Incorporating Explanations to Balance the Exploration and Exploitation of Deep Reinforcement Learning

Xinzhi Wang(✉), Yang Liu, Yudong Chang, Chao Jiang, and Qingjie Zhang

School of Computer Engineering and Science, Shanghai University, Shanghai, China
{wxz2017,lliuyang,cydshu,superjiang,ivanzhang}@shu.edu.cn

Abstract. Discovering efficient exploration strategies is a central challenge in reinforcement learning (RL). Deep reinforcement learning (DRL) methods proposed in recent years have mainly focused on improving the generalization of models while ignoring models' explanation. In this study, an embedding explanation for the advantage actor-critic algorithm (EEA2C) is proposed to balance the relationship between exploration and exploitation for DRL models. Specifically, the proposed algorithm explains agent's actions before employing explanation to guide exploration. A fusion strategy is then designed to retain information that is helpful for exploration from experience. Based on the results of the fusion strategy, a variational autoencoder (VAE) is designed to encode the task-related explanation into a probabilistic latent representation. The latent representation of the VAE is finally incorporated into the agent's policy as prior knowledge. Experimental results for six Atari environments show that the proposed method improves the agent's exploratory capabilities with explainable knowledge.

Keywords: Deep reinforcement learning · Explainable AI · Explanation fusion · Variational auto encoder

1 Introduction

Deep reinforcement learning (DRL) has been widely studied and applied in various fields such as gaming, autonomous driving, and robotics [1,14,22]. However, the evaluation-based nature of DRL makes exploitation necessary, and continuous exploration is required to improve decision-making. Therefore, the exploration-exploitation dilemma is a central challenge in DRL [19]. Exploration refers to trying new actions in unknown environments, whereas exploitation refers to acting based on historical experience. The agent must take action based on exploration and exploitation to obtain high cumulative rewards. DRL methods proposed in recent years have mainly focused on improving the generalization of models while ignoring the exploration-exploitation dilemma, and balancing the relationship between exploration and exploitation is difficult.

G. Memmi et al. (Eds.): KSEM 2022, LNAI 13369, pp. 200–211, 2022.
https://doi.org/10.1007/978-3-031-10986-7_16

Many approaches have been proposed to solve the exploration-exploitation dilemma, such as noise-based exploration strategies [7,16] and intrinsic reward-based exploration methods [2,12,20]. However, the uncertainty caused by existing noise-based exploration strategies is task-independent, which reduces exploration efficiency. The reward function in intrinsic reward-based methods is artificially designed, which does not ensure that it is suitable for complex environments, and the above methods cannot achieve effective exploration.

Moreover, the weak explainability of DRL agents makes it impossible to thoroughly understand and analyze data, models, and applications [5,23]. This causes disparities between the realistic performance of an agent and expectations. One approach to explaining the decision-making of a network is visual explanation, which has been studied in the field of computer vision [9,18]. A visual explanation analyzes the network output factors by generating a heat map, which highlights the critical regions in an input image. Visual explanation methods have also been applied to explain an agent's decision-making process [11,24]. However, these explanatory methods merely analyze how an agent makes decisions and fail to utilize the explained information to optimize the agent.

The agent assesses how well it learns about an environment task based on the uncertainty of network predictions [4,12,15]. The uncertainty of traditional exploration strategies is unrelated to environmental tasks, which leads to inefficient exploration. Recent exploratory approaches suggest that task-related information can guide an agent's exploration, such as discovering bottleneck states [10], learning feature spaces [8], and VARIBAD [25]. The activation maps generated by the visual explanation contain task targets, which can be transformed into task-related uncertainty to guide the agent's exploration.

In this study, an embedding explanation for the advantage actor-critic algorithm (EEA2C) is proposed based on the advantage actor-critic (A2C) architecture. The proposed method provides a novel exploration strategy that employs task-related explanations. EEA2C first uses a gradient-based explanation method to generate activation maps. Meanwhile, a fusion strategy is designed to retain helpful information for exploration from experience. Second, a variational auto-encoder (VAE) is designed to project the fused experience into a probabilistic latent representation, which represents the inherent structure of the task-related environment. Finally, the latent variables are fed to the actor-critic network for training. The agent's policy is conditioned on both the environmental state and latent variables. The contributions of this study are as follows.

