Two-Stream Video Classification with Cross-Modality Attention

Lu Chi, Guiyu Tian, Yadong Mu*
Peking University
{chilu,wavey,myd}@pku.edu.cn

Qi Tian
Huawei Noah's Ark Lab
tian.qi1@huawei.com

Abstract

Fusing multi-modality information is known to be able to effectively bring significant improvement in video classification. However, the most popular method up to now is still simply fusing each stream's prediction scores at the last stage. A valid question is whether there exists a more effective method to fuse information cross modality. With the development of attention mechanism in natural language processing, there emerge many successful applications of attention in the field of computer vision. In this paper, we propose a cross-modality attention operation, which can obtain information from other modality in a more effective way than two-stream. Correspondingly we implement a compatible block named CMA block, which is a wrapper of our proposed attention operation. CMA can be plugged into many existing architectures. In the experiments, we comprehensively compare our method with two-stream and non-local models widely used in video classification. All experiments clearly demonstrate strong performance superiority by our proposed method. We also analyze the advantages of the CMA block by visualizing the attention map, which intuitively shows how the block helps the final prediction.

1. Introduction

In recent years, thanks to the emergence of massive video datasets [1, 14], applications of deep learning in video classification have witnessed a rapid development. However, there is still a considerable improvement space towards human-level video understanding. The state-of-the-art video classification methods are mainly based on convolutional neural networks. Despite tremendous progress has been recently made in designing highly discriminative network architectures, there still remain many open research problems. This research essentially concerns the following two problems:

Firstly, an insufficiently explore problem in video understanding is a more powerful way to capture the dynamic information or motions in videos. As one of the main differentiators between videos and images, dynamic information is regarded to be indispensable for effective video classification. For example, it is a difficult task even for a human being to discriminate from different kinds of dances (*e.g.*, *salsa dancing*, *tango dancing* and *dancing macarena*) by only having a glimpse at a single frame. A large number of widely-known video semantic categories, such as dancing and other sports, can be faithfully classified when we can extract sufficient motion-related information like moving trajectories.

Secondly, subtle details are key for recognizing some video categories or actions. The literature still lacks some in-depth analysis on effectively attending to those discriminative video details. Attention plays an important role in the field of natural language processing and image recognition. But it is still a nascent research topic in video action recognition. By grasping subtle details, humans can easily distinguish many classes. Considering the action *sword fighting* or *playing cricket*, only a single frame is enough as long as you can find a sword or a cricket. Generally, video motions attract more human attention and are likely to be related to key clues. For example, for the two actions *making a sandwich* and *making pizza*, their key objects sandwich or pizza are both around the moving hands. In this situation, motion can help attention.

Motivated by these observations, we propose a cross-modality attention operation, which can make full use of both static and motion information. Unlike the classic two-stream framework [28] that fuses information from two modalities in a late stage, we fuse the information in a more hierarchical and effective way.

Our proposed cross-modality attention operation devises such an attention mechanism that it encourages one modality absorbs most complementary information from other modalities. In contrast to the recently-proposed non-local operation [38], the proposed cross-modality attention can pay attention to other modalities rather than being constrained in the same modality. When two modalities under investigation are identical, our proposed method boils down to the non-local operation. Another key trait is that atten-

* is the corresponding author.

tions are computed in a global manner. Specifically, spiritually alike the non-local operation, our proposed method computes the response as a weighted sum of the features of the other modality at all positions.

There are three main advantages of using the proposed cross-modality attention operation, sketched briefly as below: 1) It can effectively fuse the information between two or more modalities; 2) It can capture long-range dependencies by globally investigating the feature maps; 3) It can be wrapped as a highly compatible block that can be inserted into almost all existing neural networks and frameworks.

The rest of this paper is organized as: we first review related work in Section 2 and detail the novel cross-modality attention operation / network design in Sections 3 and 4. Section 5 shows the experiments and detailed analysis of our module.

2. Related Work

With the significant development of deep learning in image recognition [15, 29, 9, 10], a large number of active researches have emerged in the field of video classification. Karpathy *et al*. [13] contributed significant breakthrough in the video classification task. Their major contribution is 3-D convolutional neural networks trained on the Sports-1M data, which far exceeds traditional methods [35, 21] in terms of top-1 or top-5 classification accuracies. This seminal work demonstrates the power of temporal information in video-related tasks.

Optical flow fields are known as conceptually simple and empirically effective when attempting to capture the temporal information. A variety of approaches have been developed to utilize optical flow in video classification. A large body of existing works [28, 39, 3, 31] has re-iteratively found that feeding the optical flow fields into a deep model can bring comparable performance with the RGB stream in video classification. After properly fused via late-stage fusion, one can accomplish a performance better than either stream. Recent endeavor along this research thrust includes direct mapping two adjacent frames to the optical flow field [18, 11]. Researchers have also investigated using deep neural networks for computing optical flows, which can be expedited by modern GPU hardware. However, the major obstacle stems from the lack of high-quality training data. To mitigate the data scarcity, some train an optical flow model from synthesized datasets [18], or predict the label of videos in an end-to-end way for improving the accuracy[26, 19]. In addition, the optimization ideas in the traditional methods are integrated into the design of the neural networks. Fan *et al*. [7] unfold the optimization iterations in TV-L1 [41] as neural layers, and Sun *et al*. [31] propose neural networks to learn the representations orthogonal to the optical flow. We would point out that, though tremendous efforts have been noticed in computing optical flows, litter has been done to explore how to effectively using optical flow in video classification.

