ADAPTIVE RECURSIVE CIRCLE FRAMEWORK FOR FINE-GRAINED ACTION RECOGNITION

*Hanxi Lin, Wentian Zhao, Xinxiao Wu**

Beijing Laboratory of Intelligent Information Technology
School of Computer Science, Beijing Institute of Technology, Beijing, China
{hxlin, wentian_zhao, wuxinxiao}@bit.edu.cn

ABSTRACT

Intuitively, distinguishing fine-grained actions in videos requires recursively capturing subtle visual cues and learning abstract features. However, existing deep neural network based methods are counter-intuitive in that their network layers do not explicitly model the recursive feature abstraction. Therefore, we are motivated to propose an Adaptive Recursive Circle (ARC) framework that equips common neural network layers with recursive attention and recursive fusion. ARC layer inherits the same operators and parameters as the original layer, but, most critically, it treats the layer input as an evolving state, thus explicitly achieving recursive feature abstraction by alternating the state update and the feature generation. Specifically, at each recursive step, the input state is firstly updated via both recursive attention and recursive fusion from the previously generated features, and then the feature abstraction is performed with the newly updated input state. Significant improvements are observed on multiple datasets. For example, an ARC-equipped TSM-ResNet-18 outperforms TSM-ResNet-50 on the Something-Something V1 and Diving48 datasets with only half overheads. Code will be available at: https://github.com/0HaNC/ARC-ActionRecog.

Index Terms— fine-grained action recognition, recursive representation, visual reasoning, representation learning

1. INTRODUCTION

In action recognition, great progress has been made by introducing end-to-end spatial-temporal CNNs [1,2] on coarse-grained benchmarks [3]. However, actions of these benchmarks are relatively simple and are inclined to be biased by scene and object appearance [4]. Recently, more fine-grained datasets [4,5] have been proposed to evaluate fine-grained action recognition. Taking Something-Something dataset(SS-V1) [5] as an example, as illustrated in Fig.1, the action group of "Pouring something" contains five fine-grained actions. Intuitively, to distinguish the five actions, one is required to recursively capture various action-related visual cues and then make final prediction. Therefore, deep neural network layers are expected to involve more recursive abstractions for learning fine-grained spatial-temporal patterns.

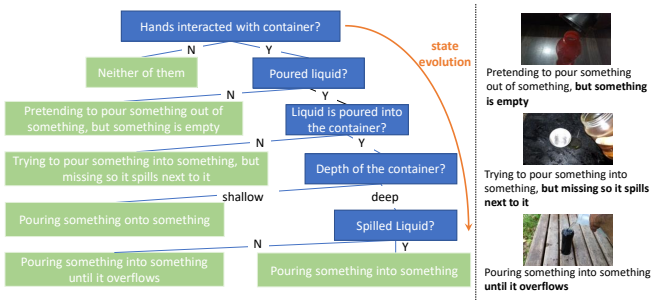

Fig. 1. Left: decision tree for distinguishing fine-grained actions of 'Pouring something' from SS-V1. Right: illustration of typical actions. ARC is conceptually similar to the execution of decision tree in the sense of the recursive feature detections and state evolution.

Most existing deep learning-based methods of fine-grained action recognition resort to better temporal modeling [6–10] or bilinear models [11]. However, none of the aforementioned methods explicitly model the intuition of making a hidden state recursively evolving into more abstract ones within one layer. These methods simply rely on stacking more layers to get finer features, which is counter-intuitive and inefficient.

In this paper, we propose an Adaptive Recursive Circle (ARC) framework to augment common layers (e.g., convolutional layers) with recursive attention and recursive fusion. An ARC layer inherits the same operators (e.g., convolution) and parameters as the original layer, but, most critically, it treats the layer input as an evolving state. The state evolution is achieved by sequentially executing the feature generation process and recursively alternating the feature generation and the state update. Guided by the philosophy of "stacked generalization" [12], the state update is achieved by recursively reusing the previously generated features to refine the input

*Corresponding Author.

This work was supported in part by the Natural Science Foundation of China (NSFC) under Grants No. 62072041.

state. Specifically, at each recursive step, the input state is first modulated by the recursive attention and then is injected with high-order transformed information produced by the recursive fusion. The recursive attention and the recursive fusion are implemented with simple linear transformations with marginal overheads. ARC helps common layers explicitly model an evolving state to progressively capture more abstract features within a layer and finally benefits fine-grained action modeling.

