Reinforcement Learning with Action-Specific Focuses in Video Games

Wang Meng
Fuxi AI Lab in Netease
Hangzhou, China
wangmeng02@corp.netease.com

Chen Yingfeng*
Fuxi AI Lab in Netease
Hangzhou, China
chenyingfeng1@corp.netease.com

Lv Tangjie
Fuxi AI Lab in Netease
Hangzhou, China
hzlvtangjie@corp.netease.com

Song Yan
Fuxi AI Lab in Netease
Hangzhou, China
songyan@corp.netease.com

Guan Kai
Fuxi AI Lab in Netease
Hangzhou, China
guankai@corp.netease.com

Fan Changjie
Fuxi AI Lab in Netease
Hangzhou, China
fanchangjie@corp.netease.com

Yu Yang
Nanjing University
Nanjing, China
yuy@nju.edu.cn

***Abstract*—It is intuitive that different actions prefer different information in human decisions. However, classical reinforcement learning models use the same information process procedure for all actions. In order to imitate human decision-making process closer, in this paper we investigate a new policy model, i.e., Action-Specific Focuses (ASF) framework, which enables different focuses when learning different actions. In the ASF framework, the whole action set is taken as part of the queries for the attention module, in which state-dependent action-specific features can be generated. Through extracting different action-specific features, our approach enables the agent to learn the action–focus map for each action separately. The ASF framework is also different from the previous usages of attention mechanisms in reinforcement learning that are mostly based on the state. Experiments on the Atari benchmark show that ASF is able to improve the performance in various types of games. Moreover, the visualizations of the attention weights suggest that ASF can learn meaningful focuses when taking different actions.**

***Index Terms*—artificial intelligence, deep reinforcement learning, attention**

I. Introduction

In recent years, deep reinforcement learning (RL) has achieved incredible performance in learning various tasks such as Go [1] and Atrai games [2]. More and more algorithms [3], [4] have been proposed and they can generate an agent that far exceeds average human-level in many Atari games. However, classical RL algorithms use the same information process procedure for all actions, whereas the human brain prefers to generate a specific focus for each action under the same state.

We provide a life-common example in Figure 1. Before taking a turn in driving, one usually glances at the rearview mirror to ensure that the left or right rear is safe for the "turn" action. Similarly, in order to keep a safe distance from the front and rear vehicles, one will focus on the areas outside the front window and in the interior mirror for "accelerate" and "brake" actions respectively. In these cases, the focus depends on the planned actions ("turn", "accelerate" or "brake"). And this action-specific attention mechanism enables the human brain to make accurate decisions with its limited capacity. In cognitive science, this intention-guided attention mechanism is called the top-down attention [5]. Correspondingly, the bottom-up attention is purely driven by the external factors, which is similar to the classical state-determined attention mechanisms used in RL.

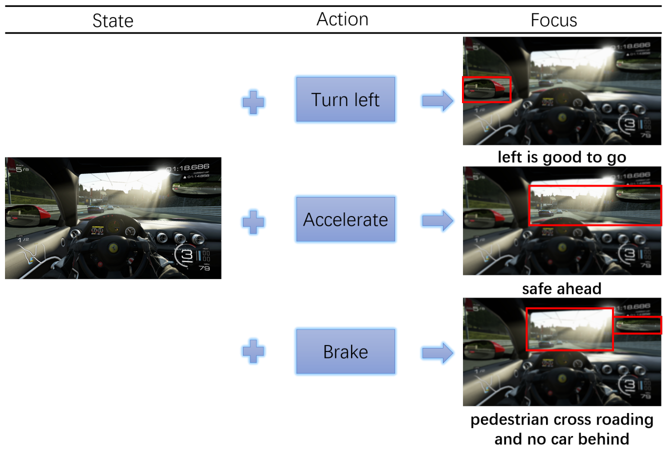

Fig. 1: Relationship between the human focuses and the actions when driving a car. The red rectangles in the right column represent the focuses for the corresponding actions in the middle column.

In this paper, we propose the Action-Specific Focuses (ASF) framework to imitate the top-down decision-making process of the human. In the ASF framework, actions play the roles of the intentions in human's top-down attention, and in order to learn the action–focus map for each action separately, we leverage the attention mechanism to select the important features specific to different actions. Note that current attention functions [6], [7] used in RL only input the states as the queries. Our approach is different from them in that we believe

*Corresponding author.

the intended action is also a cause of the attention. As a result, our method enables the agent to focus on different parts of the features for different actions.

We experimentally analyze the attention weights for different actions in ASF, checking whether the focuses generated by ASF are consistent with the human intuition. And the results show that ASF successfully produces meaningful attention for each action. We also evaluate our approach on the entire Atari benchmark, and the results demonstrate that ASF can improve the performance both in convergence speeds and best results compared to the state-of-the-art algorithms.

II. Related Work

A. Biological Motivation

There is evidence that the human brains naturally reduce the dimensions of the real-world problems with attention mechanisms to solve the “curse of dimensionality” problem [8]. Furthermore, the attention mechanism in the human cognitive system can be classified into two categories. One is generated volitionally by top-down signals determined by a specific goal, and the other is driven automatically by bottom-up signals associated with unexpected, novel or salient stimuli [5], [9]. Based on these studies, our work focuses on improving the RL algorithm by introducing a more explicit and effective way to imitate this top-down decision-making process of the human.

