

Leveraging Reward Consistency for Interpretable Feature Discovery in Reinforcement Learning

Qisen Yang, Huanqian Wang, Mukun Tong, Wenjie Shi, Gao Huang, *Member, IEEE*,
and Shiji Song, *Senior Member, IEEE*

Abstract—The black-box nature of deep reinforcement learning (RL) hinders them from real-world applications. Therefore, interpreting and explaining RL agents have been active research topics in recent years. Existing methods for post-hoc explanations usually adopt the action matching principle to enable an easy understanding of vision-based RL agents. In this paper, it is argued that the commonly used action matching principle is more like an explanation of deep neural networks (DNNs) than the interpretation of RL agents. It may lead to irrelevant or misplaced feature attribution when different DNNs’ outputs lead to the same rewards or different rewards result from the same outputs. Therefore, we propose to consider rewards, the essential objective of RL agents, as the essential objective of interpreting RL agents as well. To ensure reward consistency during interpretable feature discovery, a novel framework (RL interpreting RL, denoted as RL-in-RL) is proposed to solve the gradient disconnection from actions to rewards. We verify and evaluate our method on the Atari 2600 games as well as Duckietown, a challenging self-driving car simulator environment. The results show that our method manages to keep reward (or return) consistency and achieves high-quality feature attribution. Further, a series of analytical experiments validate our assumption of the action matching principle’s limitations.

Index Terms—Attention map, deep reinforcement learning, explainability, feature attribution, interpretability.

I. INTRODUCTION

REINFORCEMENT learning (RL) learns to solve sequential decision-making problems in an interactive environment. Combined with the powerful approximation capability of deep neural networks (DNNs), deep RL has soared and achieved impressive successes in various fields such as video games [1] and robotics [2], [3]. However, the black box nature of DNNs makes deep RL harder to understand, while trust and reliability are critical concerns in real-world applications, especially for high-stake scenarios like autonomous driving [4], [5] and medical care [6]. Hence, exploring the underlying decision-making process and its interpretation (or explanation)¹ will greatly contribute to more applicable RL.

This work was supported in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2020B1111500002, in part by the National Natural Science Foundation of China under Grants 62022048 and 62276150, and THU-Bosch JCML. (*Corresponding author: Shiji Song.*)

Qisen Yang, Huanqian Wang, Mukun Tong, Gao Huang, and Shiji Song are with the Department of Automation, Tsinghua University, Beijing 100084, China. Email: {yangqs19, wang-hq23, tmk19}@mails.tsinghua.edu.cn, {gaohuang, shijis}@tsinghua.edu.cn. Wenjie Shi is with Meituan, Beijing 100102, China. Email: shiwenjie05@meituan.com.

¹Since no consensus about the nomenclature has been reached yet [6], we use interpretation and explanation as synonyms in this paper.

As far as we know, plenty of methods attempt to explain vision-based RL agents from the perspective of feature attribution [7]–[9]. For example, the embedding-based methods [10] analyze high-dimensional feature maps, and the attention-based methods [11] learn saliency maps in the agent’s training process. However, the former approach may be intuitively complex and the latter one does not apply to the post-hoc explanation. Other methods that manage to discover the interpretable features in an easy-to-understand way without revising the DNN’s architecture (such as the perturbation-based [12], gradient-based [13], and supervision-based [14] methods) generally build on the same action matching principle. Specifically, they usually attribute features of states by ensuring action consistency when explaining a pretrained RL agent. However, we have some concerns about this commonly used principle since RL agents’ intrinsic goal is to achieve the maximum accumulated rewards instead of action cloning.

Considering the situation when agents take slightly different actions but intrinsically aim at the same behavior to get a certain reward, the attentive features discovered by action matching can be redundant or even misleading. For example, as illustrated in Figure 1(a), the agent in a driving task gets positive rewards when successfully avoiding oncoming cars. In the state s_t , the agent (the green car) needs to avoid collision with the oncoming blue car. However, the actions a_t “moving left by 0.12” in Figure 1(a.1) and “moving left by 0.15” in Figure 1(a.2) would have obvious differences, while their essence is the same “avoidance” behavior. In this case, the behavior is truly represented by the reward instead of the action. Action matching may learn some redundant attention to keep the identical speeds or angles in the action, which can be seen as a kind of “overfitting”.

Considering another situation when agents take the same action but get different rewards, the action matching methods may fail to discover the truly interpretable features. For example, as illustrated in Figure 1(b), the agents take the same action “moving right” under the same state. Then the agent in the catching-ball task gets a positive reward, while the agent in the avoiding-ball task gets a negative reward. Intuitively, the agent in Figure 1(b.1) may take the action because it intends to catch the middle ball, and the agent in Figure 1(b.1) may move right because it attempts to avoid the left ball. However, the action matching methods would learn the same attentive features because of the same action, and the task-related explanation may not be truly discovered.

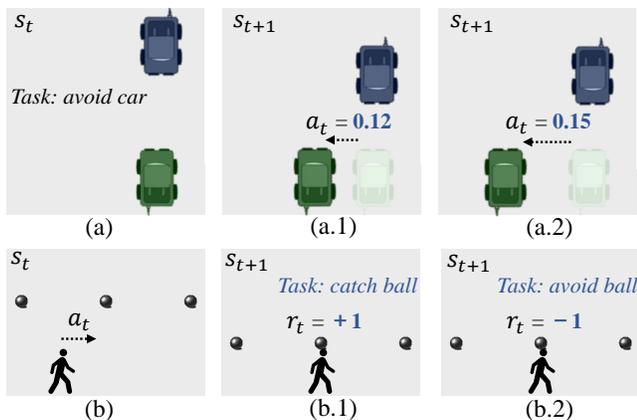


Fig. 1. Examples of action matching’s limitations. (a) Actions moving left by 0.12 and by 0.15 represent the same behavior “avoiding collision” and get the same reward. (b) The same action leads to different rewards when the tasks differ, but action matching methods would give the same explanation.