- We propose a reinforcement-learning algorithm EEA2C with a novel exploration strategy. The algorithm overcomes the inefficiency of exploration by employing VAE techniques to encode the experience of the fusion explanation.
- The proposed algorithm customizes an explanation module and a fusion strategy, compensating for traditional explanation methods that cannot optimize the agent.
- Experiments are conducted in the ATARI environment to evaluate the performance and advantages of the proposed algorithm. The experimental results prove that the proposed algorithm promotes the agent's policy learning abilities and increases explainability.

2 Preliminaries

This section introduces the concepts of reinforcement learning and variational inference and provides definitions of commonly used symbols.

2.1 Reinforcement Learning

Standard reinforcement learning based on the Markov decision process (MDP) is represented by a five-tuple of $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, and $\mathcal{P}$ is the unknown state-transition probability function. r is the immediate reward associated with taking action $a \in \mathcal{A}$ while transitioning from state $s_t \in \mathcal{S}$ to state $s_{t+1} \in \mathcal{S}$. $\gamma \in [0, 1]$ is the discount factor that represents the trade-off between maximizing immediate and future returns. The goal of the agent is to identify a policy π that maximizes its expected reward, where the cumulative return at each time step is $R_t = \sum_{n=t}^{\infty} \gamma^{n-t} r_n$.

2.2 Variational Inference

The goal of variational inference is to approximate the conditional probabilities of latent variables considering the observed variables. Optimization methods can be used to solve this problem. Suppose $x = x_{1:n}$ denotes a set of observable datasets, and $z = z_{1:m}$ is a set of latent variables. Let $p(z, x)$ denote the joint probability, $p(z|x)$ be the conditional probability, and $p(x)$ be the evidence. The conditional probability, $p(z|x) = \frac{p(z,x)}{p(x)}$, can be solved using Bayesian inference. However, it is difficult to directly compute $p(x)$. Therefore, the probability density function $q(z)$ is determined using variational inference to approximate $p^*(z|x)$. $q*(z) = \arg\min KL(q(z)||p(z|x))$ is optimized to obtain the optimal $q*(z)$, where $q(z) \in Q$ is the approximate distribution and $p(z|x)$ is the required posterior probability distribution. The KL divergence can be expressed as

$$KL(q(z)||p(z|x)) = \mathbb{E}[\log q(z)] - \mathbb{E}[\log p(z, x)] + \log p(x), \tag{1}$$

where $\log p(x)$ is a constant and the KL divergence is greater than zero. The final goal of variational inference is as follows:

$$\begin{aligned} \log p(x) &\geq \log p(z|x) - \log q(z) \\ &= ELBO, \end{aligned} \tag{2}$$

which is called the evidence lower-bound objective (ELBO).

3 Proposed Method

This section describes the architecture of the EEA2C and the workflow of the proposed algorithm.

3.1 Network Architecture

The trained network is shown in Fig. 1. The proposed model includes four main modules: actor, critic, explanation, and inference. The actor and critic share the state encoder and internal network used to train the policy. The explanation module generates a task-related activation map using an activation gradient mapping approach. The activation maps are fused to the original states using a fusion strategy. The inference module extracts the latent structure of the fused information by training a variational autoencoder (VAE). The inference module encodes a fused experience to derive a latent variable distribution. The inferred latent variables are employed to train the actor and critic modules as prior knowledge.