Extensive experiments on the SS-V1 [5], Diving48 [4] and Kinetics-400 [3] datasets show that ARC outperforms state-of-the-art methods on fine-grained action recognition. For example, an ARC-augmented ResNet-18 even outperforms the ResNet-50 counterpart with half FLOPs and half parameters on SS-V1 and Diving48, which validates the effectiveness and efficiency of ARC on fine-grained action modeling. The contributions of our work are two-fold:

1. We introduce the state-evolving intuition to augment a common layer with the capability of progressively modeling high-order transformation within a layer to recognize fine-grained actions.

2. We propose an ARC framework t to instantiate the state evolution. It significantly boosts the performance of fine-grained action recognition without bringing expensive extra computation cost.

2. RELATED WORK

2.1. Fine-grained Action Recognition

Fine-grained action recognition benchmarks [4, 5] refer to more specific action categories, complicated temporal dynamics and less bias of indirect static information (e.g., object and background). To facilitate fine-grained action recognition, Zhu *et al.* [11] explicitly introduce high-order transformations like bilinear operations to capture the fine-grained feature interactions between frames. Zhou *et al.* [13] propose to construct a mid-level object-based part representation and mine the discriminative ones. Munro *et al.* [14] exploit multi-modal alignment in a self-supervision manner.

However, all these works adopt common layers to generate fine-grained features. In this paper, we augment the common layer architecture with state evolution to progressively generate more abstract features, thus further improving the fine-grained action recognition.

2.2. Recursive Representation

Stacked generalization [12] is a kind of ensemble method that recursively stacks learnable layers on the outputs of the already learned classifiers. Following this, Vinyals *et al.* [15] propose to recursively train linear SVMs on a recursively updated hidden state to learn visual features. The state update is governed by random projections of previous SVMs' outputs and the original input.

To the best of our knowledge, we are the first to introduce "stacked generalization" to action recognition. The proposed ARC sequentially executing the feature generation process of a layer and recursively updates the input state using recursive attention and recursive fusion. Compared to traditional recursive representation learning methods, ARC is end-to-end trainable and exploits deep neural network architectures.

3. OUR METHOD

3.1. A Recap on Common Layers

Before diving into the proposed ARC, a recap of the common layers is necessary. Without loss of generality, we take the convolutional layer as an example. Each convolutional layer consists of many feature detectors (i.e., kernels), and each feature detector determines a linear transformation on a local feature map. Through such a linear transformation (i.e., convolution), the input features are abstracted once per layer. The forward pass of a convolutional layer is formulated as

$$\mathbf{Y} = Concat(\{f(\mathbf{X}; \mathbf{w}_1), f(\mathbf{X}; \mathbf{w}_2), ..., f(\mathbf{X}; \mathbf{w}_c)\}) \quad (1)$$

where $\mathbf{X} \in \mathbb{R}^{T \times H \times W \times C_{in}}$ is the input of the layer, $f(\cdot;\cdot)$ is the operator (e.g., convolution) parameterized by weight $\mathbf{w}_i$, c is the number of output channels, $Concat(\cdot)$ is the concatenation operation along the channel dimension, and $\mathbf{Y}$ is the output feature of the layer.

Recalling the intuition illustrated in Fig.1, recognizing fine-grained actions requires recursively capturing various spatial-temporal patterns and building more abstract features. From this aspect, we argue that the simple common layers are unsuitable for fine-grained action modeling. This argument encourages us to augment the common layers with larger model capacity and the sensibility of fine-grained patterns.

3.2. Adaptive Recursive Circle Framework

The core idea of ARC is to make the input state evolve during the feature generation process. As illustrated in Fig.2, we implement the evolution process with recursive global spatial-temporal attention and recursive fusions. Note that other variants of implementation are also possible.