In this paper, we argue that action matching, the classical setting of RL explanation, may have limitations because it simply assumes that the agent’s actions fully represent the underlying decision-making logic. On the contrary, for deep RL, it is suggested that merely matching actions to attribute features of states is equal to explaining the DNN’s forward process. It barely focuses on the RL agents or the controlling tasks and only relates to the DNN’s architectures and parameters. Considering that the agent’s training objective lies in the reward (or return) instead of the action, we assume that action matching cannot truly interpret the agent as it diverges from RL’s reward-oriented motivation. In fact, plenty of methods have realized the limitations of action-related explanation and attempt to understand RL agents by exploring causal effects on rewards [15]. However, they are usually not as visually understandable as the feature attribution methods because of the structural causal equations.

Therefore, we attempt to take the causal effects into consideration when visualizing feature attribution. Unlike prior methods discussing how agents take actions, we aim to focus on the causal effect of actions and consider the agent’s reward-driven behaviors. Our proposed method leverages reward consistency to explore such causal effects, which means that the interpretation model needs to discover the most salient features in the state that affect the agent’s obtained reward. To solve the gradient disconnection from actions to rewards when the agent interacts with the environment, an RL framework is adopted to achieve reward consistency (such RL interpreting RL model is denoted as RL-in-RL). Our main contributions are summarized as follows: 1) The causal effect on rewards during states’ feature attribution is firstly explored; 2) A novel framework is proposed where the interpretation problem is modeled as a new RL task; 3) We analyze the limitations of the conventional action matching principle and introduce the concept of reward matching. To substantiate these novel ideas and methodologies, we conduct comprehensive experiments. The results and detailed discussions underscore the significance and effectiveness of our proposed method, pointing out that the action matching principle in post-hoc explanations can lead to irrelevant and non-causal attention.

II. RELATED WORK

In this section, we first review the mainstream methods of explaining vision-based RL agents. Existing works that aim to discover how inputs influence agents’ decisions can be partitioned into five main categories: embedding-based methods, attention-based methods, gradient-based methods, perturbation-based methods, and supervision-based methods. Then, since this paper focuses on a post-hoc scenario in which an interpretation model learns the feature attribution of a pretrained and static policy, applications of these main methods for post-hoc explanations are summarily discussed. Besides those works that explain RL agents mainly from the perspective of feature attribution, the causality-based interpretation methods that consider causal effects are also reviewed.

Embedding-based methods. Methods in this category generally focus on visualizing high dimensional data with a commonly used non-linear dimensionality reduction method, t-SNE [16]. A simple idea is to directly draw a t-SNE map where perceptually similar states are demonstrated as points in the vicinity [10], [13], [17]. A recent work uses 1D t-SNE to design the DRLViz model [18], which is able to re-order the memory (*i.e.*, t-SNE projection of absolute values) and identify decisive sub-sets. In addition, the distances among points can also be related to the transition probabilities [19]. However, t-SNE maps are intuitively complex for those without background knowledge of machine learning.

Attention-based methods. Agents in this category learn to draw saliency maps without degrading their performance. Self-attention modules [11], [20] which lay strong emphasis on relative areas of inputs are applied to augment the actor. Key-value attention structures [21], [22] are employed to learn interpretable policies by exploring how they look in the whole environment. Varieties of attention-based interpretation methods are newly proposed, such as TAFE [23], BR-NPA [24] and Attentional Bottleneck [25]. However, these methods usually retrain the agent models and thus are inapplicable to structure-unchangeable or pretrained models.

Gradient-based methods. The main idea underlying gradient-based methods is that the most salient features in the inputs contribute the largest gradient values. A typical approach is Jacobian saliency maps [26], where spatial attributions in input states are computed as the Jacobian through back-propagation. In addition, there are several works modifying gradients to attain more meaningful and valuable saliency maps, such as LRP [27], DeepLIFT [28], Smoothgrad [29], Grad-CAM [30], and SEG-GRAD-CAM [31]. Gradient-based methods uncover the dependencies between inputs and outputs simply and efficiently, but may not produce trustworthy interpretations as the manifold is changed [12].

Perturbation-based methods. These methods aim to measure relative feature importance through perturbing the inputs, like applying occlusion masks and surrogating parts of the images. For example, RISE [32] measures the feature importance by occluding inputs with random mask patterns, and Extremal Perturbation [33] searches for a mask that exerts the most prominent impact on the outputs for a certain area. Recent works include FIDO [34], OPPSD [35], and PERT

[36]. For inputs like images and videos, perturbation-based methods illustrate the effects of distinct parts in the form of saliency maps [37], [38], but the illustration is usually coarse.

Supervision-based methods. This category of interpretation methods is first developed in supervised deep learning. These works aim to optimize for the salient regions using gradient descent techniques [33], [39], [40]. On this basis, some research has been developed to interpret RL in such a supervision-based way, like SSINet [14] and TSCI [41]. The optimization goals of SSINet are action matching and attention sparsity, while TSCI explores temporal-spatial causal interpretations for the sequential decision-making process. In addition, a self-supervised method has been recently proposed to use visual attention as an inductive bias [42].

Post-hoc Explanation Methods. In vision-based RL tasks, the objective of post-hoc explanations is to identify the salient features within state inputs. Attention deemed valid underlines features that directly influence the behaviors of the agent. Existing methods for post-hoc explanations predominantly fall into three categories: gradient-based, perturbation-based, and supervision-based approaches. Each carries its unique complexities and application conditions. The gradient-based methods involve computing the gradients of features that are notably prominent to the agent’s current action output [26]. Perturbation-based methods assess the fluctuation in the action output following the perturbation or removal of certain input information [16], [39]. Supervision-based methods employ an attention mask overlay on the input and instruct its action output to align with the original action at every step [14]. It is worth noting that embedding-based methods, due to their high dimensionality, and attention-based methods, which function only during training, are less commonly used. Fundamentally, these three methodologies are grounded in the assumption that the RL agent can be comprehensively interpreted via *action matching*. This implies the identification of attention regions that significantly influence the agent’s actions.