Optical flow can be regarded as an explicit way to utilize motion information to video classification. More recent research is pursuing other alternatives that rely on deep neural networks is automatically distill spatio-temporal video information. Typical examples include inflating 2D convolution into 3D convolution [12, 32, 33]. One of key weaknesses of these models are the gigantic parameters used for defining high-dimensional convolutions etc. Using pre-trained models is a popularly-verified effective strategy for easing the model training in many tasks [24, 4, 5], such as transferring deep models pre-trained on ImageNet to 3D CNN. A naive solution is to duplicate the parameters of the 2D filters T times along the time dimension, and rescale all parameters by dividing by T [3]. This ensures a same response from the convolution filters. To reduce parameter number in 3D CNNs, some works factorize 3D convolutional filters into separate spatial and temporal components and strike a compromise in accuracy and efficacy [23, 33, 40]. Other relevant works mix 3D convolution with 2D convolution in a neural network [36, 40, 44]. Despite the empirical success on indicative video benchmarks, 3D CNNs are far from reaching the point of fully acquiring the motion information and replacing the optical flow. Fusing with the optical flow stream is still an effective practice [3, 36]. In fact, we can regard 3D CNNs as a general tool that acquires relation among adjacent frames. It can be fed with either RGB frames or other modalities (*e.g*., optical flow).

For complex video objects, other information also provide complementary information, including audio [17], human pose [6, 16], and semantic body part [43] etc. Learning how to efficiently integrate multi-modality information is an emerging research direction. Existing researches, such as pooling at different stages [13, 20] or modeling long-range temporal structure using LSTM [39], mainly concern fusing in the temporal dimension. There are rarely relevant studies about the fusion of different modalities [39]. To date, the mainstream method is still the two-stream method [28]. Our primary goal is to design a network structure that is more effective than two-stream and meanwhile achieves higher precision.

Attention networks have been originally popularized in natural language processing [2, 34], used for comprehension and abstractive summarization etc. Recent years have observed a quick spread in computer vision [27, 8, 42]. Xie *et al* [40] place a feature gating module after some convolutional layers to weight the features in each channel in an adaptive, data-dependent way. Long *et al* [17] propose attention clusters, which aggregates local features to generate a valid global representations for video classification. Non-local networks [38] can weight all information (including space and time) by adopting a mechanism similar to self-attention. Our motivating observation is the lack of cross modality attention block, which works globally as non-local block but can make full use of cross-modality information. Importantly, this block shall be compatibly inserted into most existing network structures including the classic two-stream inputs.

3. Cross Modality Attention

In this section, we give detailed description of the proposed Cross Modality Attention(CMA) operation and its implementation.

3.1. Formulation

Our proposed Cross Modality Attention(CMA) operation can be precisely described in the Q-K-V language, namely matching a query from one modality with a set of key-value pairs from the other modality and thereby extracting most critical cross-modality information. Following the notations in [34], we define the generic CMA operation as:

$$\mathrm{CMA}(Q_1, K_2, V_2) = softmax\left(\frac{Q_1 K_2^T}{\sqrt{d_k}}\right) V_2, \quad (1)$$

where the index 1 or 2 represents different modality. Q is the set of queries, K is a matrix of the memory keys and V contains memory values. All Q, K and V are of feature dimension d_k.

Here we give a concrete instance of the CMA operation in neural networks. Given a typical two-stream data (RGB + flow), a CMA operation can be written as:

$$\mathbf{z}_i = \frac{1}{\mathcal{C}(\mathbf{x}, \mathbf{y})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{y}_j) v(\mathbf{y}_j) \quad (2)$$

$$f(\mathbf{x}_i, \mathbf{y}_j) = e^{q(\mathbf{x}_i) k(\mathbf{y}_j)^T} / \sqrt{d_k} \quad (3)$$

$$\mathcal{C}(\mathbf{x}, \mathbf{y}) = \sum_{\forall j} f(\mathbf{x}_i, \mathbf{y}_j), \quad (4)$$

where $\mathbf{x}$ is from the feature maps of specific stage of the RGB branch, such as the output of res$_4$ in ResNet [9]. $\mathbf{y}$ is from the feature maps in the flow branch. $\mathbf{z}_i$ denotes the output of the CMA operation. i and j are both indices of feature maps (can be in space, time, or spacetime). q, k and v are linear embeddings which map $\mathbf{x}$ or $\mathbf{y}$ to queries, keys and values of d_k dimensions respectively. The function f can be flexibly defined, with many instantiations discussed in [38]. For simplicity, we choose the embedded Gaussian version in this paper.

The non-local operation [38] is essentially self-attention and only pays attention to intra-modality. In comparison, our proposed CMA is cross-modal. Moreover, the non-local operation can be regarded as a special case of CMA when K, Q and V are all from the same modality.