The forward pass of an ARC layer is formulated as

$$\mathbf{Y} = Concat(\{f(\mathbf{X}_1; \mathbf{w}_1), f(\mathbf{X}_2; \mathbf{w}_2), ..., f(\mathbf{X}_c; \mathbf{w}_c)\}) \quad (2)$$

where $\mathbf{X}_1$ is the initial state that equals to the original input $\mathbf{X}$. It sequentially evolves into $\mathbf{X}_2$, $\mathbf{X}_3$ and so on. Eventually it arrives at the final state $\mathbf{X}_c$. Except for the state evolution, everything else remains unchanged when compared to Eq.1.

In practice, the forward pass of ARC layer is splited into n steps, and n is a factor of c. The values of every $\frac{c}{n}$ consecutive states (e.g., $\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_{\frac{c}{n}}$) are identical. Each step generates $\frac{c}{n}$ channels of output features and make the input evolve into a new state. The evolution of the states is formulated as

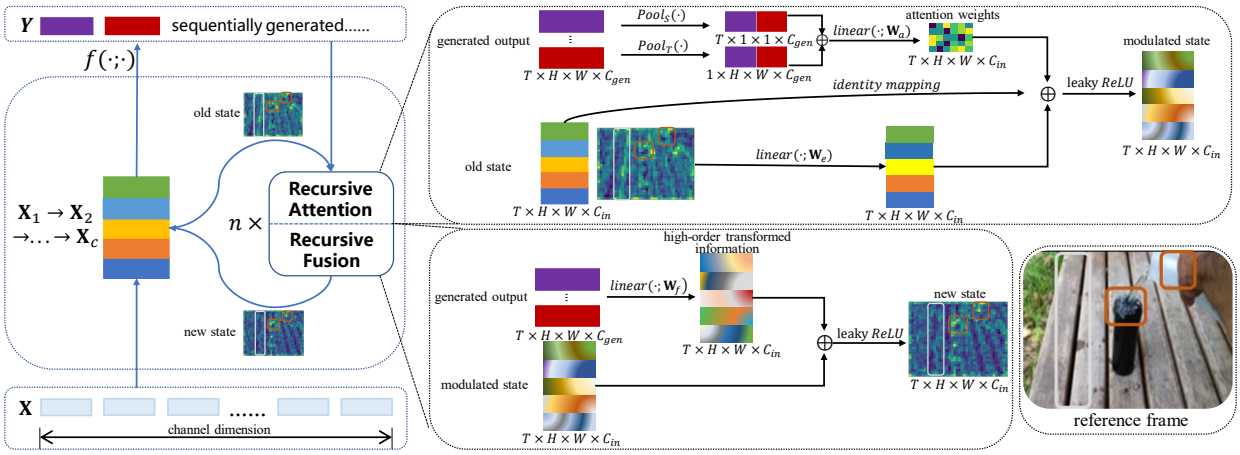

Fig. 2. The proposed ARC framework makes layer input evolve. Guided by the previously generated features, the recursive global spatial-temporal attention suppresses the irrelevant features (grey box) or excites the important ones (orange boxes). Then the recursive fusion distills the high-order transformed features into the modulated state.

$$\mathbf{X}_i = \begin{cases} \mathbf{X} & , i \leq \frac{c}{n} \\ \phi(\phi(\mathbf{X} + \mathbf{W}^e\mathbf{X} + \mathcal{A}(\mathbb{F}_{\lfloor i/\frac{c}{n} \rfloor})) + \\ \mathcal{F}(\mathbb{F}_{\lfloor i/\frac{c}{n} \rfloor})) & , \text{otherwise} \end{cases} \tag{3}$$

where ϕ is the leaky ReLU, $\mathbf{X}$ is the original input, $\mathbf{W}^e \in \mathbb{R}^{C_{in} \times C_{in}}$ is the embedding matrix initialized to zero matrix, $\mathcal{A}(\cdot)$ and $\mathcal{F}(\cdot)$ represent the recursive attention and the recursive fusion, respectively, i is the current recursive step, $\lfloor \cdot \rfloor$ denotes rounding down, $\mathbb{F}_j$ is the features that have been generated (i.e., $\{f(\mathbf{X}_1;\mathbf{w}_1), f(\mathbf{X}_2;\mathbf{w}_2), ..., f(\mathbf{X}_{j\frac{c}{n}};\mathbf{w}_{j\frac{c}{n}})\}$. $\mathbb{F}_j$ has C_{gen} channels of feature in total.