From the perspective of model understanding, matching actions is equal to matching the DNN’s output in deep RL. Therefore interpretation methods built on action matching are factually explaining the DNN itself. It works well in supervised tasks when the DNN serves as a classifier, whose objective is to make its outputs imitate the true labels. However, in RL tasks, there is no label for action outputs. The DNN, as an RL agent, is trained in a reward-driven way. In this case, using action matching to interpret the agent does not coincide with its intrinsic goal since actions only indirectly illustrate the agent’s reward-driven behaviors. It is, instead, the rewards that directly represent the agent’s behaviors. In fact, plenty of research in RL causality has emphasized the role of rewards by exploring structural equations of states, actions, and rewards [43]–[47]. Likewise, we believe that the exploration of reward-oriented effects can reasonably help post-hoc interpretation methods attend to more essential features.

Causality-based methods. Causality is a crucial topic in the explanation of RL agents [43]–[45]. It remains unclear how agents can dynamically explore new environments and curtail the number of possibly feasible causal structures [46]. Hence, it has become a popular topic whether RL agents can acquire

causal knowledge. Structural causal models [47], assisted by structural equations, are usually adopted to represent the causal relationships. Furthermore, the Action Influence [48] model, which includes actions in causal relationships, uses the state-action ensemble and structural equations to represent itself. To more effectively learn opportunity chains, the Distal Explanation [49] model utilizes a recurrent neural network for analysis and decision trees to promote the accuracy of the prediction. Another work named RAMi [50] is based on the Information Bottleneck [51] principle and aims at the Minimum Description Length. Generally speaking, existing causality-based methods can reveal the inner causality but usually cannot be visualized in a user-friendly way when it comes to understanding vision-based RL agents.

III. PRELIMINARIES

Reinforcement Learning (RL) refers to a general class of algorithms where the agent learns by interacting with the environment. Specifically, the agent takes an action a_t in a state s_t and receives a scalar reward r_t . Meanwhile, the environment E changes to the next state s_{t+1} . The RL problem is generally modeled as a Markov decision process (MDP). An MDP can be described as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, d_0, R, \gamma)$, where \mathcal{S} is the state space, \mathcal{A} is the action space, T defines a transition function of the environment $T(s_{t+1}|s_t, a_t)$, d_0 is the initial state distribution $d_0(s_0)$, R defines a reward function $R(s_t, a_t)$, and $\gamma \in (0, 1]$ is a discount factor [52], [53].

The final goal of an RL problem is to learn a policy, which defines a distribution over actions conditioned on states, $\pi(a_t|s_t)$. The trajectory is a sequence of states and actions of length H , given by $\tau = \{s_0, a_0, \dots, s_H, a_H\}$. The trajectory distribution p_π for a given MDP \mathcal{M} and policy π is:

$$p_\pi(\tau) = d_0(s_0) \prod_{t=0}^{H-1} \pi(a_t|s_t) T(s_{t+1}|s_t, a_t). \quad (1)$$

The learned policy (*i.e.*, the agent) aims at maximizing the return which means the expected sum of discounted future rewards. The RL objective $J(\pi)$ over a horizon H is:

$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[\sum_{t=0}^{H-1} \gamma^t R(s_t, a_t) \right]. \quad (2)$$

In this paper, agents are trained with one of the mainstream methods, the proximal policy optimization (PPO) algorithm following [14]. As a policy gradient method in the actor-critic version, PPO uses trust region update to improve a general stochastic policy with gradient ascent [54].

IV. METHOD

In this section, we first present a reward-oriented interpretation method in Section IV-A to leverage reward consistency for interpretable feature discovery. However, gradient disconnection from actions to rewards blocks its optimization. Hence, an RL-based framework that achieves reward consistency is further proposed in Section IV-B. Last but not least, the extended version of our method for multi-step rewards (or return) consistency is presented in Section IV-C.

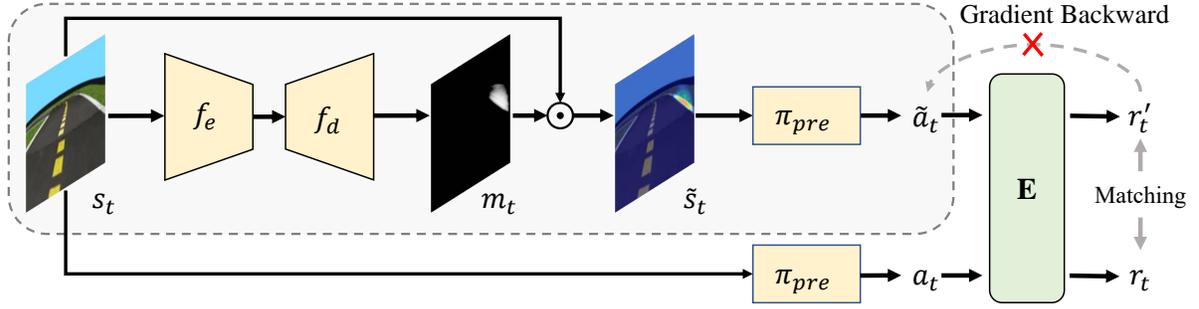


Fig. 2. (1) The gray box illustrates the architecture of our reward-oriented interpretation method. A DNN including an encoder f_e and a decoder f_d is adopted to learn the mask m_t and attentive state \tilde{s}_t . (2) To conduct reward matching, the pretrained policy π_{pre} takes the actions a_t and \tilde{a}_t respectively from the primitive state s_t and attentive state \tilde{s}_t , and then the environment E gives the corresponding rewards r_t and r'_t . (3) However, the reward matching based on supervised learning is blocked by the disconnected gradient backward. Therefore, we model the reward matching problem as an RL task as in Figure 3.