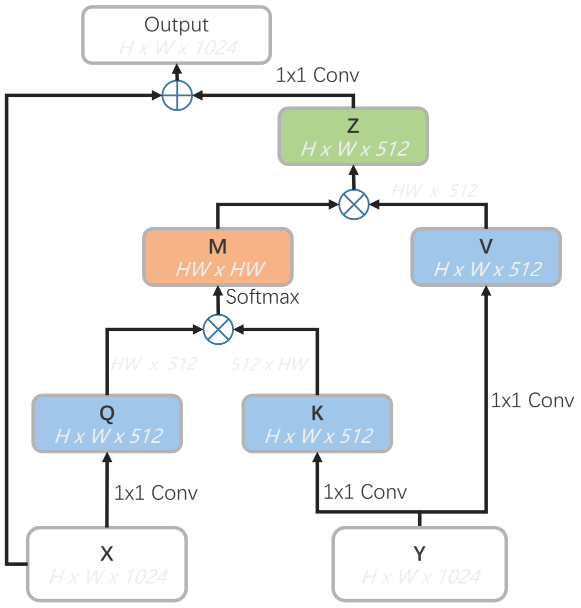

Figure 1: An example of **CMA block**. We show the shape of feature maps at each stage, such as $H \times W \times 1024$, where 1024 is the number of channels. Let X be the feature maps of the RGB branch and Y be the feature maps of flow branch. The number of channels is halved via 1×1 convolutions. Reshaping or transposing is performed whenever needed. "$\bigotimes$" denotes matrix multiplication, and "$\bigoplus$" denotes element-wise sum.

3.2. CMA Block

A CMA block is a wrapper of the CMA operation that can be inserted into many existing neural networks, which is defined as:

$$out_i = W_{out}\mathbf{z}_i + \mathbf{x}_i, \quad (5)$$

where $\mathbf{x}_i$ and $\mathbf{z}_i$ are given in Eqn. (2). W_{out} defines a linear embedding that can be implemented by convolution operation.

Figure 1 presents an example of the CMA block, where Q comes from the RGB branch and V, K come from the flow branch. This allows the RGB branch to attend over all positions in the flow branch at a specific stage. As a result, it can get more valuable information selectively from the flow branch which may be weak or even missing in itself. A CMA block can be added into any location of deep neural networks, since it can be fed with input of any shape and ensure a same-shaped output. This flexibility allows us to fuse richer hierarchical features between different modalities. To make the CMA block more compatible, we add a residual connection "$+\mathbf{x}_i$" [9]. This guarantees a non-worse accuracy with the CMA block by some simple means (*e.g.*, zeroing W_{out}).

Implementation of CMA Blocks: We implement functions q, k, and v as 1×1 convolutions in space or $1 \times 1 \times 1$

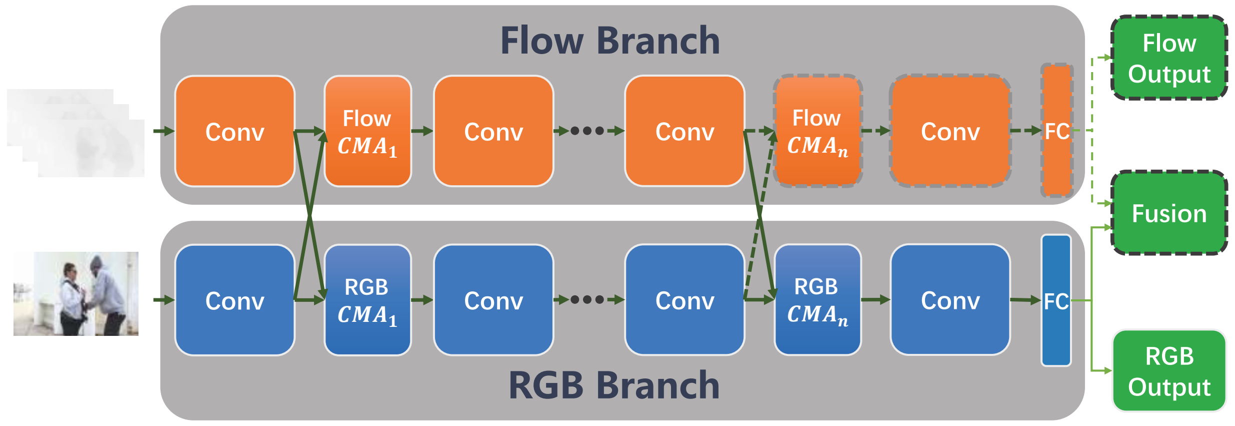

Figure 2: **An overview of the video classification model.** This model contains both RGB branch and Flow branch. In each branch, we insert n CMA blocks, which play an important role in transmitting information between different branches. There are three outputs in this model. Operations in dotted box are not essential since practically we can only use the output of the RGB branch for prediction.

convolution in space-time, denoted as conv_q, conv_k and conv_v respectively. To reduce the computational complexity and GPU memory consumption, we let conv_q reduce the number of channels to be half of that in $\mathbf{x}$, and conv_k and conv_v have same number of channels as conv_q. Note that we also insert a spatial max-pooling layer with stride 2 before k and v, further simplifying the computation. Inspired by [38], a batch normalization layer is added after W_{out} and has the scale parameters initialized to be zeros, which sets the entire CMA block as an identity mapping.

