Asynchronous Interaction Aggregation for Action Detection

Jiajun Tang, Jin Xia, Xinzhi Mu, Bo Pang, and Cewu Lu(✉)

Shanghai Jiao Tong University, Shanghai, China
{yelantingfeng,draconids,pangbo,lucewu}@sjtu.edu.cn, ga.xiajin@gmail.com

Abstract. Understanding interaction is an essential part of video action detection. We propose the Asynchronous Interaction Aggregation network (AIA) that leverages different interactions to boost action detection. There are two key designs in it: one is the Interaction Aggregation structure (IA) adopting a uniform paradigm to model and integrate multiple types of interaction; the other is the Asynchronous Memory Update algorithm (AMU) that enables us to achieve better performance by modeling very long-term interaction dynamically without huge computation cost. We provide empirical evidence to show that our network can gain notable accuracy from the integrative interactions and is easy to train end-to-end. Our method reports the new state-of-the-art performance on AVA dataset, with *3.7 mAP* gain (12.6% relative improvement) on validation split comparing to our strong baseline. The results on datasets UCF101-24 and EPIC-Kitchens further illustrate the effectiveness of our approach. Source code will be made public at: https://github.com/MVIG-SJTU/AlphAction.

Keywords: Action detection · Video understanding · Interaction · Memory

1 Introduction

The task of action detection (spatio-temporal action localization) aims at detecting and recognizing actions in space and time. As an essential task of video understanding, it has a variety of applications such as abnormal behavior detection and autonomous driving. On top of spatial representation and temporal features [3,10,21,27], the interaction relationships [13,29,39,47] are crucial for understanding actions. Take Fig. 1 for example. The appearance of the man, the

J. Tang and J. Xia—Both authors contributed equally to this work.
C. Lu is a member of Qing Yuan Research Institute and MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, China.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-58555-6_5) contains supplementary material, which is available to authorized users.

© Springer Nature Switzerland AG 2020
A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12360, pp. 71–87, 2020.
https://doi.org/10.1007/978-3-030-58555-6_5

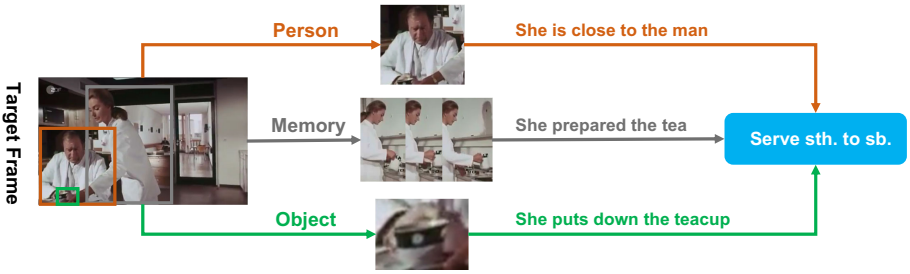

Fig. 1. Interaction Aggregation. In this target frame, we can tell that the women is serving tea to the man with following clues: (1) She is close to the man. (2) She puts down the tea cup before the man. (3) She prepared the tea a few seconds ago. These three clues correspond respectively the person-person, person-object and temporal interactions

tea cup as well as the previous movement of the woman help to predict the action of the woman. In this paper, we propose a new framework which emphasizes on the interactions for action detection.

Interactions can be briefly considered as the relationship between the target person and context. Many existing works try to explore interactions in videos, but there are two problems in the current methods: (1) Previous methods such as [13,15] focus on a single type of interaction (eg. person-object). They can only boost one specific kind of actions. Methods such as [46] intend to merge different interactions, but they model them separately. Information of one interaction can't contribute to another interaction modeling. How to find interactions correctly in video and use them for action detection remains challenging. (2) The long-term temporal interaction is important but hard to track. Methods which use temporal convolution [10,21,27] have very limited temporal reception due to the resource challenge. Methods such as [41] require a duplicated feature extracting pre-process which is not practical in reality.

In this work, we propose a new framework, the Asynchronous Interaction Aggregation network (AIA), who explores three kinds of interactions (person-person, person-object, and temporal interaction) that cover nearly all kinds of person-context interactions in the video. As a first try, AIA makes them work cooperatively in a hierarchical structure to capture higher level spatial-temporal features and more precise attentions. There are two main designs in our network: the Interaction Aggregation (IA) structure and the Asynchronous Memory Update (AMU) algorithm.

The former design, IA structure, explores and integrates all three types of interaction in a deep structure. More specifically, it consists of multiple elemental interaction blocks, of each enhances the target features with one type of interaction. These three types of interaction blocks are nested along the depth of IA structure. One block may use the result of previous interactions blocks. Thus, IA structure is able to model interactions precisely using information across different types.

Jointly training with long memory features is infeasible due to the large size of video data. The AMU algorithm is therefore proposed to estimate intractable features during training. We adopt a memory-like structure to store the spatial features and propose a series of write-read algorithm to update the content in memory: features extracted from target clips at each iteration are written to a memory pool and they can be retrieved in subsequent iterations to model temporal interaction. This effective strategy enables us to train the whole network in an end-to-end manner and the computational complexity doesn't increase linearly with the length of temporal memory features. In comparison to previous solution [41] that extracted features in advance, the AMU is much simpler and achieves better performance.