A. Interpretation with Reward Consistency

Causality is the generic relationship between an effect and the cause that gives rise to it. For the interpretation problem, the state and policy can be taken as the “cause” while the reward is the “effect”. Prior research explores causality mainly by the structural causal model (SCM) [55]. An SCM usually aims to construct a causal graph by structural equations. Causal effects in our interpretation problem can be described as $r = G(s|\pi, R)$. G refers to a pre-defined causal structure that explicitly expresses the relationships among variables. However, such a causal structure is usually unknown, and designing it can be a great challenge especially under complex tasks or with ambiguous effects.

As DNNs achieve superior performance in a variety of learning tasks with their powerful representation ability [56]–[58], we consider using a DNN to discover the causal effects. Note that the objective is $r = G(s|\pi, R)$, which can be seen as exploring what features in states affect the agent’s obtained rewards, given the policy π and reward function R . Hence, we design a mask-based architecture as an implicit causal structure to leverage reward consistency. As illustrated in Figure 2, the encoder f_e extracts high dimensional features of the input state s_t , and then the decoder f_d learns a mask m_t . The mask displays the attentive feature importance for $s_t \in \mathbb{R}^{h \times w \times c}$, and $m_t \in [0, 1]^{h \times w \times 1}$. In a frame, h is the height, w is the width, and c is the channel. Next, multiplying the original state by the mask results in the attentive state \tilde{s}_t . We adopt the input state and the attentive state to get the action a_t and the attentive action \tilde{a}_t respectively by the pretrained policy. The environment gives rewards (r_t and r'_t) according to different actions. The process can be formulated as follows:

$$r_t = R(s_t, \pi_{pre}(s_t)) \quad , \quad (3a)$$

$$r'_t = R(s_t, \pi_{pre}(\tilde{s}_t)) \quad , \quad (3b)$$

$$\tilde{s}_t = s_t \cdot f_d(f_e(s_t)) \quad . \quad (3c)$$

We match r_t and r'_t to learn the attentive state which highlights features that directly give rise to the current reward. The goal of this optimization problem is to ensure reward consistency by minimizing $|r_t - r'_t|$.

An intuitive way to achieve this reward matching is the direct gradient backward as supervision-based action matching does. Specifically, after the attentive state \tilde{s}_t is obtained,

action matching simply calculates a matching loss between the attentive action \tilde{a}_t and the original action a_t . For example, the loss is $L = \|a_t - \tilde{a}_t\|_2$. The projection from states to actions through the pretrained policy is continuous and derivable. Thus, direct gradient backward works well. However, when it comes to rewards, such a matching way meets a setback, as shown in Figure 2. The environment gives the reward when receiving the action. There exist internal transition functions in the environment which work in a black-box and end-to-end way. The reward function R is unknown and non-differentiable, which makes the direct gradient backward from rewards to actions impossible. Therefore, the classical supervised learning method doesn’t apply to this reward-related interpretation method. On the other hand, gradient disconnection between actions and rewards never hinders the optimization process in RL problems, which provides a feasible solution to our network’s optimization problem.

B. The RL-in-RL Model

Since the internal reward function R is non-differentiable, reward matching cannot be conducted in the supervised learning way. Therefore, We model the optimization problem in Section IV-A as a reinforcement learning task, which considers reward matching as part of the environment dynamic and thus avoids the gradient disconnection. Accordingly, a new policy is learned to interpret the pretrained policy via maximizing the newly designed rewards where the reward matching objective is implicitly incorporated. Such an RL interpreting RL model is referred to as the RL-in-RL model.

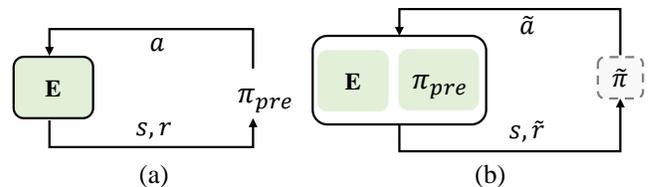


Fig. 3. The interaction process. (a) illustrates how π_{pre} interacts with the environment during pretraining. (b) illustrates the interpretation task where reward matching is modeled as an RL problem. The defined RL-in-RL policy $\tilde{\pi}$ corresponds to the gray box in Figure 2. The reward \tilde{r} is given jointly by the environment E and the pretrained policy π_{pre} .

Provided that the pretrained policy π_{pre} is given, we denote its state space as \mathcal{S} , its action space as \mathcal{A} , and the environment as E . Given the current state $s_t \in \mathcal{S}$, the action $a_t \in \mathcal{A}$

is deterministically decided by $a_t = \pi_{\text{pre}}(s_t)$. Note that both deterministic and stochastic policies are utilized to generate deterministic actions during the inference process, following the common practice for RL agents [54], [59]–[61]. The environment E receives the action a_t and gives the reward by its reward function $r_t = R(s_t, a_t)$. The goal of π_{pre} is to get the maximum accumulated rewards from the environment E . Figure 3(a) illustrates the RL interaction process of π_{pre} .

For the RL-in-RL model, our goal is to learn a new interpretation policy $\tilde{\pi}$. The state space \mathcal{S} and action space \mathcal{A} keep consistent with the pretrained policy π_{pre} . The RL-in-RL policy takes the action $\tilde{a}_t = \tilde{\pi}(s_t)$. For the current interpretation task, the goal changes to achieving the maximum consistency between rewards from the $a_t = \pi_{\text{pre}}(s_t)$ and from the $\tilde{a}_t = \tilde{\pi}(s_t)$. That is to say, for the same state s_t and a well-trained $\tilde{\pi}$, the environment E gives the same reward when receiving the action decided by the interpretation policy $\tilde{\pi}$ as that when receiving the action decided by the pretrained policy π_{pre} . The new reward \tilde{r}_t for the RL-in-RL model can be formulated as follows:

$$\begin{aligned} \tilde{r}_t &= \tilde{R}(s_t, \tilde{a}_t) \\ &= -D\left(R(s_t, a_t), R(s_t, \tilde{a}_t)\right) \\ &= -D\left(R(s_t, \pi_{\text{pre}}(s_t)), R(s_t, \tilde{\pi}(s_t))\right), \end{aligned} \quad (4)$$

where D , utilized as the Mean Squared Error (MSE) in our experiments, measures the distance between two variables. Figure 3(b) illustrates the interaction process of RL-in-RL. Note that the current reward \tilde{r}_t is jointly given by the environment E as well as the fixed policy π_{pre} . The gray box in Figure 3b corresponds to $\tilde{\pi}$ in Figure 2, which means:

$$\tilde{\pi}(s_t) = \pi_{\text{pre}}(s_t \cdot f_d(f_e(s_t))). \quad (5)$$

When calculating \tilde{r}_t as in Equation (4), the input states of π_{pre} and $\tilde{\pi}$ need to be exactly identical, otherwise matching the corresponding rewards becomes misleading. Meanwhile, minor differences between a_t and \tilde{a}_t can lead to huge pixel-wise differences in the next state s_{t+1} . It's hard to obtain the same actions from attentive states as from the original states, and exact action matching is not our goal either. Therefore, to keep the same input states of π_{pre} and $\tilde{\pi}$ at every step $(s_t, \tilde{a}_t, \tilde{r}_t)$, we consider that one trajectory only contains one step in RL-in-RL. That is to say, the environment E resets and gives initial states after every interaction. The objective of this RL task can be formulated as follows:

$$J_{\text{RL}} = \mathbb{E}_{\pi} \left[\tilde{R}(s_t, \tilde{a}_t) \right]. \quad (6)$$

Adopting RL algorithms to maximize the above objective ensures that the mask learns decisive features in states but may also lead to attention degradation. That is to say, all the features could share the same importance, *i.e.*, the maximum attention value 1. In that circumstance, we can get the maximum \tilde{r} but still have no idea what features matter. To solve such attention degradation, an auxiliary task is trained along with the RL interpretation task, in order to encourage the learned mask m_t to be as sparse as possible. In other words,

the number of features with high attentive importance needs to be possibly small. The objective of the auxiliary task is:

$$J_{\text{aux}} = -\|m_t\|_1 = -\|f_d(f_e(s_t))\|_1, \quad (7)$$

where $\|\cdot\|_1$ denotes the L_1 -norm. The final objective contains the RL term and auxiliary term, shown as follows:

$$J = J_{\text{RL}} + \alpha J_{\text{aux}} = \mathbb{E}_{\pi} \left[\tilde{R}(s_t, \tilde{a}_t) \right] - \alpha \|f_d(f_e(s_t))\|_1, \quad (8)$$

where α is a positive scalar controlling the sparseness of the mask. The pseudo-code of training is summarized in Algorithm 1. Note that the encoder in RL-in-RL shares the feature extractor parameters of the pretrained policy to speed up convergence and prevent overfitting [14].

Algorithm 1 The RL-in-RL Model

The fixed pretrained policy π_{pre} to be interpreted;
 The reward function $R(s_t, a_t)$ by environment E ;
 Load f_e and initialize f_d ;
for epoch = 0, 1, ... , until convergence **do**
 for $i = 1, 2, \dots, K$ **do**
 Initialize state s_0 ;
 Get $\tilde{a}_0 = \pi_{\text{pre}}(f_e(f_d(s_0)) \cdot s_0)$;
 Get $\tilde{r}_0 = -D(R(s_0, \pi_{\text{pre}}(s_0)), R(s_0, \tilde{a}_0))$;
 Save $\tau^i = (s_0, \tilde{a}_0, \tilde{r}_0)$;
 end for
 Update f_d by PPO algorithm to maximize the objective J in Equation (8);
end for

C. The RL-in-RL^K Model

As a first step towards using reward consistency to interpret RL agents by feature attribution, we mainly focus on the one-step reward matching for a fair comparison, since existing action matching methods are generally limited to one-step action matching. Nevertheless, RL-in-RL is compatible with the one-step reward as well as multi-step rewards (or returns) consistency. To adopt the multi-step version of RL-in-RL (denoted as RL-in-RL^K), the defined reward function in Equation (4) is reformulated as follows:

$$\begin{aligned} \tilde{r}_t^K &= -D\left(\sum_{i=0}^{K-1} \gamma^i R(s_{t+i}, a_{t+i}), \right. \\ &\quad \left. R(s_t, \tilde{a}_t) + \sum_{j=1}^{K-1} \gamma^j R(s'_{t+j}, a'_{t+j})\right), \end{aligned} \quad (9)$$

where $s_{t+i+1} = T(s_{t+i}, a_{t+i})$ and $a_{t+i} = \pi_{\text{pre}}(s_{t+i})$ are in the pretrained behavior trajectory, $s'_{t+1} = T(s_t, \tilde{a}_t)$, $s_{t+j+1} = T(s'_{t+j}, a'_{t+j})$, and $a'_{t+j} = \pi_{\text{pre}}(s'_{t+j})$ are in the interpretation behavior trajectory, γ is a discount factor, and K is the observation length.