4. Network Architecture

This section elaborates on our proposed video classification model. We first introduce two branches in our model and how the CMA blocks are inserted. Then, we depict the whole network architecture and finally describe the details of training strategy and implementation.

4.1. Two Branches for video classification

As shown in Figure 2, our model contains two branches, including the RGB branch and Flow branch. As mentioned in [28], the RGB branch carries visual information about scenes and objects in the video, while the Flow branch conveys the motion. All the information from both branches are crucial for video classification. In two-stream [28], they simply average the scores of the two branches to make the final prediction.

We add several CMA blocks at some intermediate stages of each branch, obtaining information from the other branch. Compared with the two-stream method, this fuses two modalities much earlier and hierarchically. There are three classification scores in our model. The first two scores are from the RGB branch and the Flow branch respectively, and the last one is a weighted summation of RGB / Flow branches. Empirical investigation deferred to Section 5 shows that any of these three scores can make an excellent prediction. In fact, in the scenario of highly-budgeted parameters, one can just use the scores of the RGB branch without much loss of performance.

Implementation: Both branches adopt ResNet-50 [9] as base network. Considering the limited GPU memory and precise spatial information, we add 5 CMA blocks in res_3 and res_4 to every other residual block, which is also similar to the setting in non-local neural networks [38]. The RGB branch takes only one RGB frame as input, while the Flow branch stacks five consecutive optical flow fields as input. The RGB branch can be directly initialized from the ResNet weights pre-trained on ImageNet [25]. Since the number of input channels of the Flow branch is different from that of the models pre-trained on ImageNet, we initialize the weights of the first convolution by replicating the means of the pre-trained weights across channels. The CMA blocks are initialized via the same scheme in [9]. We zero the scale parameters of the last BN layer as previously mentioned in Section 3.2.

4.2. TSN Framework

Temporal Segment Networks (TSN) has been proved to be powerful in modeling long-range temporal structure [37, 36, 31]. We also incorporate this effective albeit simple framework. Given a video, we divide it into K segments, ensuring the duration of each segment equal. For each segment, a snippet (1 RGB frame for the RGB branch and 5 consecutive optical flow fields for the Flow branch) is randomly sampled. We average the scores produced by each segment to get the final video-level score, namely

$$\mathbf{G} = \frac{1}{K}\sum_{i}^{K}\mathbf{G_i}, \tag{6}$$

where K is the number of segments and $\mathbf{G_i}$ is the score of one specific snippet.

The overall loss function can be defined as:

$$\mathcal{L}(y, \mathbf{G}) = -\sum_{c=1}^{C} y_c (G_c - \log \sum_{j=1}^{C} e^{G_j}), \tag{7}$$

where C is the number of video classes and y_c is the ground-truth label concerning class c. G_c are the scores of the same class on all snippets.

4.3. Training Strategy

Since the Flow models converge much slowly than RGB models [37], we firstly train the flow branch on Kinetics data [14]. After that, considering the limited GPU memory, we train the CMA Model in an iterative way between two branches. Thinking that the CMA blocks is initially an identity mapping and the Flow branch has been trained on the kinetics, the Flow branch can provide more reliable information to the RGB branch before the iterative training stage. Therefore, we train the RGB branch in $iter_1, iter_3, iter_5$... and train the Flow branch in $iter_2, iter_4, iter_6, ...$ When training the RGB branch, its parameters are optimized according to the loss of the current branch and we freeze all the layers in the Flow branch, including CMA blocks of the Flow branch. Similar treatment for training the Flow branch. The total number of epochs at each iteration is set to 30.

4.4. Implementation Details

Input: The video frames are scaled to size 256×256. We choose the TVL1 optical flow algorithm [41] to extract optical flow for the Flow branch, based on the GPU version from the OpenCV toolbox. The pixel values of optical flow are truncated to the range $[-20, 20]$, and then re-scaled between -1 and 1. The input size of two branches are both 224×224, cropped from the video frames or optical flow fields. The RGB branch takes only one frame ($frame_t$) as the input and the Flow branch reads a stack of consecutive optical flow fields ([of_t,of_{t+1},of_{t+2},of_{t+3},of_{t+4}]). In other words, the input shapes of two branches are $N \times 224 \times 224 \times 3$ and $N \times 224 \times 224 \times 10$ respectively, where N is the batch size and the last dimension represents the number of channels. It's important to note that the RGB frame is corresponding to the first optical flow field in the temporal dimension, and all RGB frame / optical flows are spatially aligned. For data augmentation, we use random horizontal flipping, random cropping and scale jittering [37]. And the number of segments is set to 3.