3.2.1. *Recursive Attention*

At different recursive steps, Recursive Attention (RA) learns to modulate the state by suppressing or exciting different channels according to the previously generated features. RA is formulated as

$$\begin{aligned} &\mathcal{A}(\mathbb{F}_j) = \\ &\mathbf{W}^{aj}(Pool_S(Concat(\mathbb{F}_j)) \oplus Pool_T(Concat(\mathbb{F}_j))) \end{aligned} \tag{4}$$

where $\mathbf{W}^{aj}$ is $(\mathbf{W}^a_{km})_{\substack{1 \leq k \leq C_{in} \\ 1 \leq m \leq j\frac{c}{n}}}$, $\mathbf{W}^a \in \mathbb{R}^{C_{in} \times c}$ is the weight matrix for attention generation, $Pool_S(\cdot)$ and $Pool_T(\cdot)$ represent the spatial max pooling and temporal max pooling, respectively, and $\oplus$ denotes summation with broadcasting.

RA is a kind of channel attention that is based on the high-order transformed information aggregated globally across the spatial-temporal dimensions. We use the additive attention instead of the common multiplicative attention, since the additive attention works closely with the leaky ReLU nonlinearity in Eq.3 to suppress some channels to nearly zero or excite other channels at different recursive steps. We hope such a mechanism followed by the recursive fusion can make the input state evolve into a new state that contains less irrelevant information and is sensitive to fine-grained spatial-temporal patterns.

3.2.2. *Recursive Fusion*

After RA explicitly modulated the state, Recursive Fusion (RF) fuses the modulated state and the previously generated high-order transformed features into a new state. RF is formulated as

$$\mathcal{F}(\mathbb{F}_j) = \mathbf{W}^{fj}(Concat(\mathbb{F}_j)) \tag{5}$$

where$\mathbf{W}^{fj}$ is $(\mathbf{W}^f_{km})_{\substack{1 \leq k \leq C_{in} \\ 1 \leq m \leq j\frac{c}{n}}}$, $\mathbf{W}^f \in \mathbb{R}^{C_{in} \times c}$ is an embedding matrix initialized to zero matrix.

RA and RF together turn the original input into several new states that are sensible to subtle spatial-temporal patterns, greatly increasing the model capacity for fine-grained action recognition.

4. EXPERIMENTS

4.1. Datasets

We evaluate ARC on fine-grained datasets like the SS-V1 [5] and Diving48 [4] datasets. We also conduct experiments on the coarse-grained Kinetics400 [3] dataset to show the generality of ARC.

4.2. Implementation Details

ResNet [20] equipped with temporal modeling modules [7,9] is chosen as our backbone. All convolutional layers except the 1×1 point-wise ones are augmented with ARC.

Sparse sampling [16] is adopted to sample frames. SGD with momentum is chosen as our optimizer. For SS-V1 and

Table 1. Performance comparison on SS-V1. Only RGB models are reported. † denotes our implementations.

Model	Backbone	Pre-train	Frames	FLOPs×clips/G	Param./M	Val Top-1/Top-5(%)
Spatial backbones + late fusion:						
TSN [16] from [7]	ResNet-50	ImgNet	8	33G×1	24.3	19.7/-
TRN [6]	BN-Inception	ImgNet	8	8×N/A	10.5	34.4/-
Temporal modeling modules:						
TSM-RGB [7]	ResNet-50	K400	16	65×1	24.3	47.3/77.1
GSM [8]	InceptionV3	ImgNet	16	54×2	22.2	51.7/-
CorrNet-R101 [17]	ResNet-101	-	32	187×10	-	50.9/-
MSNet [9]	TSM-ResNet-50	ImgNet	16	67×1	24.6	52.1/82.3
Spatial-temporal backbones:						
I3D from [18]	3D ResNet-50	ImgNet+K400	32	153×2	28.0	41.6/72.2
TEA [10]	ResNet-50	ImgNet	16	70×30	-	52.3/81.9
RubiksNet [19]	ResNet-50	ImgNet	8	16×1	8.5	46.4/74.5
Fine-grained models:						
ABM-iabp [11]†	ResNet-18	ImgNet	8×3	30.4×1	26.8	45.1/74.0
ABM-C-in [11]	ResNet-50	ImgNet	16×3	-	-	49.8/-
ARC (ours)	TSM-ResNet-18	ImgNet	8	17×1	14.2	47.9/76.4
ARC (ours)	TSM-ResNet-50	ImgNet	8	37×1	27.0	51.2/79.0
ARC (ours)	TSM-ResNet-50	ImgNet	16	74×1	27.0	53.4/81.8
ARC (ours)	TSM-ResNet-50	ImgNet	8+16	111×1	54.0	**55.0**/82.6

Table 2. Performance comparison with (perhaps deeper) common layer baselines. † denotes our implementations.