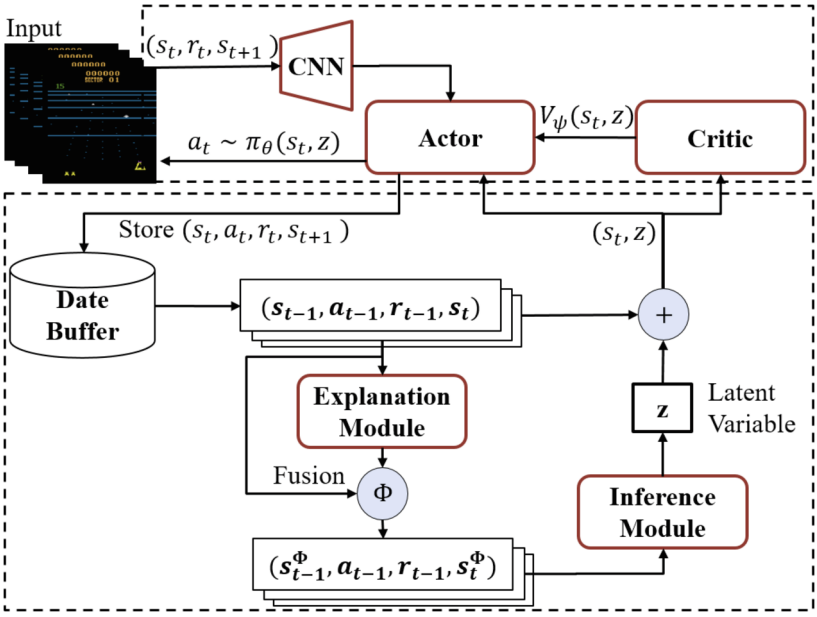


Fig. 1. Overview of the proposed model EEA2C.

Each actor, critic, and inference module is a multilayer neural network. In the inference module, let q_ω denote the encoder with parameters ω. p_ν denotes the decoder with parameter ν. Let z denote the latent variable and $z \sim q_\omega$. Let π_θ denote an actor with parameters θ. Let V_ψ denote the critic with parameter ψ, and let D denote the experience data buffer. The module outputs can either be concatenated or added to obtain the other module's input. This algorithm is introduced in detail in the following sections.

3.2 Explanation of Actions with Activation Maps

During each iteration, the agent employs policy π_θ to interact with the environment and obtain a series of experiences, which are stored in the data buffer D. Then, the experience sample $\tau_{:t}$ is sampled from D and denoted as

$$\tau_{:t} = \{s_0, a_0, r_0, ..., s_m, ..., a_{t-1}, r_{t-1}, s_t\}. \tag{3}$$

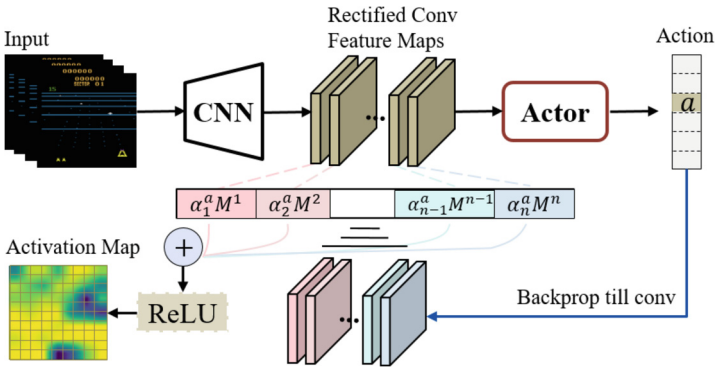


Fig. 2. Explanation module: activation map produced by the explanation module based on the state-action pair.

To obtain the activation map of state s_m in $\tau_{:t}$, an explanation module is designed (see Fig. 2) by modifying the Grad-Cam approach [18]. Each state s_m in $\tau_{:t}$ is convolved and computed by the actor to obtain a_m. Subsequently, the gradient of the score for action a_m is computed with respect to the feature map activations M^k of the last convolutional layer. These gradients flowing back are averaged over the width and height dimensions (indexed by i and j, respectively) to obtain the neuron importance weights $\alpha_k^{a_m}$:

$$\alpha_k^{a_m} = \frac{1}{T} \sum_i \sum_j \frac{\partial \pi^{a_m}}{\partial M_{ij}^k}, \tag{4}$$

where $a_k^{a_m}$ is the weight of action a in the k-th channel. π^{a_m} is the probability that the agent selects action a_m and T denotes the normalization factor used to ensure that the sum of the gradient weights of each channel is 1 (for example, $\sum_k \alpha_k^{a_m} = 1$). The weight $a_k^{a_m}$ represents a partial linearization of the agent downstream from M and captures the importance of feature map k for action a_m. We perform a weighted combination of forward activation maps, followed by a $ReLU$, to obtain the final activation map e_m, which can be denoted as

$$e_m = ReLU(\sum_k \alpha_k^{a_m} M^k), \tag{5}$$

where the $ReLU$ operation is used to retain regions that have a positive effect on action a_m. The activation map e_m takes values in the range of $[0, 1]$, indicating the relevance of the region in s_m to a_m. Each state s_m in $\tau_{:t}$ is explained sequentially, and the explanation module outputs the set of activation maps $e_{:t}$:

$$e_{:t} = \{e_0, e_1, ..., e_m, ..., e_t\}. \tag{6}$$

3.3 Fusion Activation Maps and States

Based on the above methods, EEA2C can obtain activation maps $e_{:t}$ of $\tau_{:t}$ from the explanation module. A straightforward method that may be used to focus an agent on the task-related regions is to multiply e_m and s_m. However, when multiplying e_m and s_m, the zero values in the activation maps lead to incomplete state information. This disrupts the agent's training, makes policy learning unstable, and fails to enable the agent to focus on task-related regions. To overcome this problem, we propose a state fusion strategy in which the state in $\tau_{:t}$ is reformulated as the weighted sum of the original state s_m and noisy background image g_m. The original state s_m is replaced with s_m^{Φ} via the fusion operation $\Phi(s_m, e_m)$, which is formulated as follows:

$$s_m^{\Phi} = \Phi(s_m, e_m) = s_m \cdot e_m + g_m \cdot (1 - e_m), \tag{7}$$

where g_m is a blurred image obtained by applying a Gaussian blur filter to the original image s_m [6]. The representation s_m^{Φ} of s_m is located in the natural state-space manifold. This formulation not only creates state s_m^{Φ} that matches the inner feature representation at s_m but also preserves the object localization information in e_m. The weight vector $e_m \in [0, 1]$ denotes the significance of each region that contributes to the state representation s_m. The experience sample $\tau_{:t}$ is updated to $\tau_{:t}^{\Phi}$ after fusing the activation map e_m:

$$\tau_{:t}^{\Phi} = \{s_0^{\Phi}, a_0, r_0, ..., s_m^{\Phi}, ..., a_{t-1}, r_{t-1}, s_t^{\Phi}\}. \tag{8}$$

3.4 Encoding the Fused State with Variational Inference

To incorporate the fused experience $\tau_{:t}^{\Phi}$ into the training process of the policy, an additional probabilistic model is constructed to generate prior knowledge of the actor-critic network. The latent variables generated by the probabilistic model serve as task-related uncertainties to motivate exploration. Inferring the latent distribution $p(z|\tau_{:t}^{\Phi})$ is difficult owing to the intractable integrals involved. Inspired by VAEs, an inference module is designed to approximate the posterior distribution with a variational distribution $q(z|\tau_{:t}^{\Phi})$. The inference module consists of an encoder q_ω with parameter ω and a decoder p_ν with parameter ν. The encoder q_ω takes the experience $\tau_{:t}^{\Phi}$ and projects it into latent space Z. This latent space captures the factors of variation within a fused experience. The latent variable z ($z \in Z$) is subject to a normal distribution, and its parameters (μ and σ) are predicted directly by $q_\omega(z|\tau_{:t}^{\Phi})$, as shown in Fig. 3.

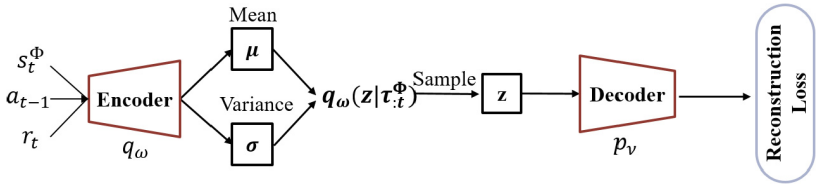


Fig. 3. Inference module: a trajectory of the fused explanation information is processed using an encoder to generate the posterior of the task embedding process. The posterior is trained with a decoder that predicts the past and future states and rewards from the fused state.

The inference module approximates the conditional probability of the latent variables in the posterior sample $\tau_{:t}^{\Phi}$. As with the target function of VAE, the optimization objective of the inference module is to maximize $ELBO$:

$$\begin{aligned}\mathcal{L}(\omega,\nu) &= \mathbb{E}_{z\sim q_\omega(z|\tau_{:t}^{\Phi})}[\log p_\nu(\tau_{:t}^{\Phi}|z)] - KL(q_\omega(z|\tau_{:t}^{\Phi})||p_\nu(z)) \\ &= ELBO(\omega,\nu),\end{aligned} \tag{9}$$

where the first term $\log p_\nu(\tau_{:t}^{\Phi}|z)$ denotes the reconstruction loss. The second term $KL(q_\omega(z|\tau_{:t}^{\Phi})||p_\nu(z))$ encourages the approximate posterior q_ω to remain close to the prior over the embedding $p_\nu(z)$. $p_\nu(z)$ is the prior probability of the latent variable z using a standard normal distribution, $p_\nu(z) = N(0, I)$.