Training: We use a standard cross-entropy loss and mini-batch stochastic gradient descent algorithm to optimize the network parameters, where the batch size is 128. We train the model with BN enabled, which is the same to [38]. To make the statistics of each BN layer more accurate, we use the synchronized batch normalization [22]. The learning rate is initialized as 0.01 and get reduced by a factor of 10 when the accuracy is stuck in some plateau. At the beginning of each iteration, we reset the learning rate to the initial value. The dropout ratio is 0.7 and the weight decay is 5^{-4}, which are introduced to reduce over-fitting.

Testing: During test time we use ten-croppings and flip four corners and the center of the frame or optical flow filed as [15]. The number of segments is set to 25 and the temporal spacing between each segment is equal. We average the scores across all the samples and crops of them to get the final video-level score. For the fusion score, we firstly get the frame-level scores via weighted sum and then average all the scores to get the video-level score. We will provide an empirical study on the fusion weights in Section 5.2.

5. Experiment

We evaluate the proposed methods and perform ablation studies on two popular datasets, **UCF101** [30] and **Kinetics** [14]. For clarity, let CMA_$iter_i$ be the model trained after ith iterations. We add the suffix "-R", "-F", "-S" for the RGB / Flow streams or two-stream respectively.

5.1. Dataset

UCF-101 [30] consists of 101 action classes and over 13-K clips. All the videos are downloaded from YouTube, and all of them are recorded in unconstrained environments, including various lighting conditions, partial occlusion, low quality frames etc.

Kinetics [14] is a large-scale trimmed video dataset which contains more than 300-K video clips in total, and each clip has a duration of around 10 seconds. The dataset covers 400 human-centric classes and each class has at least 400 video clips. For unknown reasons, there are some invalid urls and we are unable to crawl some of the videos. We get 232,679 videos for training and 19,169 for validation. We skip processing the testing set since their labels are not provided.

5.2. Investigation of Fusion Weights

To get the fusion score, two-stream [28] averages the scores from two modalities and [37] gives more credits to the Flow modality by setting its weight as 1.5 and that of RGB modality as 1. But our proposed model contains two branches which are interdependent, consequently, training one branch inevitably have an effect on the other one. In this situation, exploring suitable fusion weights is necessary. Figure 3 shows the top-1 accuracy with different fusion weights. We use the two-stream as a baseline whose base model is the same as ours (ResNet50). The two-stream model can achieve a higher accuracy when the weight of the two branches is almost the same. But for the CMA

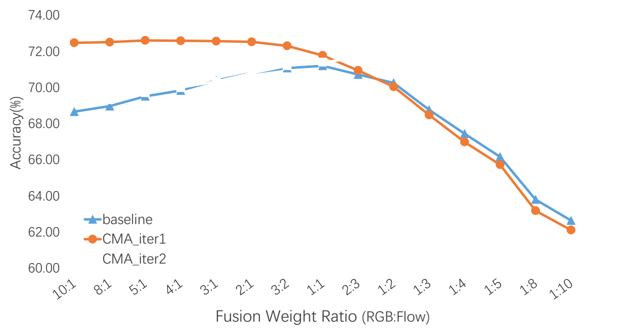

Figure 3: **The top-1 accuracies with different fusion weights.** The CMA models perform better than the two-stream when we give higher weight to the more reliable branch.

models, at the first training iteration, we just train the RGB branch with the Flow branch fixed, so the RGB branch performs better than the Flow branch. In other words, the RGB branch is more reliable. Giving the RGB branch more weight will make the final accuracy higher, but too much weight will make the other branch almost completely ignored. At the second iteration, we should similarly give more weight to the Flow branch. From Figure 3 one can see that the fusion accuracy of CMA models is always higher than the baseline, as long as we give more weight to the more reliable branch.

Based on the above analysis, we give the equal weights to the two branches in two-stream, identical to [28], and set the weights of the RGB / Flow branches as $5:1$ at $iter_1, iter_3, ...$ and $1:5$ at $iter_2, iter_4, ...$. For all the following experiments we adopt such setting.

5.3. Performance at Each Iteration

Iteration	RGB		Flow		two-stream	
	top-1	top-5	top-1	top-5	top-1	top-5
0	67.73	87.94	55.73	79.04	71.21	89.92
1	72.17	**90.70**	55.73	79.04	**72.62**	**91.04**
2	68.45	88.54	**71.17**	**90.12**	71.55	90.24
3	**72.19**	90.63	69.81	89.41	72.55	90.82

Table 1: Accuracies at each iteration on the Kinetics dataset.

In Section 4.3, we introduce the iterative training strategy. Here let us study how many iterations we need for convergence. Table 1 lists the accuracy at different iterations. $iter_0$ represents the baseline that has the two branches trained independently. The fusion accuracy is equal to two-stream [1]. After $iter_1$, the RGB branch has exceeded the two-stream, and the Flow branch keeps the same as the baseline because we have not trained on it at this iteration and the CMA blocks in this branch are now just an identity mapping. Additionally, the accuracy of fusion is much higher than others. In order to achieve higher accuracy for the Flow branch, we train the Flow branch with the RGB branch freezed at the second iteration. As expected, the accuracy of the Flow branch is improved and can be comparable to the two-stream. But the performance of the RGB branch drops due to that the distribution of the feature maps of the Flow branch has changed, which can affect the RGB branch through the CMA blocks. After $iter_3$, the accuracy of the RGB branch returns to the relative high level while the Flow branch degrades slightly. The fusion score doesn't be improved any more. It is thus observed that the first iteration is almost sufficient for our models.