When $K = 1$, it equals the interpretation problem with one-step reward consistency, *i.e.*, RL-in-RL. When $K > 1$, the RL-in-RL^K focuses on observing the long-term effects of the behavior, and keeps the multi-step rewards or even return

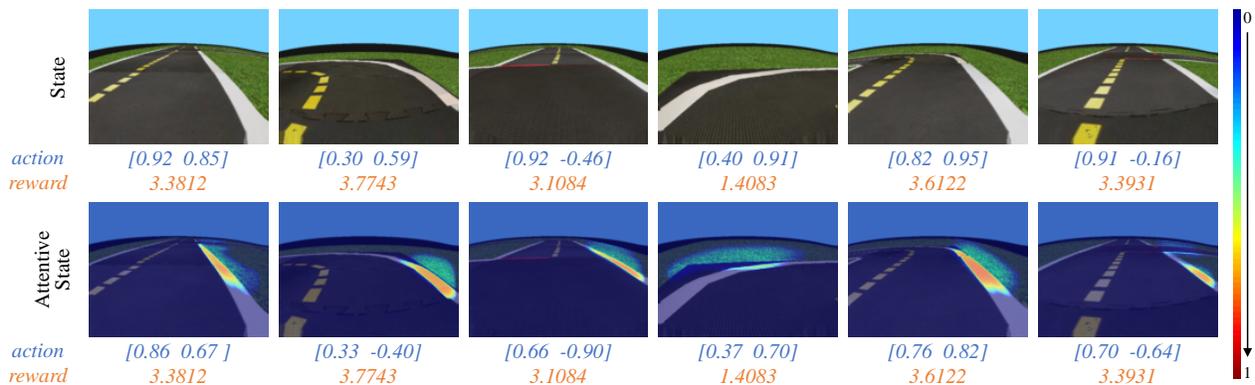


Fig. 4. The performance of our proposed RL-in-RL model in the Duckietown environment. The attentive state is an overlaid combination of the state and the attentive heatmap of feature attribution. The feature importance ranges from 0 to 1 as the heatmap color changes from blue to red.

($K = H$) consistency. By lengthening the step of reward consistency (*i.e.*, adopting a larger observation length K), the attentive features would illustrate more information that helps explain long-term behaviors.

V. VALIDITY OF OUR METHOD

In this section, we first describe experimental settings and adopted environments in Section V-A. Then, results are shown to validate RL-in-RL for its interpretability in Section V-B.

A. Setup

Our experiments are conducted on the Atari 2600 games and the Duckietown environment. The Euclidean Distance is adopted as the D function in Equation (4) and the UNet [62] serves as the encoder-decoder architecture in Figure 2. Other experimental details and ablation studies on our hyperparameters are in the Appendices.

Atari 2600 [63] is a widely used benchmark in the field of RL interpretation. It consists of various RL tasks like playing ping-pong and hiding from monsters. Actions in these tasks are all discrete such as “left”, “right”, “up”, and “down”. The states are color images and we adopt the $84 \times 84 \times 4$ stacked grayscale images as inputs following [17]. The reward during training is normalized to -1, 0, and 1.

Duckietown [64] is an autonomous driving simulator environment. The states are $120 \times 160 \times 3$ color images from a single camera. The actions contain two continuous numbers between -1 and 1, corresponding to the forward velocity and steering angle. A positive velocity makes the agent go forward, and a positive steering angle makes it turn left. The reward function consists of the speed reward, the lane reward, and the obstacle penalty. The speed reward term encourages a large driving speed. The lane reward requires the agent to drive on the right side of the lane. The obstacle penalty asks the agent to avoid obstacles and invalid driving areas (for example, the grass outside the lane).

B. Evaluations

We investigate whether the proposed method provides valid explanations and compare RL-in-RL with other popular approaches. Specifically, experiments in this section aim at

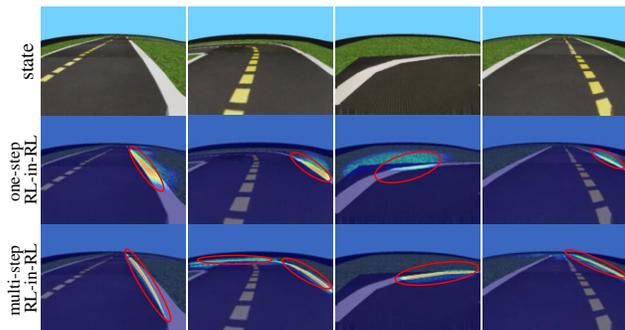


Fig. 5. The attentive features of RL-in-RL^K with observation length $K = 10$.

answering two questions. **First**, does RL-in-RL achieve reward consistency as expected? **Second**, can RL-in-RL result in high-quality feature attribution for RL agents compared to action matching methods? We emphasize that this work primarily focuses on validating reward consistency for interpretable feature discovery (Section V) and studying differences between reward matching and action matching (Section VI). Our aim is not to “beat” all action matching methods or present a perfect method for RL interpretation.

Validity of RL-in-RL. We compare the actions, rewards, and RL-in-RL’s learned attention in the challenging self-driving environment. As shown in Figure 4, the attentive state illustrates an overlaid state with its attentive heatmap of feature attribution. Quantitative results show that RL-in-RL manages to keep the same reward as that of the policy to be interpreted, and thus achieves our motivation to discover the reward-related features during interpretation. Note that actions are factually varied for the same rewards. It empirically proves our assumption that the intrinsic goal of a policy, the pursuit of rewards, cannot be fully represented by action matching. Meanwhile, visualized results show that the pretrained policy mainly focuses on the right white line but attends to the near left white line when other lines are unobservable, which conforms to human attention under the right-hand traffic rule.

In addition to one-step reward consistency, we validate the proposed method with multi-step rewards consistency in Figure 5. The RL-in-RL^K generally maintains the same attention pattern (*i.e.*, the left and right white lines) with one-step RL-in-RL, but the sights are set further ahead. It is suggested that RL-in-RL^K can not only leverage the reward