In summary, our key contributions are: (1) A deep IA structure that integrates a diversity of person-context interactions for robust action detection and (2) an AMU algorithm to estimate the memory features dynamically. We perform an extensive ablation study on the AVA [17] dataset for spatio-temporal action localization task. Our method shows a huge boost on performance, which yields the new state-of-the-art on both validation and test set. We also test our method on dataset UCF101-24 [32] and a segment level action recognition dataset EPIC-Kitchens [6]. Results further validate its generality.

2 Related Works

Video Classification. Various 3D CNN models [21,33,34,36] have been developed to handle video input. To leverage the huge image dataset, I3D [3] has been proposed to benefit from ImageNet[7] pre-training. In [4,8,27,35,44], the 3D kernels in above models are simulated by temporal filters and spatial filters which can significantly decrease the model size.

Previous two-stream methods [11,30] use optical flow to extract motion information, while recent work SlowFast [10] manages to do so using only RGB frames with different sample rates.

Spatio-temporal Action Detection. Action detection is more difficult than action classification because the model needs to not only predict the action labels but also localize the action in time and space. Most of the recent approaches [10,12,17,19,42] follow the object detection frameworks [14,28] by classifying the features generated by the detected bounding boxes. In contrast to our method, their results depend only on the cropped features. While all the other information is discarded and contributes nothing to the final prediction.

Attention Mechanism for Videos. The transformer [37] consists of several stacked self-attention layers and fully connected layers. Non-Local [38] concludes that the previous self-attention model can be viewed as a form of classical computer vision method of non-local means [2]. Hence a generic non-local block[38] is introduced. This structure enables models to compute the response by relating the features at different time or space, which makes the attention mechanism applicable for video-related tasks like action classification. The non-local block

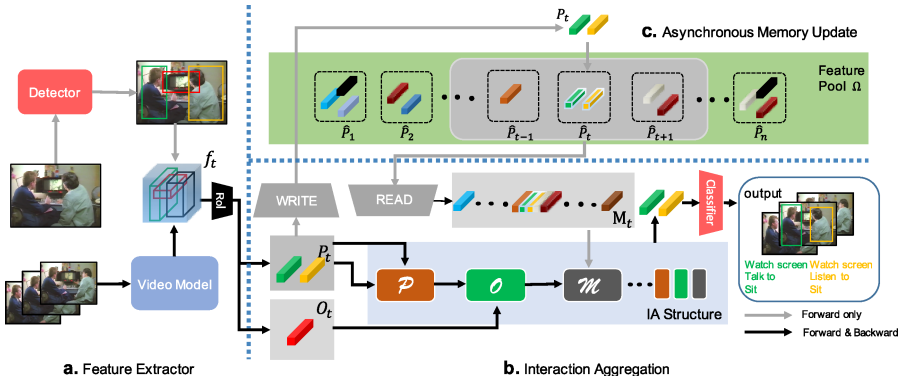

Fig. 2. Pipeline of the proposed AIA. **a**. We crop features of persons and objects from the extracted video features. **b**. Person features, object features and memory features from the feature pool Ω in **c** are fed to IA in order to integrate multiple interactions. The output of IA is passed to the final classifier for predictions. **c**. Our AMU algorithm reads memory features from feature pool and writes fresh person features to it

also plays an important role in [41] where the model references information from the long-term feature bank via a non-local feature bank operator.

3 Proposed Method

In this section, we will describe our method that localizes actions in space and time. Our approach aims at modeling and aggregating various interactions to achieve better action detection performance. In Sect. 3.1, we describe two important types of instance level features in short clips and the memory features in long videos. In Sect. 3.2, the Interaction Aggregation structure (IA) is explored to gather knowledge of interactions. In Sect. 3.3, we introduce the Asynchronous Memory Update algorithm (AMU) to alleviate the problem of heavy computation and memory consumption in temporal interaction modeling. The overall pipeline of our method is demonstrated in Fig. 2.

3.1 Instance Level and Temporal Memory Features

To model interactions in video, we need to find correctly what the queried person is interacted with. Previous works such as [38] calculate the interactions among all the pixels in feature map. Being computational expensive, these brute-force methods struggle to learn interactions among pixels due to the limited size of video dataset. Thus we go down to consider how to obtain concentrated interacted features. We observe that persons are often interacting with concrete objects and other persons. Therefore, we extract object and person embedding as the instance level features. In addition, video frames are usually highly correlated, thus we keep the long-term person features as the memory features.

Instance level features are cropped from the video features. Since computing the whole long video is impossible, we split it to consecutive short video clips $[v_1, v_2, \ldots, v_T]$. The d-dimensional features of the t^{th} clip v_t are extracted using a video backbone model: $f_t = \mathcal{F}(v_t, \phi_{\mathcal{F}})$ where $\phi_{\mathcal{F}}$ is the parameters.

A detector is applied on the middle frame of v_t to get person boxes and object boxes. Based on the detected bounding boxes, we apply RoIAlign [18] to crop the person and object features out from extracted features f_t. The person and object features in v_t are denoted respectively as P_t and O_t.

One clip is only a short session and misses the temporal global semantics. In order to model the temporal interaction, we keep tracks of memory features. The memory features consist of person features in consecutive clips: $M_t = [P_{t-L}, \ldots, P_t, \ldots, P_{t+L}]$, where $(2L + 1)$ is the size of clip-wise reception field. In practice, a certain number of persons are sampled from each neighbor clip.