[1]Although we name the baseline as $iter_0$, we don't initialize the CMA model with the parameters in $iter_0$. The train strategy keep the same as described in Section 4.3

5.4. Analysis and Visualization

model	params	top-1	top-5
ResNet50-R	1×	67.73	87.94
two-stream	2×	71.21	89.92
CMA_$iter_1$-R	**1.8×**	**72.17**	**90.70**

Table 2: CMA model *vs* two-stream in terms of parameter number and accuracy. The number of parameters are relative to the ResNet50 baseline.

Table 2 compares our method with two-stream in terms of a few key factors, including number of parameters and final accuracy. CMA_$iter_1$-R is more accurate than two-stream, though fewer parameters are used. That validates that our CMA model is more effective than two-stream for fusing.

groundtruth	confusing category
gargling	trimming or shaving beard
tying tie	tying bow tie
yawning	baby waking up
cracking neck	massaging back
kissing	hugging
rock scissors paper	shaking hands
running on treadmill	waxing leg
water sliding	jumping into pool
sneezing	crying
breading or breadcrumbing	cooking chicken

Table 3: The top-10 confusing categories on which the CMA model achieves the largest gain compared with two-stream in Kinetics. The gain is the improved accuracy (%).

Figure 4 showed the top-20 most improved categories and compare between our CMA model / two-stream. We

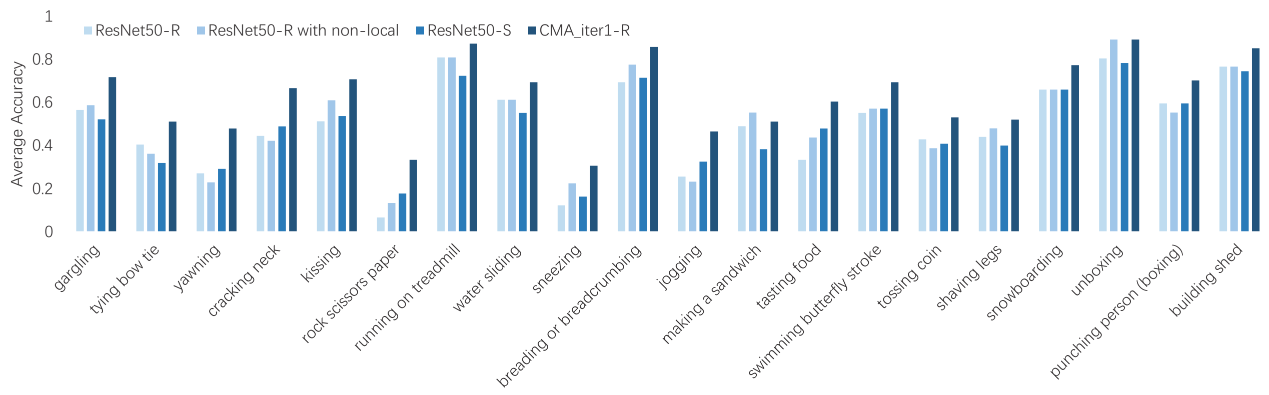

Figure 4: **Comparing top-20 most improved categories between the proposed CMA model and two-stream**.

also list the top-10 confusing categories in Table 3. Compared with two-stream, the proposed CMA model is more sensitive about the motion trajectories, such as *water sliding* and *jumping into pool*, although the background is similar. Due to the fact that almost all of the samples of *yawning* are about babies, it's very easy to confuse with *baby waking up*. But our model can improve the performance according to the different motion between two categories. Additionally, the CMA model can pay more attention to the moving objects, such as jaw or mouth, or the tools (cup or razor) held on the hand, while discriminating between *gargling* and *trimming or shaving beard*. We also visualize some attention maps in Figure 5.

5.5. Comparison with Non-local Blocks

model	non-local	modality	top-1	top-5
ResNet50	-	RGB	67.73	87.94
ResNet50	-	Flow	55.73	79.04
ResNet50	-	RGB + Flow	71.21	89.92
ResNet50	Yes	RGB	68.74	88.43
ResNet50	Yes	Flow	56.66	80.11
ResNet50	Yes	RGB + Flow	71.67	89.87
CMA_$iter_1$-R	-	RGB + Flow	72.17	90.70
CMA_$iter_1$-S	-	RGB + Flow	**72.62**	**91.04**
CMA_$iter_1$-R	Yes	RGB + Flow	72.27	90.76
CMA_$iter_1$-S	Yes	RGB + Flow	72.60	91.01

Table 4: Comparisons with non-local networks on Kinetics.