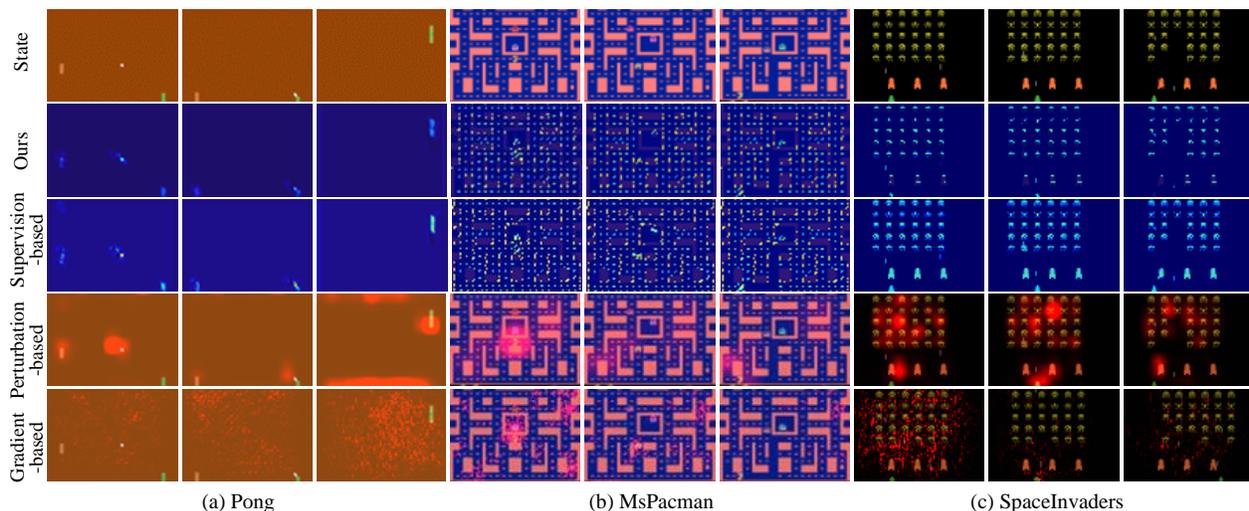


Fig. 6. Comparisons among different interpretation methods on Atari 2600. Our RL-in-RL model and the supervision-based method are visualized in the overlaid heatmap as in Figure 4. The perturbation-based and gradient-based methods highlight the attention areas on the saliency-overlaid state.

information but also provide an adjustable observation length for long-term behavior interpretation.

Comparative Evaluation. Several popular RL explanation methods based on gradient [13], perturbation [12], and supervision [14] are compared with our proposed method. The first metric is the saliency map, one of the most well-known visual methods where the intensity of the pixel color at a particular location corresponds to the importance of the input value [65]–[67]. As illustrated in Figure 6, the RL-in-RL and supervision-based methods learn more precise attention than that of the perturbation-based and gradient-based methods. The former two methods also display the relative feature importance in the heatmap visualization while the latter two are incapable of telling. The second metric is the average return of the policy that only takes masked states as inputs, which quantitatively evaluates the quality of masks generated by various explanation methods. In Table I, the RL-in-RL and supervision-based methods achieve comparable performance while the other two methods lead to significant performance degradation, when the policy can only access the attentive pixels learned by corresponding explanation methods.

TABLE I
AVERAGE RETURNS OVER 5 RANDOM SEEDS

Task	Ours	Supervision method [14]	Perturbation method [12]	Gradient method [13]
Enduro	2802.43	2755.42	1741.10	819.92
Seaquest	2560.77	2356.67	1830.00	836.00
SpaceInvaders	738.46	740.45	562.92	297.73

The supervision-based method with action matching has competitive precision with the RL-in-RL model but some attentive details are different. However, such differences are subtle and analyses may be subjective. Therefore, we further compare the supervision-based action matching method with our reward-oriented method in the Duckietown environment. As shown in Figure 7, the main attention regions by action matching are the left white line, the middle yellow line, and the right white line (denoted as F_w^{left} , F_y , and F_w^{right} respectively). The results conform to our intuition since such three parts

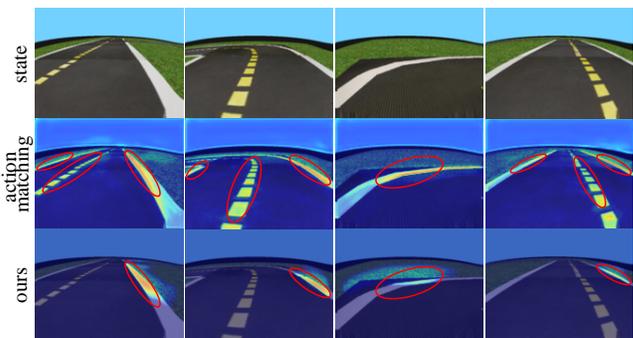


Fig. 7. Comparisons between the best-performing action matching method and our reward-oriented RL-in-RL on the Duckietown environment.

exactly constitute the whole lane. On the other hand, the RL-in-RL model merely highlights the right white line F_w^{right} except that if only the left white line is in sight (as shown in the third column of Figure 7). Since the widths of the lane and the car are constants in Duckietown, the location of the right line can uniquely determine the locations of other lines. Hence, the attentive pattern discovered by RL-in-RL coincides with the task’s reward function, where the agent is rewarded for driving on the right side of the lane and penalized for driving in the grass outside. To further understand their attention differences, a series of analytical experiments are designed in Section VI.

VI. UNDERSTANDING REWARD AND ACTION MATCHING

In this section, we aim to understand reward matching and action matching while interpreting RL agents by exploring their attention differences in Figure 7. The fine-grained action matching method based on the supervised learning [14] is compared with RL-in-RL in the Duckietown environment. Since the right white line F_w^{right} always has high importance in the above two explaining patterns in Figure 7, our main concern is whether the left white line F_w^{left} and the middle yellow line F_y are redundant attention. Hence, to explore truly crucial features of the pretrained policy, experiments are designed to answer the following three questions.