B. Attention in RL

Attention models are widely used in domains such as image captioning [10] and machine translation [11], [12]. Recently more and more researchers introduce the attention mechanism into RL, and these studies can be divided into three categories: temporal attention, spatial attention and the combination of them. Similar to the attention methods used in machine translation, [6] models the RL as a multiple-step decision-making problem and then uses an attention function to compute the priorities of the information belonging to different former steps in memory. The differential neural computer (DNC) [13] involves a neural network controller that reads/writes from an external memory matrix through attention mechanism, which outperforms the long short-term memory (LSTM) in a variant of the Tower of Hanoi task. In contrast, some other researchers leverage the spatial attention mechanism used in image captioning to help the agent to concentrate on crucial regions [7] or important entities [14] in the input image. Reference [15] uses a spatial attention mechanism to determine whether each region in the input image is controllable by the agent for tackling exploration problems in Atari games. Furthermore, [16] and [17] combine the temporal attention and the spatial attention in the framework of deep recurrent Q-network (DRQN) and deep recurrent deterministic policy gradient (DRDPG) respectively, which use the hidden state of the LSTM as the attention query. From the perspective of attention mechanism, the biggest difference between our work and the previous work is that: for a given state, we generate different attentions for different actions, whereas the traditional methods use the same attention for all actions.

C. Top-Down Attention

In practice, different approaches model the top-down attention in different ways. In the human visual system, [18] uses the top-down attention to locate a given target object in an image, and the ”given target object” is involved for the attention computation. In the autonomous driving systems, [19] and [20] introduce the high-level navigation commands (generated by an external navigation module or human) as the conditions and the supervised learning losses are computed based on them. In this paper, we construct the top-down attention by imitating the human decision-making process, i.e., generating different attentions for different actions under the same state. To the best of our knowledge, the most similar work to our method in this regard is [21], which reconstructs different images according to different prior biases when classifying noisy handwriting digits, and these prior biases indicate the probabilities that the input image belongs to each class. The method in [21] needs a pre-train step and is used in the supervised learning setting with the direct and undelayed error signals, whereas our method is end-to-end differentiable and able to adapt to the RL process with delayed reward signals.

III. Method

As shown in Figure 2, we introduce an additional Action-Specific Focuses (ASF) module into the pipeline of the actor network. The ASF module processes the state and outputs multiple ASF feature vectors, and each vector corresponds to a specific action. An attended feature vector is computed based on each of these ASF feature vectors, and then is sent to the policy network. The multiple attended feature vectors lead to multi-dimensional policy network output. After the output reduction module and a softmax function, we will get the final action distribution. Note that the attention function is differentiable, so we can train the whole neural network end-to-end with the policy gradient algorithm. We present details of the approach in the following subsections.

A. Notations

We use the tuple $(\mathcal{S}, \mathcal{A}, T, r, \gamma)$ to define a discounted Markov decision process (MDP), where $\mathcal{S}$ represents the state space, $\mathcal{A}$ represents the action space, $T_{s'}^{s,a} = P(s'|s,a)$ is a function denoting the probability of the environment transition from the state s to the next state s' if the action a is selected, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function mapping a state-action pair to a reward that is provided by the environment, and $\gamma \in [0, 1]$ is a discount factor. A policy in MDP is denoted as π and $\pi(a|s)$ is the probability of taking action a under the state s with π. Similarly, $\pi(\cdot|s)$ is the probability distribution of the actions given state s. Furthermore, let $|\mathcal{S}|$ denote the size of the state space. Let $|\mathcal{A}|$ denote the size of the action space in the discrete action environments and the dimension of the action vector in the continuous action environments.

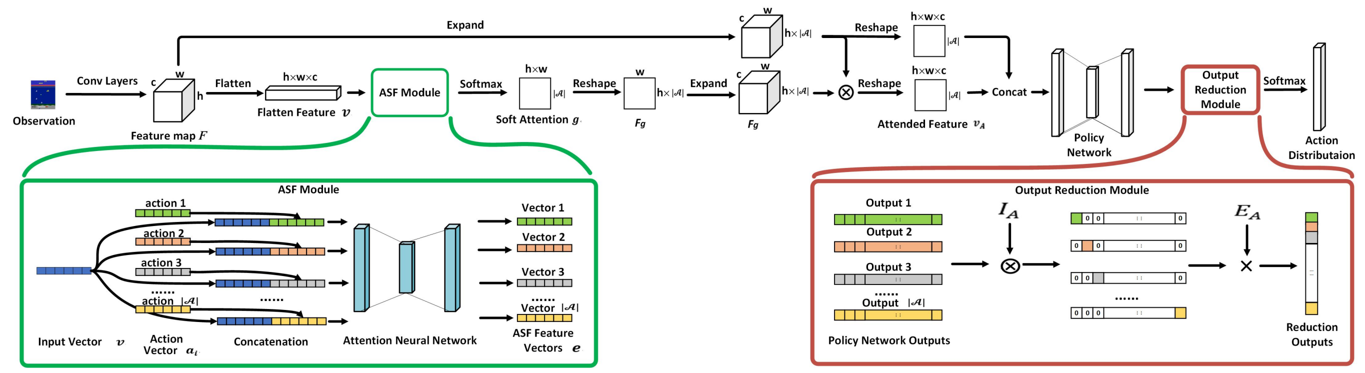

Fig. 2: Framework of our method, where $\times$ denotes the matrix multiplication, $\otimes$ denotes the element-wise multiplication, $I_A \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{A}|}$ is an identity matrix and $E_A \in \mathbb{R}^{|\mathcal{A}|}$ is a vector with all elements equaling to one.