Non-local block [38] is also a kind of attention-based model which pays attention to the intra-modality features. It also shows good performance in video classification. In order to compare the performance and mutual influence between self-attention blocks and our proposed cross-modality attention blocks, we carry out some experiments, and the results are shown in Table 4. Following [38], we add five blocks to ResNet50 in res$_3$ and res$_4$, the same numbers and locations as that of the CMA blocks. To explore the influence between these two kinds of blocks, we conduct the experiments that adding CMA blocks just behind the non-local blocks. To ensure the comparisons more tractable, we only add the nonlocal blocks in the RGB branch, which implies that the Flow branch is the same to the Flow modality of the ResNet50 model.

From the results in Table 4, we find that non-local blocks can roughly improve top-1 accuracy by 1% in both RGB and Flow modalities of ResNet50 model. For our proposed model, even only the results of the RGB branch outperform the fusion results of ResNet50 with nonlocal blocks. More importantly, the non-local blocks seem unnecessary while using our CMA blocks, which shows that the CMA blocks can also play a role of non-local blocks while fusing different modalities. In other words, the CMA blocks can also capture global information.

5.6. 3D-CMA Blocks

model	# input frames	RGB	Flow	two-stream
P3D	12	70.98	63.80	73.91
P3D	16	71.50	66.20	74.62
CMA_$iter_1$	12	74.41	63.80	75.22
CMA_$iter_1$	16	**74.86**	**66.20**	**75.98**

Table 5: Performance of P3D and 3D-CMA models on Kinetics when varying the count of input frames. All models adopt ResNet-152 as the backbone, and the input of CMA blocks are all 3D.

To illustrate that the proposed CMA blocks can also be compatible with 3D convolutional neural networks and further improve its performance, we insert this operation into P3D ResNet [23]. We initialize the P3D network with the

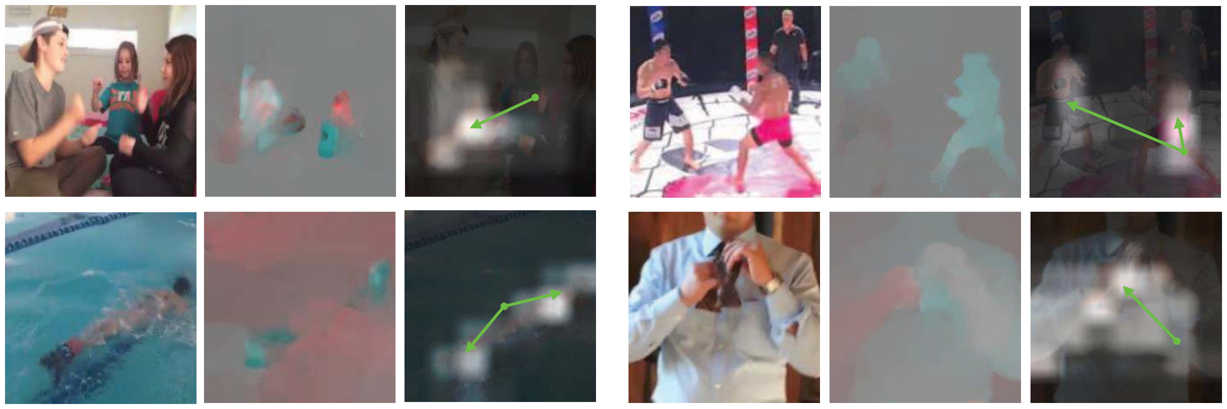

Figure 5: **Examples of the attention maps**. We train CMA_$iter_1$-R on Kinetics and visualize the attention maps of the last CMA block in res$_4$ since the last block is the most related to the final classification. These samples are taken from Kinetics randomly. Each set contains three images, including (from left to right) RGB frame, optical flow fields, and the attention map. In the attention map, we draw some arrows that start from the query location and point to the more interesting parts in the CMA block. We observe that the block can easily focus on moving objects, such as the moving hand in the top-left set and the swimming person in the bottom-left set. And as a result, the RGB branch can take important information from the Flow branch as much as possible within limited capacity. We also find that the CMA operation is global. Looking at the example in the top-right, the pixel on the right person can not only focus on the nearby region but also pay attention to the other boxer, which shows our CMA block can capture long-range dependencies. Moreover, not all the moving objects can attract the attention, only key information does. In the last example *tying bow tie*, the block pays more attention to the region around the hand although the whole upper body is moving, because that the object held in hands often has more impact on the prediction. And more attention maps can be found in supplemental material.

weights duplicated from the official caffemodel[2] and fine-tune it using data augmentation. For the CMA model, considering the limited GPU memory, we only add one CMA block after the last layer in res$_4$, and train the CMA block and all layers behind it. We train our CMA model with different numbers of input frames. Table 5 summarizes the experimental results. As seen, the CMA block can also bring an improvement for P3D compared with two-stream. Fusion with two branches can further improve the accuracy.