Owing to the introduction of the task-related latent variable z, the optimization objective of the actor-critic network becomes

$$\mathcal{L}(\theta) = -log\pi_\theta(a_t|s_t,z)(R_t^n - V_\psi(s_t,z)) - \alpha\mathcal{H}_t(\pi_\theta), \tag{10}$$

$$\mathcal{L}(\psi) = \frac{1}{2}(V_\psi(s_t,z) - R_t^n)^2, \tag{11}$$

where $R_t = \sum_{i=0}^{n-1}\gamma^i r_{t+i} + \gamma^n V_\psi(s_{t+n})$ is the n-step bootstrapped return and α is the weight of the standard entropy regularization loss term. Our overall objective is to minimize

$$\mathcal{L}(\theta,\psi,\omega,\nu) = \mathcal{L}(\theta) + \beta\mathcal{L}(\psi) - \lambda\mathcal{L}(\omega,\nu), \tag{12}$$

where β is the weight for the critic loss and λ is the weight inference module. The actor-critic network and the inference module are trained using different data buffers: the actor-critic network is only trained with the most recent data because we use on-policy algorithms in our experiments; for the inference module, we maintain a separate, larger buffer of observed trajectories.

4 Experiments and Results

4.1 Environment and Experimental Settings

EEA2C was evaluated for six control tasks in OpenAI Gym [3]. In all tasks, the reward for each step of the agent was limited to $[0, 1]$ during training. The

algorithm uses 16 parallel actors to collect the agent's experience with a five-step rollout, yielding a minibatch of size 80 for on-policy transitions. The last four observation frames are stacked as inputs, each of which is resized to 84×84 and converted to grayscale in [13]. The episode is set as the end of the game instead of losing a life. Each episode is initialized with a random seed of no-ops [14].

The method proposed in this study is based on the A2C framework [13]. The actor and critic networks use the same structure, employing a multi-layer perceptron (MLP) with two implicit layers 64×64. The inference module uses a simpler network structure that outputs the mean and variance of the latent variables. The experimental results are obtained by averaging five random seeds, each with 2×10^7 frames.

4.2 Comparisons with Benchmark Algorithms

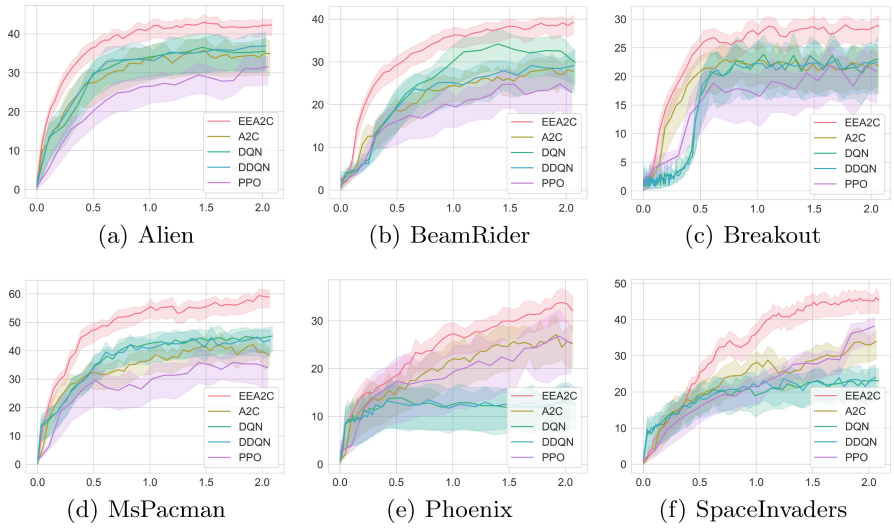


Fig. 4. Learning curves: expected return vs. number of training frames.

The DQN [14], DDQN [21], A2C [13] (with both separate and shared actor-critic networks), and PPO [17] algorithms were employed as benchmark algorithms for comparison. The learning speed of each algorithm was measured over a limited number of game frames. Figure 4 illustrates the learning curves of all algorithms. The learning curves were obtained by averaging the results over five random seeds. The experiments showed that EEA2C outperformed all benchmark algorithms in terms of the learning speed and improved the cumulative score for all exploration tasks.