B. *Action-specific Focuses*

In the ASF module, we aims to find the important dimensions of the input if we plan to execute a specific action at the current step. According to [12], an attention function can be described as mapping a query and a set of key-value pairs to an output. Thus, in the view of attention mechanism, our goal can be achieved through regarding the action as the query and each dimension of the feature vector as the key/value. Given an input vector v, we will compute a normalized attention weight vector g_{a_i} for each of the $|\mathcal{A}|$ actions as follows:

$$g_{a_i} = softmax([v, a_i]W + b) \tag{1}$$

where $a_i \in \mathbb{R}^{|\mathcal{A}|}$ is the one-hot representation of an action, $W \in \mathbb{R}^{(|\mathcal{A}|+|v|)\times|v|}$ and $b \in \mathbb{R}^{|v|}$ are the parameters to be learned, and $[v, a_i]$ means the concatenation of v and a_i. Our attention function is a variant of the perceptron attention function. We follow the understanding that attention is a kind of mapping and leverage a small neural network to approach this mapping. Then we can obtain the new attended feature v_{a_i} for action a_i through multiplying g_{a_i} with v element-wise, which is formulated as follows:

$$v_{a_i} = g_{a_i} \otimes v \tag{2}$$

where $\otimes$ denotes the element-wise multiplication. Since there are totally $|\mathcal{A}|$ actions available, we can repeat Eq.(1) and Eq.(2) for each of the $|\mathcal{A}|$ actions. In order to accelerate the computation, we apply the attention function on all candidate actions simultaneously. In practice, all actions are packed together as a one-hot identity matrix $I_A \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{A}|}$. The input vector v is also expanded to a matrix V by repeating v for $|\mathcal{A}|$ times and stacking them together. The attention weight matrix G and the attended feature matrix V_A are computed as:

$$G = softmax([V, I_A]W + b) \tag{3}$$

$$V_A = G \otimes V \tag{4}$$

Previously, it is an intuitive idea to use the attended feature matrix V_A as the only input to the subsequent networks, but recent study [22] has shown that it is helpful to take the original features into consideration. Therefore, we add a highway connection from the original feature matrix V and input the the concatenation of V_A and V to the policy network. Since we have transformed a single state vector to a two-dimensional matrix in ASF module, we introduce the output reduction module to obtain the one-dimensional $\pi(\cdot|s)$ as the vanilla policy network generates. The output reduction module will select the ith element from the ith output of the policy network. After the softmax layer, we can get $\pi(\cdot|s)$ as follows:

$$\pi(\cdot|s) = softmax((FC_n([V, V_A]) \otimes I_A) \times E_A) \tag{5}$$

where $\times$ denotes the matrix multiplication, FC_n is a fully connected neural network with n hidden layers, and $E_A \in \mathbb{R}^{|\mathcal{A}|}$ is a vector with all elements equaling to one.

C. *Training*

We use the actor-critic architecture to train the overall RL model. The output of the actor network is given in Eq.(5), and the meaning of it is an action probability distribution, which is the same as the output of the vanilla actor network without our method. We do not introduce the attention mechanism in the critic so its architecture is not changed. Given the above similarities, we can train our model through the policy gradient algorithm as the vanilla actor-critic model does. The loss functions of the actor and the critic are shown in Eq.(6) and Eq.(7) respectively, where $A(s, a)$ is an estimator of the advantage function, s' is the next state, $H(\pi)$ is the entropy of policy π, β is a hyperparameter and $\pi(a|s)$ is given by Eq.(5).

$$L_{actor} = -\mathbb{E}(\log(\pi(a|s)))A(s, a)) - \beta H(\pi) \tag{6}$$

$$L_{critic} = \frac{1}{2}\mathbb{E}(r + \gamma V(s') - V(s))^2 \tag{7}$$

We train the attention network and the policy network simultaneously acorrding to the loss computed with Eq.(6) and Eq.(7). Note that the forms of them are the same as the loss functions of the vanilla actor-critic model, and thus all algorithms based on actor-critic architecture can use our method by generating $\pi(a|s)$ following Eq.(5).

