | 68.5% |
| PA3D (Yan et al., 2019) | - | 55.3% |
| Hidden Two-stream(TSN) (Zhu et al., 2018) | 93.2% | 66.8% |
| Coarse-to-fine(Motion) (Ji et al., 2019) | 93.6% | 69.3% |
| MRST-T (Wu et al., 2019) | 92.2% | 68.9% |
| **Attention-base methods** | | |
| ISTA (Meng et al., 2019) | 87.1% | 53.1% |
| ISPAN (Du et al., 2018) | 94.8% | 64.6% |
| Two-Stream(VGG16)+AARM | 89.1% | 60.9% |
| Hidden Two-stream+AARM | 95.3% | 68.7% |
| ST-Res+AARM | 95.4% | 68.3% |
| C3D+AARM | 86.9% | - |
| **AARMNet (Ours)** | **93.4%** | **67.6%** |
| **TSN+ AARMNet (Ours)** | **96.1%** | **70.4%** |

has been verified to further improve deep networks' performance on UCF101 and HMDB51. It is clear that the AARM can provide a new aspect for action recognition by applying attention recalibration mechanism. In the future, we will be dedicated to improving the efficiency of the attention modules.

## Acknowledgments

This paper is partly supported by Guangzhou Science and Technology Project with No.202002030273, and by National Natural Science Foundation of China (No. 61672546 and No. 61573385)

## References

Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

Maurizio Corbetta and Gordon L Shulman. Control of goal-directed and stimulus-driven attention in the brain. *Nature reviews neuroscience*, 3(3):201–215, 2002.

Ali Diba, Vivek Sharma, and Luc Van Gool. Deep temporal linear encoding networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2329–2338, 2017.

Yang Du, Chunfeng Yuan, Bing Li, Lili Zhao, Yangxi Li, and Weiming Hu. Interaction-aware spatio-temporal pyramid attention networks for action classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 373–389, 2018.

Kensho Hara, Hirokatsu Kataoka, and Yutaka 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*, pages 6546–6555, 2018.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

Yanting Hu, Jie Li, Yuanfei Huang, and Xinbo Gao. Channel-wise and spatial feature modulation network for single image super-resolution. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

Yanli Ji, Yue Zhan, Yang Yang, Xing Xu, Fumin Shen, and Heng Tao Shen. A context knowledge map guided coarse-to-fine action recognition. *IEEE Transactions on Image Processing*, 2019.

Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2000–2009, 2019.

Nikhil Ketkar. Introduction to pytorch. In *Deep learning with python*, pages 195–208. Springer, 2017.

Jun-Hyuk Kim, Jun-Ho Choi, Manri Cheon, and Jong-Seok Lee. Ram: Residual attention module for single image super-resolution. *arXiv preprint arXiv:1811.12043*, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563. IEEE, 2011.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lili Meng, Bo Zhao, Bo Chang, Gao Huang, Wei Sun, Frederick Tung, and Leonid Sigal. Interpretable spatio-temporal attention for video action recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

Jongchan Park, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Bam: Bottleneck attention module. *arXiv preprint arXiv:1807.06514*, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram E Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4597–4605, 2015.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5552–5561, 2019.

Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE international conference on computer vision*, pages 3551–3558, 2013.

Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

Haoze Wu, Jiawei Liu, Zheng-Jun Zha, Zhenzhong Chen, and Xiaoyan Sun. Mutually reinforced spatio-temporal convolutional tube for human action recognition. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 968–974, 2019.

An Yan, Yali Wang, Zhifeng Li, and Yu Qiao. Pa3d: Pose-action 3d machine for video recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7922–7931, 2019.

Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 286–301, 2018.

Yi Zhu, Zhenzhong Lan, Shawn Newsam, and Alexander Hauptmann. Hidden two-stream convolutional networks for action recognition. In *Asian Conference on Computer Vision*, pages 363–378. Springer, 2018.