The three features above have semantic meaning and contain concentrated information to recognize actions. With these three features, we are now able to model semantic interactions explicitly.

3.2 Interaction Modeling and Aggregation

How do we leverage these extracted features? For a target person, there are multiple detected objects and persons. The main challenge is how to correctly pay more attention to the objects or the persons that the target person is interacted with. In this section, we introduce first our Interaction Block that can adaptively model each type of interactions in a uniform structure. Then we describe our Interaction Aggregation (IA) structure that aggregates multiple interactions.

Overview. Given different human P_t, object O_t and memory features M_t, the proposed IA structure outputs action features $A_t = \mathcal{E}(P_t, O_t, M_t, \phi_{\mathcal{E}})$, where $\phi_{\mathcal{E}}$ denotes the parameters in the IA structure. A_t is then passed to the final classifier for final predictions.

The hierarchical IA structure consists of multiple interaction blocks. Each of them is tailored for a single type of interactions. The interaction blocks are deep nested with other blocks to efficiently integrate different interactions for higher level features and more precise attentions.

Interaction Block. The structure of interaction block is adapted from Transformer Block originally proposed in [37] whose specific design basically follows [38,41]. Briefly speaking, one of the two inputs is used as the query and the other is mapped to key and value. Through the dot-product attention, which is the output of the softmax layer in Fig. 3 **a**, the block is able to select value features that are highly activated to the query features and merge them to enhance the query features. There are three types of interaction blocks in our design, which are P-Block, O-Block and M-Block.

–*P-Block:* P-Block models person-person interaction in the same clip. It is helpful for recognizing actions like listening and talking. Since the query input

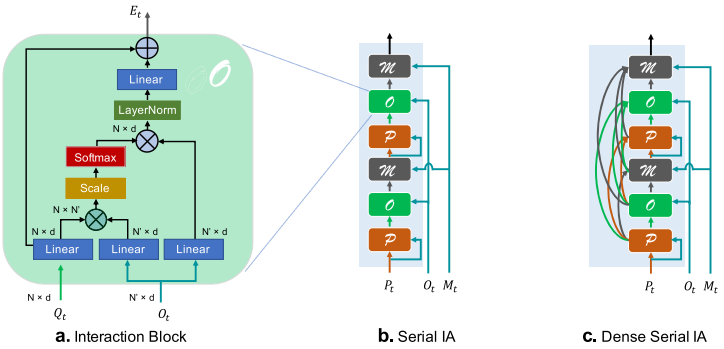

Fig. 3. Interaction Block and IA structure. **a**. The O-Block: the query input is the feature of the target person and the key/value input is the feature of objects. The P-Block and M-Block are similar. **b**. Serial IA. **c**. Dense Serial IA

is already the person features or the enhanced person features, we take the key/value input the same as the query input.
–*O-Block:* In O-Block, we aim to distill person-object interactions such as pushing and carrying an object. Our key/value input is the detected object features O_t. In the case where detected objects are too many, we sample based on detection scores. Figure 3**a** is an illustration of O-Block.
–*M-Block:* Some actions have strong logical connections along the temporal dimension like opening and closing. We model this type of interaction as temporal interactions. To operate this type, we take memory features M_t as key/value input of an M-Block.

Interaction Aggregation Structure. The Interaction Blocks extract three types of interaction. We now propose two IA structures to integrate these different interactions. The proposed IA structures are the naive parallel IA, the serial IA and the dense serial IA. For clarity, we use $\mathcal{P}$, $\mathcal{O}$, and $\mathcal{M}$ to represent the P-Block, O-Block, and M-Block respectively.

–*Parallel IA:* A naive approach is to model different interactions separately and merge them at last. As displayed in Fig. 4**a**, each branch follows similar structure to [13] that treats one type of interactions without the knowledge of other interactions. We argue that the parallel structure struggles to find interaction precisely. We illustrate the attention of the last P-Block in Fig. 4**c** by displaying the output of the softmax layer for different persons. As we can see, the target person is apparently watching and listening to the man in red. However, the P-block pays similar attention to two men.
–*Serial IA:* The knowledge across different interactions is helpful for recognizing interactions. We propose the serial IA to aggregate different types of interactions. As shown in Fig. 3**b**, different types of interaction blocks are stacked in sequence. The queried features are enhanced in one interaction block and then passed to an interaction block of a different type. Figure 4**f**

Fig. 4. We visualize attention by displaying the output of the softmax layer in P-Block. The original output contains the attention to zero padding person. We remove those meaningless attention and normalize the rest attention to 1

and 4**g** demonstrate the advantage of serial IA: The first P-block can not differ the importance of the man in left and the man in middle. After gaining knowledge from O-block and M-block, the second P-block is able to pay more attention to man in left who is talking to the target person. Comparing to the attention in parallel IA (Fig. 4**c**), our serial IA is better in finding interactions.

–*Dense Serial IA:* In above structures, the connections between interaction blocks are totally manually designed and the input of an interaction block is simply the output of another one. We expect the model to further learn which interaction features to take by itself. With this in mind, we propose the Dense Serial IA extension. In Dense Serial IA, each interaction block takes all the outputs of previous blocks and aggregates them using a learnable weight. Formally, the query of the i^{th} block can be represent as

$$Q_{t,i} = \sum_{j \in \mathbf{C}} W_j \odot E_{t,j}, \tag{1}$$

where $\odot$ denotes the element-wise multiplication, $\mathbf{C}$ is the set of indices of previous blocks, W_j is a learnable d-dimenional vector normalized with a Softmax function among $\mathbf{C}$, $E_{t,j}$ is the enhanced output features from the j^{th} block. Dense Serial IA is illustrated in Fig. 3**c**.