The last problem of the training is the additional computation cost for each action. In the common environments with

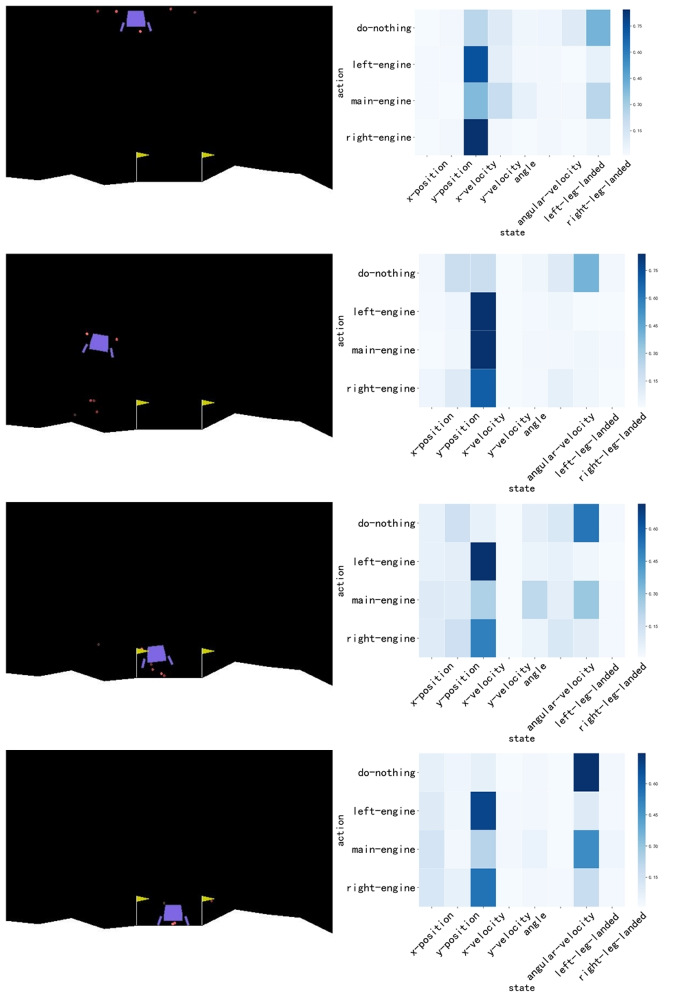

Fig. 3: Game screens (*left*) and corresponding visualizations of attentions (*right*) in LunarLander, in which the state is an eight-dimensional vector and there are four discrete actions. In the attention matrices (*right*), the abscissa indicates the state and the ordinate indicates the action.

moderate action space size, we can accelerate the training process through introducing the vectorization operation to compute all actions' focuses simultaneously, and the training speed of ASF on GPU is acceptable in practice. For the environments with extremely large action space size, we can use the action embedding techniques proposed by [23] and sample some representative action vectors from the action embedding space, and then calculate the focuses based on them. Note that even for the RL algorithms without ASF, the additional action embedding process is needed for solving the problems with extremely large action space [23]. Furthermore, our method is also suitable for the environments with continuous action space, in which the policy network will output the means and the standard deviations of the actions. The one-hot identity matrix I_A used in Eq.(3) and Eq.(5) will not change, but each row of it represents a dimension of the continuous action vector instead of a candidate action. We leave the application of ASF in the environments with extremely large action space or continuous action space to future work.

IV. Experiment and Analysis

A. Visualization of Attention

First, in order to verify whether the attentions generated by the ASF is consistent with the human prior knowledge, we test our method on a tiny game named LunarLander in OpenAI Gym [24], and record the attention matrices during training. Examples of the game screens and corresponding attention matrices are shown in Figure 3. At the beginning of one episode (the first row in Figure 3), the lander is high and we can see that different actions focus on different dimensions of the state according to their effects: the "left-engine" action and the "right-engine" action focus on the velocity in horizontal direction because their major task is to produce horizontal forces, whereas the "main-engine" action focuses on the velocity in both vertical and horizontal direction. As the lander gets closer to the ground, if it has a large bias in the horizontal direction (the second row in Figure 3), all actions related to the engine will focus on the horizontal velocity to control the lander more precisely in that direction, which is important for getting back to the right track. When the lander gets close enough to the ground (the third row in Figure 3), the "main-engine" will change its focus to the angle and the legs of the lander, because these dimensions are important for adjusting the attitude of the lander so as to make it ready for the final contact. At the end of the landing (the last row in Figure 3), the "do-nothing" action and the "main-engine" action will attach great importance to the status of the legs to confirm that the landing is completed. From this process, we can clearly observe that different actions will focus on different dimensions in the state, and our approach successfully produces a proper attention weight for each action as well as captures the changes in attention during the whole episode. We provide a video example at https://youtu.be/CWyJSA__vp4.

B. Results in Atari Games

Next, we test our method on the Atari benchmark and record the scores to quantitatively evaluate the performance of ASF. We use Proximal Policy Optimization (PPO) [4] as the baseline and implement ASF based on it. Furthermore, in order to figure out the effects of the slight differences between the vanilla PPO and the ASF in the network architecture (shown in Figure 4(a) and Figure 4(c) respectively), we add another baseline called State Attention and its network architecture is shown in Figure 4(b). State Attention uses the same attention mechanism as ASF except that it only uses the states as the queries. All three methods take the raw images as inputs and leverage a three-layer CNN network to extract the vision features, and we use the sigmoid activation function after ASF module becase it performs better for images than the softmax activation function. Each method is trained for 20 million timesteps per game with 8 actors. We tested the three methods on the whole Atari suite, which contains 58 games as the time of writing. And in the beginning of every episode, the agent will execute