Figure 4 shows the mean reward for the proposed method and the other benchmark algorithms, and the shaded areas represent the variance of the mean

reward for each seed. From the above experimental results, the learning speed and target rewards of the EEA2C algorithm outperformed those of the other algorithms in all control tasks. The rewards that the EEA2C algorithm reaches require 0.2×10^7 time steps, whereas the benchmark algorithms require 1×10^7 time steps to reach these rewards. As the number of training frames increased, superior rewards to the benchmark algorithms were eventually achieved. In each environment, the shaded area of EEA2C is smaller than those of the other algorithms, indicating that EEA2C performs more consistently across multiple seeds.

From Fig. 4, the prior experience sample is shown to be insufficient at the beginning of training, and the target in the activation map is scattered. In this case, the confidence interval of the latent variables is large, which leads to an unstable distribution. Therefore, the agent was more inclined to explore. As more data are collected, the high-value regions in the activation map become concentrated, and the distribution of the latent variables tends to stabilize. Meanwhile, the agent begins to exploit and the cumulative rewards tend to stabilize.

Table 1. Evaluation scores over 50 recent episodes on average. The best scores are in bold.

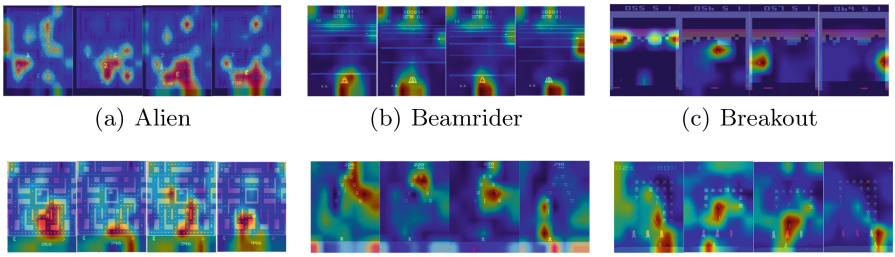
Environment	Random	DQN	DDQN	PPO	A2C	EEA2C
Alien	240	2391	2041	1970	2091	**3161**
BeamRider	264	3627	3172	2750	3164	**4356**
Breakout	3	518	520	417	435	**624**
MsPacman	150	3180	2960	2350	2880	**4830**
Phoenix	440	10840	12250	20840	22530	**28530**
SpaceInvaders	120	3929	3672	4855	4673	**7673**

Table 1 shows that EEA2C achieves a better performance than the benchmark with a large margin in six tasks. Compared with the best score of the benchmark algorithm, EEA2C achieved a 32.2% improvement on Alien, 20.1% improvement on BeamRider, 20.0% improvement on Breakout, 51.8% improvement on MsPacman, 26.6% improvement on Phoenix, and 58.0% improvement on SpaceInvaders. EEA2C achieved a better average reward than the benchmark when trained with limited frames. Therefore, further performance improvements can be expected when EEA2C uses more training frames or a large-scale distributed computing platform.

4.3 Analysis of Explainability

This section analyzes how the activation maps generated by the explanation module are associated with the action of the agent. The activation map can be overlayed onto the original state to indicate evidence of the agent's action. As shown in Fig. 5, for the action predicted by the agent, a more-intensely red-colored area indicates a more significant contribution, which can be used as the

main basis for determining the predicted action. The activation map enables an understanding of the regions that the agent focuses on when taking action, which improves the explainability of the algorithm.



(a) Alien (b) Beamrider (c) Breakout

(d) MsPacman (e) Phoenix (f) SpaceInvaders

Fig. 5. Regions related to the agent's decision are shown by adding the activation maps to the image, where red regions correspond to high scores while blue corresponds to low scores. (Color figure online)

The target in the ATARI environment appears periodically at a specific location. Based on Fig. 5, we can conclude that the agent's focus becomes increasingly concentrated. The agent gradually learns to focus on task-related targets and takes corresponding actions rather than reacting irrelevantly to state-specific patterns. For example, in Alien 5(a) and SpaceInvader 5(f), the agent appropriately reacts to the emergence of the enemy. The agent notices the enemy, moves towards it, fires at it, and then turns away. When there are no objects in the image, the agent moves as little as possible to avoid mistakes. The agent learns a path to the target object among the worlds by forwarding planning elements and lattice classes. Figure 5(d) shows an example of MsPacman. The agent scans the route to ensure that there are no enemies or ghosts ahead. When the food appears on the screen, the agent generates a path to this food.