3.3 Asynchronous Memory Update Algorithm

Long-term memory features can provide useful temporal semantics to aid recognizing actions. Imagine a scene where a person opens the bottle cap, drinks water, and finally closes the cap, it could be hard to detect opening and closing

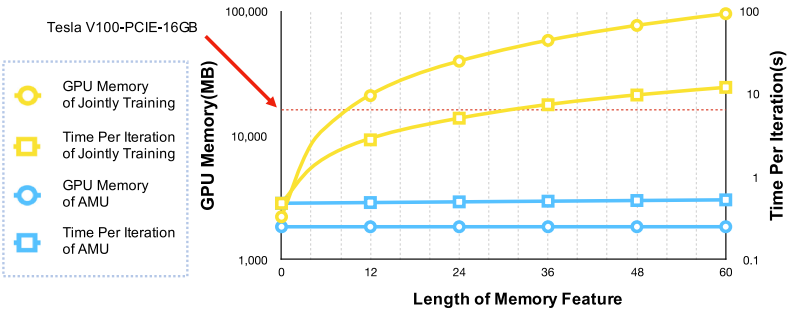

Fig. 5. Joint training with memory features is restricted by limited hardware resource. In this minor experiment, we take a 32-frame video clip with 256×340 resolution as input. The backbone is ResNet-50. During joint training (yellow line), rapidly growing GPU memory and computation time restricted the length of memory features to be very small value (8 in this experiment). With larger input or deeper backbone, this problem will be more serious. Our method (cyan line) doesn't have such problem. (Color figure online)

with subtle movements. But knowing the context of drinking water, things get much easier.

Resource Challenge. To capture more temporal information, we hope our M_t can gather features from enough number of clips, however, using more clips will increase the computation and memory consumption dramatically. Depicted with Fig. 5, when jointly training, the memory usage and computation consumption increase rapidly as the temporal length of M_t grows. To train on one target person, we must propagate forward and backward $(2L + 1)$ video clips at one time, which consumes much more time, and even worse, cannot make full use of enough long-term information due to limited GPU memory.

Insight. In the previous work [41], they pre-train another duplicated backbone to extract memory features to avoid this problem. However, this method makes use of frozen memory features, whose representation power can not be enhanced as model training goes. We expect the memory features can be updated dynamically and benefit from the improvement from parameter update in training process. Therefore, we propose the asynchronous memory update method which can generate effective dynamic long-term memory features and make the training process more lightweight. The details of training process with this algorithm are presented in Algorithm 1.

A naive design could be: pass forward all clips to get memory features and propagate current clip backward to calculate the gradients. This method alleviates the memory issue but is still slow in the training speed. We could also try to utilize the memory features like in Transformer-XL [5], but this requires training along the sequence direction and is thus unable to access future information.

Inspired by [40], our algorithm is composed of a memory component, the memory pool Ω and two basic operations, *READ* and *WRITE*. The memory

Algorithm 1. Training with asynchronous memory update

Input: Video dataset $\mathbf{V} = \{v^{(1)}, v^{(2)}, \ldots, v^{(|\mathbf{V}|)}\}$ with $v^{(i)} = [v_1^{(i)}, v_2^{(i)}, \ldots, v_{T_i}^{(i)}]$;
The whole network $\mathcal{N}$, with its parameter $\phi_{\mathcal{N}}$;
Output: Optimized network $\mathcal{N}$ with $\phi_{\mathcal{N}}$ for inference.
// Initialization :
$\Omega = \{(\hat{P}_t^{(i)} \leftarrow \text{zero vectors}, \delta_t^{(i)} \leftarrow 0) \mid \forall t, i\}$.
$err \leftarrow \infty$.
// Training Process:
for $iter = 1$ to $iter_{max}$ **do**
  Sample a video clip $v_t^{(i)}$ from dataset $\mathbf{V}$.
  for $t' = t - L$ to $t + L$ **do**
    if $t' \neq t$ **then**
      READ $\hat{P}_{t'}^{(i)}$ and $\delta_{t'}^{(i)}$ from memory pool Ω.
      $w_{t'}^{(i)} = \min\{err/\delta_{t'}^{(i)}, \delta_{t'}^{(i)}/err\}$.
      Impose penalty: $\hat{P}_{t'}^{(i)} \leftarrow w_{t'}^{(i)} \hat{P}_{t'}^{(i)}$.
    end if
  end for
  Extract $P_t^{(i)}$ and $O_t^{(i)}$ with the backbone in $\mathcal{N}$.
  Estimated memory features: $\hat{M}_t^{(i)} \leftarrow [\hat{P}_{t-L}^{(i)}, \ldots, \hat{P}_{t-1}^{(i)}, P_t^{(i)}, \hat{P}_{t+1}^{(i)}, \ldots, \hat{P}_{t+L}^{(i)}]$.
  Forward $(P_t^{(i)}, O_t^{(i)}, M_t^{(i)})$ with the head in $\mathcal{N}$ and backward to optimize $\phi_{\mathcal{N}}$.
  Update err as the output of current loss function.
  WRITE $\hat{P}_t^{(i)} \leftarrow P_t^{(i)}, \delta_t^{(i)} \leftarrow err$ back to Ω.
end for
return $\mathcal{N}$, $\phi_{\mathcal{N}}$

pool Ω records memory features. Each feature $\hat{P}_t^{(i)}$ in this pool is an estimated value and tagged with a loss value $\delta_t^{(i)}$. This loss value $\delta_t^{(i)}$ logs the convergence state of the whole network. Two basic operations are invoked at each iteration of training:

–*READ:* At the beginning of each iteration, given a video clip $v_t^{(i)}$ from the i^{th} video, estimated memory features around the target clip are read from the memory pool Ω, which are $[\hat{P}_{t-L}^{(i)}, \ldots, \hat{P}_{t-1}^{(i)}]$ and $[\hat{P}_{t+1}^{(i)}, \ldots, \hat{P}_{t+L}^{(i)}]$ specifically.
–*WRITE:* At the end of each iteration, personal features for the target clip $P_t^{(i)}$ are written back to the memory pool Ω as estimated memory features $\hat{P}_t^{(i)}$, tagged with current loss value.
–*Reweighting:* The features we *READ* are written at different training steps. Therefore, some early written features are extracted from the model whose parameters are much different from current ones. Therefore, we impact a penalty factor $w_{t'}^{(i)}$ to discard badly estimated features. We design a simple yet effective way to compute such penalty factor by using loss tag. The difference between the loss tag $\delta_{t'}^{(i)}$ and current loss value is expressed as,

$$w_{t'}^{(i)} = \min\{err/\delta_{t'}^{(i)}, \delta_{t'}^{(i)}/err\}, \tag{2}$$

which should be very close to 1 when the difference is small. As the network converges, the estimated features in the memory pool are expected to be closer and closer to the precise features and $w_{t'}^{(i)}$ approaches to 1.

As shown in Fig. 5, the consumption of our algorithm has no obvious increase in both GPU memory and computation as the length of memory features grows, and thus we can use long enough memory features on current common devices. With dynamic updating, the asynchronous memory features can be better exploited than frozen ones.

4 Experiments on AVA

The Atomic Visual Actions (AVA) [17] dataset is built for spatio-temporal action localization. In this dataset, each person is annotated with a bounding box and multiple action labels at 1 FPS. There are 80 atomic action classes which cover pose actions, person-person interactions and person-object interactions. This dataset contains 235 training movie videos and 64 validation movie videos.

Since our method is originally designed for spatio-temporal action detection, we use AVA dataset as the main benchmark to conduct detailed ablation experiments. The performances are evaluated with official metric frame level mean average precision(mAP) at spatial IoU ≥ 0.5 and only the top 60 most common action classes are used for evaluation, according to [17].

4.1 Implementation Details

Instance Detector. We apply Faster R-CNN [28] framework to detect persons and objects on the key frames of each clip. A model with ResNeXt-101-FPN [23,43] backbone from maskrcnn-benchmark [26] is adopted for object detection. It is firstly pre-trained on ImageNet [7] and then fine-tuned on MSCOCO [25] dataset. For human detection, we further fine-tune the model on AVA for higher detection precision.

Backbone. Our method can be easily applied to any kind of 3D CNN backbone. We select state-of-the-art backbone SlowFast [10] network with ResNet-50 structure as our baseline model. Basically following the recipe in [10], our backbone is pre-trained on Kinetics-700 [3] dataset for action classification task. This pre-trained backbone produces 66.34% top-1 and 86.66% top-5 accuracy on the Kinetics-700 validation set.

Training and Inference. Initialized from Kinetics pre-trained weights, we then fine-tune the whole model with focal loss [24] on AVA dataset. The inputs of our network are 32 RGB frames, sampled from a 64-frame raw clip with one frame interval. Clips are scaled such that the shortest side becomes 256, and then fed into the fully convolution backbone. We use only the ground-truth human boxes for training and the randomly jitter them for data augmentation. For the object boxes, we set the detection threshold to 0.5 in order to have higher recall. During inference, detected human boxes with a confidence score larger than 0.8 are used. We set $L = 30$ for memory features in our experiments. We train our network using the SGD algorithm with batch size 64 on 16 GPU (4 clips per device). BatchNorm(BN) [20] statistics are set frozen. We train for 27.5k iterations with

Table 1. Ablation Experiments. We use a ResNet-50 SlowFast backbone to perform our ablation study. Models are trained on the AVA (v2.2) training set and evaluated on the validation set. The evaluation metric mAP is shown in %

(a) **3 Interactions**

$\mathcal{P}$	$\mathcal{O}$	$\mathcal{M}$	mAP
			26.54
✓	✓		28.04
✓		✓	28.86
	✓	✓	28.92
✓	✓	✓	**29.26**

(b) **Num of I-Blocks**

blocks	mAP
$1 \times \{\mathcal{P}, \mathcal{M}, \mathcal{O}\}$	29.26
$2 \times \{\mathcal{P}, \mathcal{M}, \mathcal{O}\}$	**29.64**
$3 \times \{\mathcal{P}, \mathcal{M}, \mathcal{O}\}$	29.61