5.7. Transfer learning

We also conduct transfer learning experiments from Kinetics to UCF-101. We only fine-tune the last fc layer of our 2-D CMA model. Table 6 shows the results. We find that our model is somewhat easier to over-fit on the small dataset. Nonetheless, the proposed CMA_$iter_1$-S can outperform most of the state-of-the-art 2D models. It even approximates the performance of 3D models (*e.g.*, I3D with 64 RGB frames and 64 flows as its input) although only 2D convolutional network as base model is used.

model	use 3D-Conv	top-1
ECO [44]	Yes	94.8
ARTNet [36]	Yes	94.3
I3D [3]	Yes	98.0
Two-stream[28]	No	88.0
TSN [37]	No	94.0
Attention Cluster [17]	No	94.6
ResNet50-R	No	90.9
ResNet50-F	No	92.4
ResNet50-S	No	95.5
CMA_$iter_1$-R	No	95.3
CMA_$iter_1$-S	No	96.5

Table 6: Comparison with state-of-the-art on the UCF-101. The first set is the results reported by other papers, and the second set is our results of transfer learning.

6. Conclusion and Future work

We have shown that cross-modality operations can significantly improve the performance in video classification. The proposed CMA block can be compatibly inserted to most existing neural networks. It proves very effective to fuse information between different modalities. Our future works include extending to more sophisticated deep models, and evaluating the CMA operation among more modalities beyond the RGB and Flow branches.

Acknowledgments: This work is supported by Beijing Municipal Commission of Science and Technology under Grant Z181100008918005, National Natural Science Foundation of China (NSFC) under Grant 61772037.

[2]https://github.com/ZhaofanQiu/pseudo-3d-residual-networks

References

[1] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *CoRR*, abs/1609.08675, 2016.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[3] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *CVPR*, pages 4724–4733, 2017.

[4] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun. Cascaded pyramid network for multi-person pose estimation. *CoRR*, abs/1711.07319, 2017.

[5] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[6] W. Du, Y. Wang, and Y. Qiao. Rpan: An end-to-end recurrent pose-attention network for action recognition in videos. In *ICCV*, volume 2, 2017.

[7] L. Fan, W. Huang, S. E. Chuang Gan, B. Gong, and J. Huang. End-to-end learning of motion representation for video understanding. In *CVPR*, pages 6016–6025, 2018.

[8] J. Fu, J. Liu, H. Tian, Z. Fang, and H. Lu. Dual attention network for scene segmentation. *CoRR*, abs/1809.02983, 2018.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[10] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, pages 2261–2269, 2017.

[11] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, pages 1647–1655, 2017.

[12] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1):221–231, 2013.

[13] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. Li. Large-scale video classification with convolutional neural networks. In *CVPR*, pages 1725–1732, 2014.

[14] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[16] M. Liu and J. Yuan. Recognizing human actions as the evolution of pose estimation maps. In *CVPR*, pages 1159–1168, 2018.

[17] X. Long, C. Gan, G. de Melo, J. Wu, X. Liu, and S. Wen. Attention clusters: Purely attention based local feature integration for video classification. In *CVPR*, pages 7834–7843, 2018.

[18] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, pages 4040–4048, 2016.

[19] J. Y. Ng, J. Choi, J. Neumann, and L. S. Davis. Actionflownet: Learning motion representation for action recognition. In *WACV*, pages 1616–1624, 2018.

[20] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, pages 4694–4702, 2015.

[21] J. C. Niebles, C. Chen, and F. Li. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*, pages 392–405, 2010.

[22] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun. Megdet: A large mini-batch object detector. *CoRR*, abs/1711.07240, 2017.

[23] Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, pages 5534–5542, 2017.

[24] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149, 2017.

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[26] L. Sevilla-Lara, Y. Liao, F. Güney, V. Jampani, A. Geiger, and M. J. Black. On the integration of optical flow and action recognition. *CoRR*, abs/1712.08416, 2017.

[27] S. Sharma, R. Kiros, and R. Salakhutdinov. Action recognition using visual attention. *CoRR*, abs/1511.04119, 2015.

[28] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, pages 568–576, 2014.

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[30] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012.

[31] S. Sun, Z. Kuang, L. Sheng, W. Ouyang, and W. Zhang. Optical flow guided feature: A fast and robust motion representation for video action recognition. In *CVPR*, 2018.

[32] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, pages 4489–4497, 2015.

[33] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, pages 6450–6459, 2018.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, pages 6000–6010, 2017.

[35] H. Wang, M. M. Ullah, A. Kläser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, pages 1–11, 2009.

[36] L. Wang, W. Li, W. Li, and L. V. Gool. Appearance-and-relation networks for video classification. In *CVPR*, pages 1430–1439, 2018.

[37] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, pages 20–36, 2016.

[38] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. *CVPR*, 2018.

[39] Z. Wu, X. Wang, Y. Jiang, H. Ye, and X. Xue. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *ACM Multimedia*, pages 461–470, 2015.

[40] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning for video understanding. *CoRR*, abs/1712.04851, 2017.

[41] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-L^1 optical flow. In *DAGM*, pages 214–223, 2007.

[42] H. Zhang, I. J. Goodfellow, D. N. Metaxas, and A. Odena. Self-attention generative adversarial networks. *CoRR*, abs/1805.08318, 2018.

[43] Z. Zhao, H. Ma, and S. You. Single image action recognition using semantic body part actions. In *ICCV*, pages 3411–3419, 2017.

[44] M. Zolfaghari, K. Singh, and T. Brox. ECO: efficient convolutional network for online video understanding. In *ECCV*, 2018.