	Model	Backbone	FLOPs(G)	Param.(M)	Top-1(%)	Gain
SS-V1	TSM†	ResNet-18	15	11	41.8	-
	TSM†	ResNet-50	33	24	47.8	-
	TSM†	ResNet-101	66	44	50.0	-
	ARC-TSM	ResNet-18	**17**	**14**	**47.9**	**+6.1**
	ARC-TSM	ResNet-50	**37**	**27**	**51.2**	**+3.4**
Diving48	TSM†	ResNet-18	58	11	36.0	-
	TSM†	ResNet-50	132	24	38.8	-
	TSM†	ResNet-101	264	44	41.9	-
	ARC-TSM	ResNet-18	**69**	**14**	**39.6**	**+3.6**
	ARC-TSM	ResNet-50	**149**	**27**	**42.4**	**+3.6**

Kinetics400, multi-step learning rate schedule is adopted. For Diving48, a cosine learning rate schedule with gradual warm-up is adopted, following [8]. For all the datasets, the number of recursive steps of ARC is set to 4, unless otherwise specified. Other training hyper-parameters are summarized in the supplementary material.

4.3. Experimental Results

4.3.1. Comparison with Common Layer Baselines

We choose TSM [7] backboned by ResNets [20] as a general baseline that is composed of common layers, and conduct a fair comparison between the baselines and the ARC-augmented counterparts on fine-grained datasets.

The results are reported in Table 2. We have the following observations. First, ARC achieves significant improvements over the baseline. Second, the overhead increase for ARC is marginal. The ARC-augmented ResNet-18 models even outperform the ResNet-50 counterparts with half overheads. Third, ARC scales well to larger backbones. These observations highlight the superiority of making the input state evolve by involving RA and RF in fine-grained action recognition.

4.3.2. Comparison with Existing Fine-grained Models

We compare ARC with the bilinear families which are classic solutions for fine-grained modeling. Results are summarized in Table 1. We find that ARC models outperform ABM [11] by a large margin. Note that the results are reproduced in the same setting for a fair comparison.

4.3.3. Comparisons with State-of-the-art Methods

Table 1 summarizes the results on SS-V1. We divide the existing methods into three categories: spatial backbones with late temporal fusion, temporal modeling module, and spatial-temporal backbones. The first category loses fine-grained temporal information in the early stages, resulting in low performance. The others incorporate temporal modeling across all stages, thus obtaining relatively higher performance. However, these methods all adopt common layers. Augmenting the same backbones with ARC achieves best performance. These observations demonstrate the significant improvement of ARC augmentation on fine-grained action modeling. In terms of efficiency, we also report the FLOPs of the ARC models backboned by ResNet-18. Compared to the most efficient method, i.e. RubiksNet [19], our ARC models achieve 1.5% and 3.1% higher accuracy respectively with similar FLOPs.

Table 3 summarizes the results on Diving48. For Diving48-V1, with nearly half overheads, the ARC-augmented TSM-ResNet-18 even outperforms TSM-ResNet-50 by 0.8%. It also outperforms other methods with much

Table 3. Performance comparison on Diving48-V1 and Diving48-V2. † denotes our implementations.

Model	Backbone	Pre-train	Frames	FLOPs/G× clips	Param./M	V1 Top-1(%)	V2 Top-1(%)
TSM [7]†	ResNet-50	ImgNet	16	65×2	24.3	38.8	-
CorrNet-R101 [17]	ResNet-101	-	32	187×10	-	38.2	-
GSM [8]	IncV3	ImgNet	16	54×2	-	40.3	-
SlowFast16x8(from [21])	ResNet101	ImgNet	16×30	213×30	-	-	77.6
TimeSformer-L [21]	ViT-base	ImgNet	16×30	238×30	-	-	81.0
ARC (ours)	TSM-ResNet-18	ImgNet	16	34.2×2	14.2	39.6	-
ARC (ours)	TSM-ResNet-50	ImgNet	16	74.2×2	27.0	**42.4**	**89.8**

fewer overheads. Furthermore, the ARC-augmented TSM-ResNet-50 achieves best performance of 42.4% top-1 accuracy. For Diving48-V2, similar results are observed. ARC achieves best performance in both accuracy and efficiency.