(c) **Interaction Order**

order	mAP	order	mAP
$\mathcal{M} \to \mathcal{O} \to \mathcal{P}$	29.48	$\mathcal{M} \to \mathcal{P} \to \mathcal{O}$	29.46
$\mathcal{O} \to \mathcal{P} \to \mathcal{M}$	29.51	$\mathcal{O} \to \mathcal{M} \to \mathcal{P}$	29.53
$\mathcal{P} \to \mathcal{M} \to \mathcal{O}$	29.44	$\mathcal{P} \to \mathcal{O} \to \mathcal{M}$	**29.64**

(d) **IA Structure**

structure	mAP
Parallel	28.85
Serial	29.64
Dense Serial	**29.80**

(e) **Asynchronous Memory Update**

model	params	FLOPs	mAP
Baseline	1.00×	1.00×	26.54
LFB(w/o AMU)	2.18×	2.12×	27.02
LFB(w/ AMU)	1.18×	1.12×	28.57
IA(w/o AMU)	2.35×	2.15×	28.07
IA(w/ AMU)	1.35×	1.15×	29.64

(f) **Compare to NL**

model	mAP
Baseline	26.54
+NL	26.85
+IA(w/o $\mathcal{M}$)	**28.23**

base learning rate 0.004 and the learning rate is reduced by a factor 10 at 17.5k and 22.5k iteration. A linear warm-up [16] scheduler is applied for the first 2k iterations.

4.2 Ablation Experiments

Three Interactions. We first study the importance of three kinds of interactions. For each interaction type, we use at most one block in the experiment. These blocks are then stacked in serial. To evaluate the importance of person-object interaction, we remove the O-Block in the structure. Other interactions are evaluated in the same way. Table 1a compares the model performance, where used interaction types are marked with "✓". A backbone baseline without any interaction is also listed in this table. Overall we observe that removing any of these three type interactions results in a significant performance decrease, which confirms that all these three interactions are important for action detection.

Number of Interaction Blocks. We then experiment with different settings for the number of interaction blocks in our IA structure. The interaction blocks are nested in serial structure in this experiment. In Table 1b, $N \times \{\mathcal{P}, \mathcal{M}, \mathcal{O}\}$ denotes N blocks are used for each interaction type, with the total number as $3N$. We find that with the setting $N = 2$ our method can achieve the best performance, so we use this as our default configuration.

Interaction Order. In our serial IA, different type of interactions are alternately integrated in sequential. We investigate effect of different interaction order design in Table 1c. As shown in this experiment, the performance with different

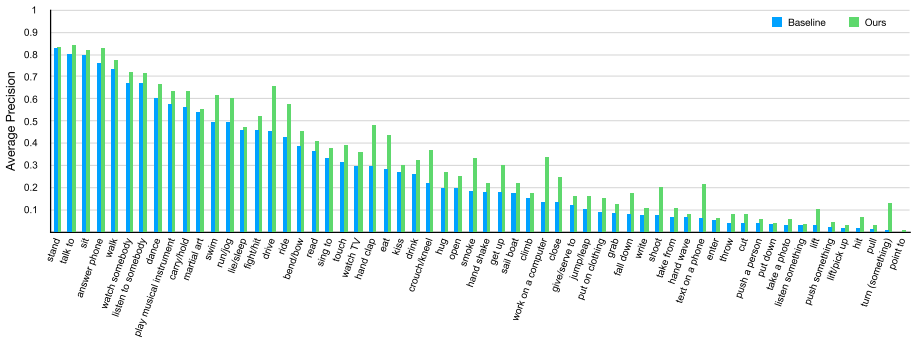

Fig. 6. Per category results comparison on the validation set of AVA v2.2

order are quite similar, we thus choose the slightly better one $\mathcal{P} \rightarrow \mathcal{O} \rightarrow \mathcal{M}$ as our default setting.

Interaction Aggregation Structure. We analyze different IA structure in this part. Parallel IA, serial IA and the dense serial IA extension are compared in Table 1d. As we expect, the parallel IA performs much worse than serial structure. With dense connections between blocks, our model is able to learn more knowledge of interactions, which further boosts the performance.

Asynchronous Memory Update. In the previous work LFB [41], the memory features are extracted with another backbone, which is frozen during training. In this experiment we compare our asynchronous memory features with the frozen ones. For fair comparison, we re-implement LFB with SlowFast backbone, and also apply our AMU algorithm to LFB. In Table 1d, we find that our asynchronous memory features can gain much better performance than the frozen method with nearly half of the parameters and computation cost. We argue that this is because our dynamic features can provide better representation.

Comparison to Non-local Attention. Finally we compare our interaction aggregation method with prior work non-local block [38] (NL). Following [10], we augment the backbone with a non-local branch, where attention is computed between the person features and global pooled features. Since there is no long-term features in this branch, we eliminate $\mathcal{M}$ in this experiment. In Table 1f, we see that our serial IA works significantly better than NL block. This confirms that our method can better learn to find potential interactions than NL block.

4.3 Main Results

Finally, we compare our results on AVA v2.1 and v2.2 with previous methods in Table 2. Our method surpasses all previous works on both versions.

The AVA v2.2 dataset, is the newer benchmark used in ActivityNet challenge 2019 [9]. On the validation set, our method reports a new state-of-the-art *33.11 mAP* with one single model, which outperforms the strong baseline SlowFast by

Table 2. Main results on AVA. Here, we display our best results with both ResNet50(R50) and ResNet101(R101). "*" indicates multi-scale testing. The input sizes are shown in frame number and sample rate. SlowFast R101 backbone models re-implemented in this work are also displayed as "ours" for comparison.