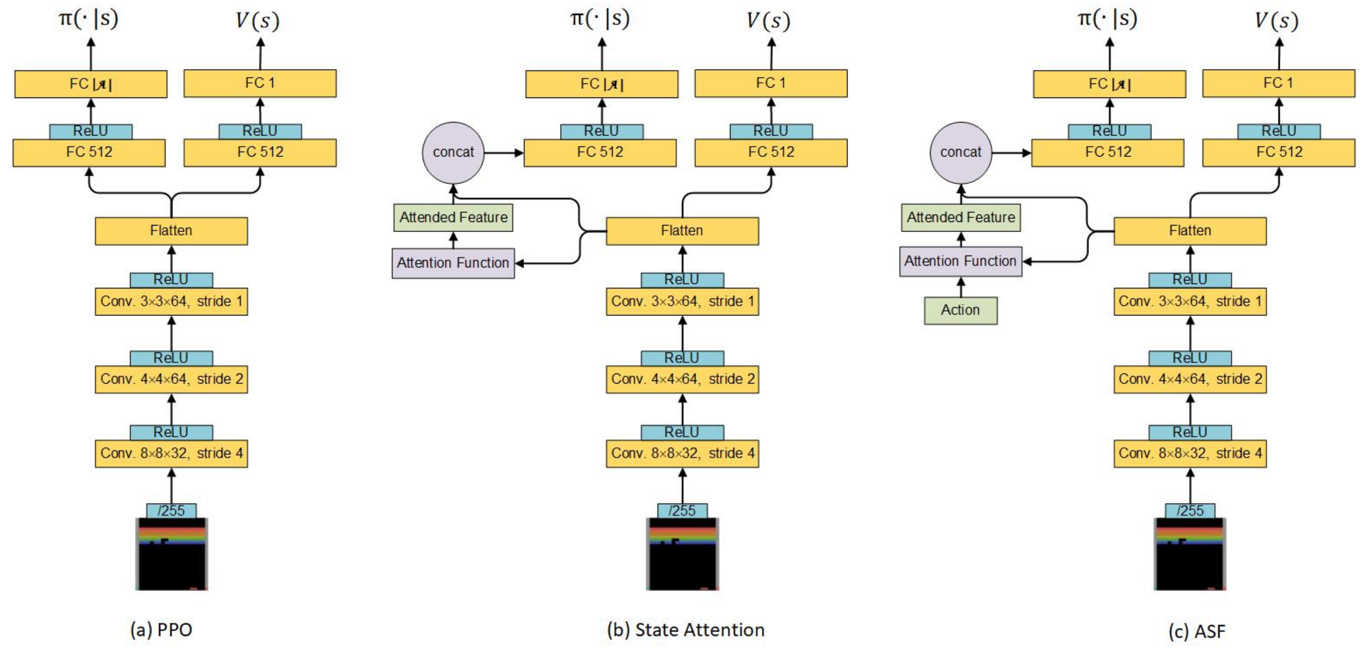

Fig. 4: Network architectures used in the Atari experiments: (a) network architecture of the vanilla PPO, (b) network architecture of PPO with State Attention, (c) network architecture of PPO with ASF.

random numbers of no-ops to introduce the stochasticity. To evaluate the overall performance, we introduce two metrics used in [4]:

1) The average reward per episode over the last 100 episodes of training.
2) The average reward per episode over the entire training period.

The first metric (denoted as *Final Score*) favors final performance and the second one (denoted as *Learning Speed*) favors fast learning. All metrics are computed across three trials with different seeds and Table I shows the number of games "won" by each method, and we judge a method to win a game if this method gets the highest score in this game.

The results show that ASF outperforms the others in most of the games both in *Final Score* and *Learning Speed*. In the total 58 games, our method won 32 (55.17%) games in *Final Score* and 28 (48.28%) games in *Learning Speed*. We summarize the *Final Score* results and the corresponding improvement ratios for all 58 games in Table III (in appendix). The average improvement ratios of ASF compared to PPO and State Attention are 12.57% and 27.63% respectively. The improvements of the ASF are more than 20% in 1/5 games and more than 10% in 1/3 games compared to the vanilla PPO. Through more detailed analysis, we can find that the improvement ratios of the ASF differs from game to game. In the games that the differences among the action–focus maps are small, e.g., Boxing, Breakout and Pong, the performance of our method is similar to the vanilla PPO. In contrast, in the more complex games with diverse objects such as MontezumaRevenge, Tutankham, and ElevatorAction, our method has more obvious improvements. This is consistent

TABLE I: Number of games "won" by each method.

	PPO	State Attention	ASF
(1) Final Score	15	11	**32**
(2) Learning Speed	18	12	**28**

with human intuition because the attention mechanism is usually more important for the complex problems in human decision-making process. Furthermore, although the State Attention achieves the highest scores in some games, the vanilla PPO "won" more games than the State Attention both in *Final Score* and *Learning Speed*, which indicates that an improper attention mechanism may bring negative effects for RL, and the improvements of ASF is because of the action-specific focuses rather than including the state into the attention. We implement our algorithm based on OpenAI Baselines [25] and the code is available online[1]. We also provide the values of the hyperparameters used in our experiments in Table II (in appendix) and the learning curves in Figure 5 (in appendix).