Table 4. Ablation study.

(a) Ablations on RA and RF.

recursive attention	recursive fusion	Top-1(%)
		41.8
	✓	46.3
✓		46.7
✓	✓	**47.9**

(b) Performance comparison on different number of recursive steps.

n	FLOPs/G	Param./M	Top-1(%)
1	14.6	11.3	41.8
2	17.2	14.2	46.5
4	17.2	14.2	**47.9**

(c) Ablations on designs of recursive attention.

Interaction	Aggregation	Attention module	Top-1(%)
Multiplicative	S	GRU	45.0
	S	FC	46.2
Additive	S	GRU	45.6
	ST	FC	46.5
	T	FC	47.5
	S	FC	47.0
	S+T	FC	**47.9**

Table 5 summarizes the results on Kinetics400. Even though Kinetics400 is not a fine-grained dataset, ARC still achieves comparable results with existing state-of-the-art methods, showing the generality of ARC.

4.3.4. Ablation Analysis

We analyze the ablation of ARC on SS-V1. Unless otherwise specified, the backbone, n and the number of sampled frames are fixed to TSM-ResNet-18, 4 and 8, respectively.

RA and RF. We remove RA or RF to evaluate the contributions of each module. Results are summarized in Table 4(a). Interestingly, RA alone works better than RF , while the feature resolution is hurt by the pooling operations of RA. It suggests that the separated design of spatial-temporal pooling preserves enough action structures to produce attention while reducing computational burdens. Overall, all components contribute to the improvements, especially for the RA.

Number of Recursive Steps. We tune n to see the performance trends concerning n. The results are shown in Table 4(b). Performance improves when n is larger. It is intuitive in that larger n brings longer recursions and generates more abstract features, thus benefiting fine-grained action recognition. It is worth noting that the FLOPs and the number of parameters are constant after n exceeds 2. But, as shown in the overhead analysis in supplementary material, the linearly growing memory consumption hinders the increase of n.

Design of RA. We investigate the three aspects of RA as reported in Table 4(c). For the interaction type, additive attention outperforms multiplicative attention. We argue that additive operations are more inclined to bias some activations to the zero-response zone of the ReLU non-linearity and excite others to learn the feature modulation. On the contrary, the diminished gradient problem of the sigmoid function of the multiplicative attention prevents ARC from generating concentrated attention. For the information aggregation, we compare spatial pooling ("S"), spatial-temporal pooling ("ST"), temporal pooling ("T"), and broadcasting summation of separated spatial pooling and temporal pooling ("S+T"). "S+T" performs best since it preserves more coordinated information. For the attention module, simple fully connected layer ("FC") outperforms GRU, probably due to the over-parameterization of GRU.

4.4. Interpretation

We visualize the state evolution and the class activation maps (CAM) [22] of the ARC models.

State Evolution. The process of recursive refinement is the key feature of the proposed ARC. Thus we are interested in how it works. As shown in Fig.3, the states of all recursive steps are averaged across channel dimensions, reflecting the degree of excitation in different regions. At the first recursive step, the feature maps show clear interests in object of both the foreground and the noisy background (e.g., the gap between the planks). But as the recursion goes on, the action-irrelevant are greatly suppressed. At the end of the recursion,

ARC mainly focuses on action-related and discriminative patterns (e.g., the overflowing water and the withdrawn hand).

Class Activation Mapping. We visualize the spatial attention of the ARC model. As shown in Fig.4, ARC captures fine-grained action-related details, such as, the overflowing water, the empty container, the absence of poured water, the torn paper, and the withdrawn hand.

Table 5. Performance comparison on Kinetics400.