(a) **Comparison on AVA v2.1.**

model	input	pretrain	val
SlowFast [10]	32 × 2	K400	26.3
LFB [41]	32 × 2	K400	27.7
I3D [12]	64 × 1	K600	21.9
SlowFast+NL [10]	32 × 2	K600	28.2
SlowFast (ours)	32 × 2	K600	-
SlowFast (ours)	32 × 2	K700	28.1
AIA R50	32 × 2	K700	28.9
AIA R101	32 × 2	K700	**31.2**

(b) **Comparison on AVA v2.2.**

model	input	pretrain	val	test
SlowFast+NL [10]	32 × 2	K600	29.1	-
SlowFast+NL [10]	64 × 2	K600	29.4	-
SlowFast*, 7 ens. [10]	-	K600	-	34.25
SlowFast (ours)	32 × 2	K600	-	-
SlowFast (ours)	32 × 2	K700	29.3	-
AIA R50	32 × 2	K700	29.80	-
AIA R101	32 × 2	K700	32.26	-
AIA R101*	32 × 2	K700	**33.11**	**32.25**
AIA R101*, 3 ens.	-	K700	-	**34.42**

Table 3. Results on UCF101-24 Split1

Method	mAP	Method	mAP	Method	mAP	Method	mAP
ACT [22]	69.5	Gu *et al.* [17]	76.3	C2D (ours)	75.5	I3D (ours)	79.6
STEP [45]	75.0	Zhang *et al.* [46]	77.9	C2D+AIA	78.8	I3D+AIA	**81.7**

3.7 mAP. On the test split, we train our model on both training and validation splits and use a relative longer scheduler. With an ensemble of three models with different learning rates and aggregation structures, our method achieves better performance than the winning entry of AVA challenge 2019 (an ensemble with 7 SlowFast [10] networks). The per category results for our method and SlowFast baseline is illustrated in Fig. 6. We can observe the performance gain for each category, especially for those who contain interactions with video context.

As shown in Table 2, we pre-train the backbone model with a new larger Kinetics-700 for better performance. However, it is worth noting that we do not use non-local block in our backbone model and there are some other slight differences between our implementation and the official one [10]. As a result, our K700 backbone model has a similar performance to the official K600 one. That is to say, very most of the performance advantages benefit from our proposed method instead of the backbone.

5 Experiments on UCF101-24

UCF101-24 [32] is an action detection set with 24 action categories. We conduct experiments on the first split of this dataset following previous works and use the corrected annotations provided by Singh *et al.* [31].

We experiment two different backbone models, C2D and I3D. Both of them are pre-trained on the Kinetics-400 dataset. Other settings are basically the

Table 4. EPIC-Kitchens validation results

	Verbs		Nouns		Actions	
	top-1	top-5	top-1	top-5	top-1	top-5
Baradel [1]	40.9	–	–	–	–	–
LFB NL [41]	52.4	80.8	29.3	54.9	20.8	39.8
SlowFast (ours)	56.8	82.8	32.3	56.7	24.1	42.0
AIA-Parallel	57.6	83.9	36.3	63.0	26.4	47.4
AIA-Serial	59.2	84.2	**37.2**	**63.2**	**27.7**	**48.0**
AIA-Dense-Serial	**60.0**	**84.6**	**37.2**	62.1	27.1	47.8

same as AVA experiments. More implementation details are provided in Supplementary Material. Table 3 shows the result on UCF101-24 test split in terms of Frame-mAP with 0.5 IOU threshold. As we can see in the table, AIA achieves 3.3% and 2.1% improvement over two different backbones. Moreover, with a relative weak 2D backbone, our method still achieves very competitive results.

6 Experiments on EPIC-Kitchens

To demonstrate the generalizability of AIA, we evaluate our method on the segment level dataset EPIC-Kitchens [6]. In EPIC Kitchens, each segment is annotated with one verb and one noun. The action is defined by their combination.

For both verb model and noun model, we use the extracted segment features (global average pooling of f_t) as query input for IA structure. Hand features and object features are cropped and then fed into IA to model person-person and person-object interactions. For verb model, the memory features are the segment features. For noun model, the memory features are the object features extracted from object detector feature map, thus the AMU algorithm is only applied to the verb model. More details are available in Supplementary Material. From Table 4, we observe a significant gain for all three tasks. All the variants of AIA outperform the SlowFast baseline. Among them, the dense serial IA achieves the best performance for the verbs test, leading to 3.2% improvement on top-1 score. The serial IA results in 4.9% for the nouns test and 3.6% for the action test.

7 Conclusion

In this paper, we present the Asynchronous Interaction Aggregation network and its performance in action detection. Our method reports the new start-of-the-art on AVA dataset. Nevertheless, the performance of action detection and the interaction recognition is far from perfect. The poor performance is probably due to the limited video dataset. Transferring the knowledge of action and interaction from image could be a further improvement for AIA network.

Acknowledgements. This work is supported in part by the National Key R&D Program of China, No. 2017YFA0700800, National Natural Science Foundation of China under Grants 61772332 and Shanghai Qi Zhi Institute.