V. Conclusion and Future Work

We propose a novel top-down attention framework for RL, which can generate multiple action-specific features for each state and thus is able to focus on different sub-parts of the feature for different actions. With a more accurate feature, our method shows better performance and faster learning speed compared to the state-of-the-art algorithms. More importantly, we provide a new way to introduce attention mechanisms into RL. The key insight that the execution of each action relies on

[1]https://github.com/NeteaseFuxiRL/asf.git.

different information could inspire future research dealing with even more complex actions. In addition, our approach is also adapt to supervised learning in which the labels play a similar role as the actions in RL. In future work, firstly we intend to explore the performance of our method in the supervised learning problems. Secondly, we are interested in using ASF in off-poilcy RL algorithms [26] or imitation learning algorithms [27]. Lastly, the combination of hierarchical RL and ASF is also worth research. In theory, the goals generated by the high-level module in hierarchical RL can also guide the lower level module to adjust its focuses. On the other hands, we are also very interested in applying ASF in more complex environments such as autonomous driving [28] or multiplayer online games [29], in which humans will naturally use top-down attention to sovle these challenging problems.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, “Mastering the game of go with deep neural networks and tree search,” *nature*, vol. 529, no. 7587, p. 484, 2016.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, p. 529, 2015.

[3] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver, “Distributed prioritized experience replay,” *arXiv:1803.00933*, 2018.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv:1707.06347*, 2017.

[5] F. Katsuki and C. Constantinidis, “Bottom-up and top-down attention: different processes and overlapping neural systems,” *The Neuroscientist*, vol. 20, no. 5, pp. 509–521, 2014.

[6] J. Oh, V. Chockalingam, S. Singh, and H. Lee, “Control of memory, active perception, and action in minecraft,” *arXiv:1605.09128*, 2016.

[7] J. Choi, B.-J. Lee, and B.-T. Zhang, “Multi-focus attention network for efficient deep reinforcement learning,” *arXiv:1712.04603*, 2017.

[8] Y. Niv, R. Daniel, A. Geana, S. J. Gershman, Y. C. Leong, A. Radulescu, and R. C. Wilson, “Reinforcement learning in multidimensional environments relies on attention mechanisms,” *Journal of Neuroscience*, vol. 35, no. 21, pp. 8145–8157, 2015.

[9] T. J. Buschman and E. K. Miller, “Top-down versus bottom-up control of attention in the prefrontal and posterior parietal cortices,” *science*, vol. 315, no. 5820, pp. 1860–1862, 2007.

[10] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in *ICML*, 2015, pp. 2048–2057.

[11] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv:1409.0473*, 2014.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[13] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou *et al.*, “Hybrid computing using a neural network with dynamic external memory,” *Nature*, vol. 538, no. 7626, p. 471, 2016.

[14] V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, K. Tuyls, D. Reichert, T. Lillicrap, E. Lockhart *et al.*, “Relational deep reinforcement learning,” *arXiv:1806.01830*, 2018.

[15] J. Choi, Y. Guo, M. Moczulski, J. Oh, N. Wu, M. Norouzi, and H. Lee, “Contingency-aware exploration in reinforcement learning,” *arXiv:1811.01483*, 2018.

[16] I. Sorokin, A. Seleznev, M. Pavlov, A. Fedorov, and A. Ignateva, “Deep attention recurrent q-network,” *arXiv:1512.01693*, 2015.

[17] Y. Chen, C. Dong, P. Palanisamy, P. Mudalige, K. Muelling, and J. M. Dolan, “Attention-based hierarchical deep reinforcement learning for lane change behaviors in autonomous driving,” in *CVPR Workshops*, 2019.

[18] J. Zhang, S. A. Bargal, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff, “Top-down neural attention by excitation backprop,” *International Journal of Computer Vision*, vol. 126, no. 10, pp. 1084–1102, 2018.

[19] A. Sauer, N. Savinov, and A. Geiger, “Conditional affordance learning for driving in urban environments,” *arXiv:1806.06498*, 2018.

[20] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, “End-to-end driving via conditional imitation learning,” in *ICRA*. IEEE, 2018, pp. 1–9.

[21] Q. Wang, J. Zhang, S. Song, and Z. Zhang, “Attentional neural network: Feature selection using cognitive feedback,” in *Advances in Neural Information Processing Systems*, 2014, pp. 2033–2041.

[22] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, “Residual attention network for image classification,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 6450–6458.

[23] Y. Chandak, G. Theocharous, J. Kostas, S. Jordan, and P. S. Thomas, “Learning action representations for reinforcement learning,” *arXiv:1902.00183*, 2019.

[24] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” 2016.

[25] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, “Openai baselines,” https://github.com/openai/baselines, 2017.

[26] L. Li, “A perspective on off-policy evaluation in reinforcement learning,” *Frontiers of Computer Science*, vol. 13, no. 5, pp. 911–912, 2019.

[27] J. Ho and S. Ermon, “Generative adversarial imitation learning,” in *Advances in neural information processing systems*, 2016, pp. 4565–4573.

[28] S. Chen, Z. Jian, Y. Huang, Y. Chen, Z. Zhou, and N. Zheng, “Autonomous driving: cognitive construction and situation understanding,” *Science China Information Sciences*, vol. 62, p. 81101, 2019.