Model	Backbone	Frames	FLOPs/G×clips	Val Top-1(%)
TSN [16]	InceptionV3	25	3.2×250	72.5
MSN [9]	TSM-ResNet-50	8	34×10	75.0
TSM [7]	ResNet-50	8	33×30	74.1
I3D [2]	BN-Inception	64	108×N/A	72.1
SlowOnly [23]	ResNet-50	8	41.9×10	74.8
TEA [10]	ResNet-50	8	35×10	75.0
ARC (ours)	TSM-ResNet-50	8	37×30	75.0

Fig. 3. Visualization of state evolution. ARC suppresses the irrelevant objects and scenes (grey boxes), and enhances the fine-grained action-related details (orange boxes).

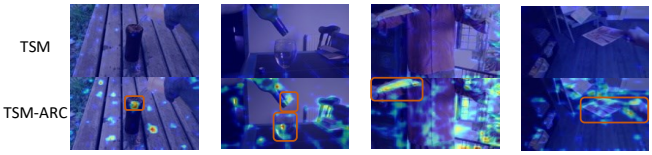

Fig. 4. CAM visualizations. ARC captures action-related fine-grained details (orange boxes) that the baseline misses.

5. CONCLUSION

We have presented the state-evolving intuition for fined-grained action recognition. We implement it with a newly proposed ARC framework, which is the first to model the recursive fine-grained refinement of actions within a neural network layer. ARC is general and can be easily applied to most existing backbones. Extensive experiments demonstrate the strong performance and efficiency of ARC on fine-grained action recognition. In the future, we are going to extend our method to more network architectures such as Transformer.

6. REFERENCES

[1] Tran et al., "Learning spatiotemporal features with 3d convolutional networks," in *ICCV*, 2015. 1

[2] Carreira et al., "Quo vadis, action recognition? a new model and the kinetics dataset," in *CVPR*, 2017. 1, 6

[3] Kay et al., "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017. 1, 2, 3

[4] Li et al., "Resound: Towards action recognition without representation bias," in *ECCV*, 2018. 1, 2, 3

[5] Goyal et al., "The" something something" video database for learning and evaluating visual common sense.," in *ICCV*, 2017. 1, 2, 3

[6] Zhou et al., "Temporal relational reasoning in videos," in *ECCV*, 2018. 1, 4

[7] Lin et al., "Tsm: Temporal shift module for efficient video understanding," in *ICCV*, 2019. 1, 3, 4, 5, 6

[8] Sudhakaran et al., "Gate-shift networks for video action recognition," in *CVPR*, 2020. 1, 4, 5

[9] Kwon et al., "Motionsqueeze: Neural motion feature learning for video understanding," in *ECCV*, 2020. 1, 3, 4, 6

[10] Li et al., "Tea: Temporal excitation and aggregation for action recognition," in *CVPR*, 2020. 1, 4, 6

[11] Zhu et al., "Approximated bilinear modules for temporal modeling," in *ICCV*, 2019. 1, 2, 4

[12] Wolpert et al., "Stacked generalization," *Neural networks*, 1992. 1, 2

[13] Zhou et al., "Interaction part mining: A mid-level approach for fine-grained action recognition," in *CVPR*, 2015. 2

[14] Munro et al., "Multi-modal domain adaptation for fine-grained action recognition," in *CVPR*, 2020. 2

[15] Vinyals et al., "Learning with recursive perceptual representations," *NIPS*, 2012. 2

[16] Wang et al., "Temporal segment networks: Towards good practices for deep action recognition," in *ECCV*, 2016. 3, 4, 6

[17] Wang et al., "Video modeling with correlation networks," in *CVPR*, 2020. 4, 5

[18] Wang et al., "Non-local neural networks," in *CVPR*, 2018. 4

[19] Fan et al., "Rubiksnet: Learnable 3d-shift for efficient video action recognition," in *ECCV*, 2020. 4

[20] He et al., "Deep residual learning for image recognition," in *CVPR*, 2016. 3, 4

[21] Gedas Bertasius, Heng Wang, and Lorenzo Torresani, "Is space-time attention all you need for video understanding?," in *ICML*, 2021. 5

[22] Zhou et al., "Learning deep features for discriminative localization," in *CVPR*, 2016. 5

[23] Feichtenhofer et al., "Slowfast networks for video recognition," in *ICCV*, 2019. 6