References

1. Baradel, F., Neverova, N., Wolf, C., Mille, J., Mori, G.: Object level visual reasoning in videos. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11217, pp. 106–122. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01261-8_7
2. Buades, A., Coll, B., Morel, J.M.: A non-local algorithm for image denoising. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), vol. 2, pp. 60–65. IEEE (2005)
3. Carreira, J., Zisserman, A.: Quo Vadis, action recognition? A new model and the kinetics dataset. In: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6299–6308 (2017)
4. Christoph, R., Pinz, F.A.: Spatiotemporal residual networks for video action recognition. In: Advances in Neural Information Processing Systems, pp. 3468–3476 (2016)
5. Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q.V., Salakhutdinov, R.: Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860 (2019)
6. Damen, D., et al.: Scaling egocentric vision: the epic-kitchens dataset. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11208, pp. 753–771. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01225-0_44
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE (2009)
8. Diba, A., et al.: Spatio-temporal channel correlation networks for action classification. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11208, pp. 299–315. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01225-0_18
9. Caba Heilbron, F., Victor Escorcia, B.G., Niebles, J.C.: Activitynet: a large-scale video benchmark for human activity understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 961–970 (2015)
10. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 6202–6211 (2019)
11. Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional two-stream network fusion for video action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1933–1941 (2016)
12. Girdhar, R., Carreira, J., Doersch, C., Zisserman, A.: A better baseline for AVA. CoRR abs/1807.10066 (2018). http://arxiv.org/abs/1807.10066
13. Girdhar, R., Carreira, J., Doersch, C., Zisserman, A.: Video action transformer network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 244–253 (2019)
14. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448 (2015)

15. Gkioxari, G., Girshick, R., Dollár, P., He, K.: Detecting and recognizing human-object interactions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8359–8367 (2018)
16. Goyal, P., et al.: Accurate, large minibatch SGD: training imagenet in 1 hour. arXiv preprint arXiv:1706.02677 (2017)
17. Gu, C., et al.: Ava: a video dataset of spatio-temporally localized atomic visual actions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6047–6056 (2018)
18. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2961–2969 (2017)
19. Hou, R., Chen, C., Shah, M.: Tube convolutional neural network (T-CNN) for action detection in videos. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5822–5831 (2017)
20. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
21. Ji, S., Xu, W., Yang, M., Yu, K.: 3D convolutional neural networks for human action recognition. IEEE Trans. Pattern Anal. Mach. Intell. **35**(1), 221–231 (2012)
22. Kalogeiton, V., Weinzaepfel, P., Ferrari, V., Schmid, C.: Action tubelet detector for spatio-temporal action localization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4405–4413 (2017)
23. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2117–2125 (2017)
24. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2980–2988 (2017)
25. Lin, T.-Y.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
26. Massa, F., Girshick, R.: Mask R-CNN-benchmark: fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch (2018). https://github.com/facebookresearch/maskrcnn-benchmark. Accessed 29 Feb 2020
27. Qiu, Z., Yao, T., Mei, T.: Learning spatio-temporal representation with pseudo-3D residual networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5533–5541 (2017)
28. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)
29. Sigurdsson, G.A., Divvala, S., Farhadi, A., Gupta, A.: Asynchronous temporal fields for action recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
30. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Advances in Neural Information Processing Systems, pp. 568–576 (2014)
31. Singh, G., Saha, S., Sapienza, M., Torr, P.H., Cuzzolin, F.: Online real-time multiple spatiotemporal action localisation and prediction. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3637–3646 (2017)
32. Soomro, K., Zamir, A.R., Shah, M.: UCF101: a dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012)

33. Taylor, G.W., Fergus, R., LeCun, Y., Bregler, C.: Convolutional learning of spatio-temporal features. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6316, pp. 140–153. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15567-3_11
34. Tran, D., Bourdev, L.D., Fergus, R., Torresani, L., Paluri, M.: C3D: generic features for video analysis. CoRR, abs/1412.0767, vol. 2, no. 7, p. 8 (2014)
35. Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 6450–6459 (2018)
36. Varol, G., Laptev, I., Schmid, C.: Long-term temporal convolutions for action recognition. IEEE Trans. Pattern Anal. Mach. Intell. **40**(6), 1510–1517 (2017)
37. Vaswani, A., et al.: Attention is all you need. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems, vol. 30, pp. 5998–6008. Curran Associates, Inc. (2017). http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf
38. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7794–7803 (2018)
39. Wang, X., Gupta, A.: Videos as space-time region graphs. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11209, pp. 413–431. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01228-1_25
40. Weston, J., Chopra, S., Bordes, A.: Memory networks. arXiv preprint arXiv:1410.3916 (2014)
41. Wu, C.Y., Feichtenhofer, C., Fan, H., He, K., Krahenbuhl, P., Girshick, R.: Long-term feature banks for detailed video understanding. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
42. Xia, J., Tang, J., Lu, C.: Three branches: detecting actions with richer features. arXiv preprint arXiv:1908.04519 (2019)
43. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1492–1500 (2017)
44. Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K.: Rethinking spatiotemporal feature learning for video understanding. arXiv preprint arXiv:1712.04851, vol. 1, no. 2, p. 5 (2017)
45. Yang, X., Yang, X., Liu, M.Y., Xiao, F., Davis, L.S., Kautz, J.: Step: spatio-temporal progressive learning for video action detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 264–272 (2019)
46. Zhang, Y., Tokmakov, P., Hebert, M., Schmid, C.: A structured model for action detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9975–9984 (2019)
47. Zhou, B., Andonian, A., Oliva, A., Torralba, A.: Temporal relational reasoning in videos. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11205, pp. 831–846. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01246-5_49