[29] H. Jia, C. Ren, Y. Hu, Y. Chen, T. Lv, C. Fan, H. Tang, and J. Hao, “Mastering basketball with deep reinforcement learning: An integrated curriculum training approach,” in *AAMAS*, 2020, pp. 1872–1874.

Appendix

A. Hyperparameters

TABLE II: Hyperparameters for Atari experiments, where α is linearly annealed from 1 to 0 over the course of learning.

Parameter	Value
Horizon (T)	128
Learning rate	$2.5 \times 10^{-4} \times \alpha$
Num. epochs	4
Entropy Regularizer(β)	0.01
GAE parameter	0.95
Number of actors	8
Discount (γ)	0.99
FrameStack	4
Reward Clipping	[-1,1]
Frame Skipping	No
Env Randomness	Random No-ops

B. Performance on Atari games.

Table III shows the mean final scores (the last 100 episodes) of the three methods on all Atari games, and Figure 5 shows the learning curves. Each method is trained with three different seeds and the curves are smoothed to make the plots more readable. Each point in Figure 5 is the average reward of the last 100 episodes and the initial parts of the curves in *ElevatorAction* are missed because the episode of this game is very long and it costs lots of timesteps to collect the initial 100 episodes.

TABLE III: Mean final scores (last 100 episodes) of PPO, StateAttention and ASF on Atari games after 20M timesteps across three trials with different seeds. The column named “Improvement-PPO” represents the improvement ratio of ASF compared to PPO, and the column named “Improvement-SA” represents the improvement ratio of ASF compared to State Attention.

Game	PPO	State Attention	ASF	Improvement-PPO	Improvement-SA
AirRaid	7417	5042.1	**8318.3**	12.15%	64.98%
Alien	2159.8	2235.7	**2309.1**	6.91%	3.28%
Amidar	**1002**	989.5	917.7	-8.41%	-7.26%
Assault	6033.3	6049.5	**6240.8**	3.44%	3.16%
Asterix	6556.3	**6893.5**	6121.2	-6.64%	-11.20%
Asteroids	2033.5	1726.4	**2135.6**	5.02%	23.70%
Atlantis	2818823	2790869.7	**2958376.7**	4.95%	6.00%
BankHeist	1252.2	**1261.4**	1252.2	0.00%	-0.73%
BattleZone	**23493.3**	20983.3	18456.7	-21.44%	-12.04%
BeamRider	3918.2	4139.2	**4313.1**	10.08%	4.20%
Berzerk	1094.6	1282.9	**1360**	24.25%	6.01%
Bowling	45.3	**52.4**	50.2	10.82%	-4.20%
Boxing	95.8	**96.6**	96.1	0.31%	-0.52%
Breakout	326.8	287.9	**333.3**	1.99%	15.77%
Carnival	3658.9	4206.7	**4470.6**	22.18%	6.27%
Centipede	4097.4	**4278**	4095.5	-0.05%	-4.27%
CrazyClimber	**119256**	112149.3	114108	-4.32%	1.75%
DemonAttack	**37907**	20958.5	17283.4	-54.41%	-17.54%
DoubleDunk	-5.3	-3.9	**-2**	62.26%	48.72%
ElevatorAction	7854.3	7018	**16914.3**	115.35%	141.01%
Enduro	1095.3	1065.9	**1122.9**	2.52%	5.35%
FishingDerby	23.9	25.4	**25.6**	7.11%	0.79%
Freeway	33.2	**33.7**	33.5	0.90%	-0.59%
Frostbite	307.6	**629.6**	293.6	-4.55%	-53.37%
Gopher	**6527.6**	5547.7	6309.8	-3.34%	13.74%
Gravitar	**1084.5**	1035	1043.3	-3.80%	0.80%
IceHockey	-4.9	-4.2	**-3.9**	20.41%	7.14%
Jamesbond	895.5	823	**908**	1.40%	10.33%
JourneyEscape	-925	-716	**-654.7**	29.22%	8.56%
Kangaroo	**9516**	5836	8358.7	-12.16%	43.23%
Krull	**8737.6**	8550.3	8513.5	-2.56%	-0.43%
KungFuMaster	28321.7	27777	**31261**	10.38%	12.54%
MontezumaRevenge	13.7	5.7	**62.7**	357.66%	1000.00%
MsPacman	2766.4	2235.6	**2786.6**	0.73%	24.65%
NameThisGame	6267.6	6469.8	**6866.7**	9.56%	6.13%
Phoenix	13217.4	13820.2	**16582.8**	25.46%	19.99%
Pitfall	-0.8	-0.6	**0**	100.00%	100.00%
Pong	20.6	20.5	**20.9**	1.46%	1.95%
Pooyan	3301	**3421.7**	3268.8	-0.98%	-4.47%
PrivateEye	89.4	66.9	**100**	11.86%	49.48%
Qbert	18506.4	20237.8	**20274.7**	9.56%	0.18%
Riverraid	11555.4	8594.4	**11949.8**	3.41%	39.04%
RoadRunner	40458.3	48938	**49561**	22.50%	1.27%
Robotank	**18.3**	18.3	16.7	-8.74%	-8.74%
Seaquest	1694	1782.3	**1927.7**	13.80%	8.16%
Skiing	-22979.6	**-14647.1**	-15548.7	32.34%	-6.16%
Solaris	2247.2	**2395.2**	2310.3	2.81%	-3.54%
SpaceInvaders	**1601.2**	1493.3	1596.2	-0.31%	6.89%
StarGunner	57001	56776.3	**58680.7**	2.95%	3.35%
Tennis	**-6.5**	-8.6	-9.7	-49.23%	-12.79%
TimePilot	6681.3	7275.7	**7537**	12.81%	3.59%
Tutankham	207.8	221.5	**255.2**	22.81%	15.21%
UpNDown	**205443.2**	191766.9	179574.9	-12.59%	-6.36%
Venture	**149**	0	0	-100.00%	0.00%
VideoPinball	**78672.5**	53752.7	78666.8	-0.01%	46.35%
WizardOfWor	**6617.7**	6248.3	6095.7	-7.89%	-2.44%
YarsRevenge	15850.1	**23078.7**	22092.8	39.39%	-4.27%
Zaxxon	10305.7	10307.7	**11301.7**	9.66%	9.64%
Average Improvement Ratio	-	-	-	12.57%	27.63%