5 Conclusion

This study proposes EEA2C to balance the relationship between exploration and exploitation. Unlike previous research, this algorithm explains the agent's actions and employs the explanation to guide the agent's exploration. Simultaneously, an inference module was designed to encode the explanation into a probabilistic latent representation, which can be incorporated into the policy as task-related uncertainty. Experiments were conducted in the ATARI environment to evaluate the performance and advantages of the proposed algorithm. The experimental results showed that the EEA2C outperformed existing methods in terms of the achieved reward during a single episode. The uncertainty provided by the latent variables promoted policy learning. Furthermore, the activation maps generated by the explanation module augmented the agent's explainability. Task-related

uncertainty can be obtained from explanation or provided by external expert experience. In future work, we will consider other ways of capturing task-related uncertainties to guide agents in more efficient exploration.

Acknowledgements. This work is sponsored by Shanghai Sailing Program (NO. 20YF1413800).

References

1. Baker, B., et al.: Emergent tool use from multi-agent autocurricula. In: 8th International Conference on Learning Representations. OpenReview.net (2020)
2. Bellemare, M.G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., Munos, R.: Unifying count-based exploration and intrinsic motivation. In: Advances in Neural Information Processing Systems, vol. 29, pp. 1471–1479 (2016)
3. Brockman, G., et al.: OpenAI gym. arXiv preprint arXiv:1606.01540 (2021)
4. Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., Efros, A.A.: Large-scale study of curiosity-driven learning. arXiv preprint arXiv:1808.04355 (2018)
5. Chakraborty, S., et al.: Interpretability of deep learning models: a survey of results. In: Smartworld, pp. 1–6 (2017)
6. Fong, R.C., Vedaldi, A.: Interpretable explanations of black boxes by meaningful perturbation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3429–3437 (2017)
7. Fortunato, M., et al.: Noisy networks for exploration. arXiv preprint arXiv:1706.10295 (2017)
8. François-Lavet, V., Bengio, Y., Precup, D., Pineau, J.: Combined reinforcement learning via abstract representations. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 3582–3589 (2019)
9. Fukui, H., Hirakawa, T., Yamashita, T., Fujiyoshi, H.: Attention branch network: learning of attention mechanism for visual explanation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10705–10714 (2019)
10. Goyal, A., et al.: InfoBot: transfer and exploration via the information bottleneck. arXiv preprint arXiv:1901.10902 (2019)
11. Greydanus, S., Koul, A., Dodge, J., Fern, A.: Visualizing and understanding Atari agents. In: International Conference on Machine Learning, pp. 1792–1801 (2018)
12. Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., Abbeel, P.: VIME: variational information maximizing exploration. In: Advances in Neural Information Processing Systems, vol. 29 (2016)
13. Mnih, V., et al.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning, pp. 1928–1937. PMLR (2016)
14. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
15. Pathak, D., Agrawal, P., Efros, A.A., Darrell, T.: Curiosity-driven exploration by self-supervised prediction. In: International Conference on Machine Learning, pp. 2778–2787. PMLR (2017)
16. Plappert, M., et al.: Parameter space noise for exploration. arXiv preprint arXiv:1706.01905 (2017)
17. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)

18. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-CAM: visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 618–626 (2017)
19. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2018)
20. Tang, H., et al.: Exploration: a study of count-based exploration for deep reinforcement learning. In: Advances in Neural Information Processing Systems, vol. 30, pp. 2753–2762 (2017)
21. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double Q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence (2016)
22. Vinyals, O., et al.: Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature **575**(7782), 350–354 (2019)
23. Wang, X., Sugumaran, V., Zhang, H., Xu, Z.: A capability assessment model for emergency management organizations. Inf. Syst. Front. **20**(4), 653–667 (2018). https://doi.org/10.1007/s10796-017-9786-7
24. Wang, X., Yuan, S., Zhang, H., Lewis, M., Sycara, K.P.: Verbal explanations for deep reinforcement learning neural networks with attention on extracted features. In: 28th IEEE International Conference on Robot and Human Interactive Communication, pp. 1–7. IEEE (2019)
25. Zintgraf, L.M., et al.: VariBAD: variational Bayes-adaptive deep RL via meta-learning. J. Mach. Learn. Res. **22**, 289:1–289:39 (2021)