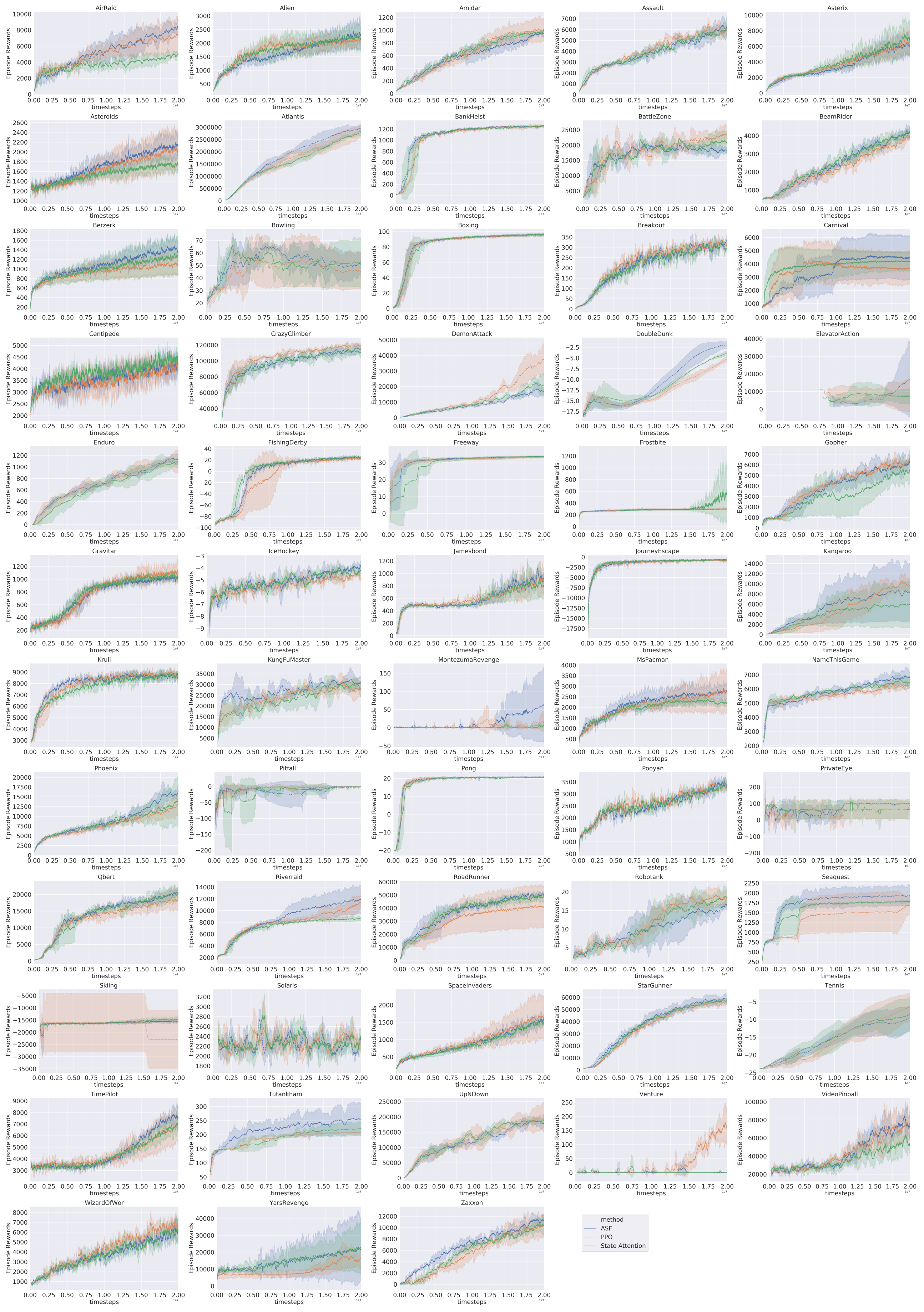

Fig. 5: Learning curves of PPO, State Attention and ASF on Atari games after 20M timesteps. The error bands are computed based on three trials with different seeds.