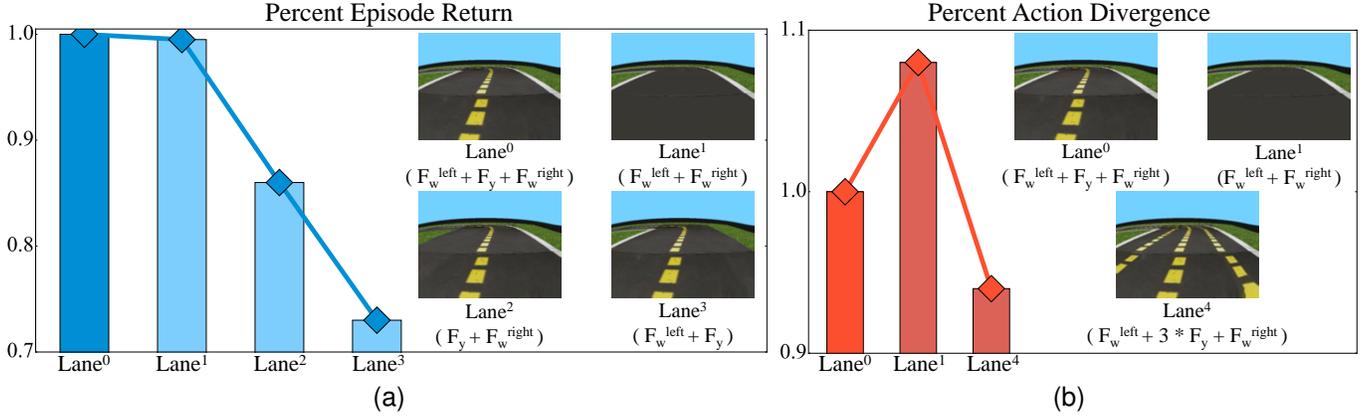


Fig. 8. Quantitative analyses averaged across 100 random seeds. (a) The percent episode return of the pretrained policy under different lane patterns. This measures how three lines (*i.e.*, F_w^{left} , F_y , and F_w^{right}) affect the policy’s performance respectively. (b) The percent action divergence of the interpretation model under different lane patterns, compared to the pretrained policy’s actions. This measures how three lines affect the interpretation model’s action consistency with the pretrained policy respectively.

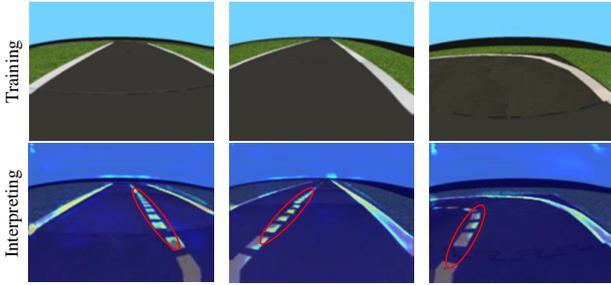


Fig. 9. The interpretation results from the action matching method, when the policy is pretrained without the middle yellow line.

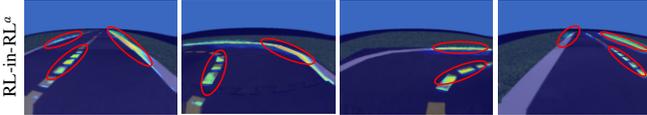


Fig. 10. The attention pattern of RL-in-RL^a.

1) Is the extra attention discovered by the action matching method redundant?

The first question is whether the extra attention F_w^{left} and F_y are redundant or not. In other words, do they truly affect the performance of the pretrained policy? To figure out this question, the pretrained policy is evaluated under different lane patterns as shown in Figure 8a. The lane⁰ is the primitive pattern that the policy is pretrained with, which has two white lines and one yellow line.

Experimental results are illustrated in Figure 8a. Firstly, F_y is removed in the lane¹, and the pretrained policy maintains the comparative performance. It validates that the yellow line is not the essential feature of the pretrained policy and thus the attention is redundant. Secondly, F_w^{left} is removed in the lane², and the pretrained policy’s performance slightly decreases. In fact, the RL-in-RL model does highlight this region in cases where no other lines are in sight. Therefore, we suggest that the left white line occasionally affects the agent’s decision but is redundant most of the time. Thirdly, F_w^{right} is removed in the lane³, and a sharp decrease in performance is observed. It

proves that the right white line is crucial for the agent and the RL-in-RL model discovers the truly essential feature.

To further verify that the yellow line is the redundant attention, the policy π_{pre} is re-trained in the lane¹ as shown in Figure 8a. Since the agent is trained without any involvement of the yellow line, its decision-making process obviously has nothing to do with this feature. We adopt the supervision-based action matching method to figure out its attention features and the results are illustrated in Figure 9. The interpretation method based on action matching still focuses on the middle yellow line, which further proves that action matching leads to irrelevant and redundant feature attention.

2) Does the action matching principle lead to redundant attention?

After F_y is proved to be redundant attention of the action matching method, the following question is what the cause is. It remains unclear whether it comes from the action matching principle or the supervised learning approach. To decompose these two factors, a variant of RL-in-RL (denoted as RL-in-RL^a) is designed to conduct action matching in the RL framework. RL-in-RL^a changes the reward function in Equation (4) to:

$$\tilde{r}_t^a = -D(a_t, \tilde{a}_t) = -D(\pi_{\text{pre}}(s_t), \tilde{\pi}(s_t)). \quad (10)$$

The goal of RL-in-RL^a is the same as that of action matching, while its optimization method is reinforcement learning instead of supervised learning. Figure 10 shows that attentive regions in RL-in-RL^a are similar to that in the supervision-based action matching method (as shown in Figure 7), where the redundant attention F_y still has high importance. In this case, we can conclude that the redundant attention results from the action matching principle instead of the optimization method.

3) Why does the action matching principle lead to redundant attention?

As far as we know, the action matching principle can lead to fake attention like F_y . Here comes the next question: why does it cause fake attention? In other words, what is the contribution of F_y during the process of matching actions? Considering a

deterministic environment where the same action a_t under the same state s_t results in the same next state s_{t+1} , matching actions factually equals matching the next states. In this case, a minor variance in the agent’s position can induce significant pixel differences in the next state. Hence, an assumption of the yellow line’s role is to help the agent locate its current position and thus take the same action. To verify our assumption, another two yellow lines are added to the lane, as shown in lane⁴ of Figure 8b. Actions of one trajectory are collected separately for lane⁰, lane¹, and lane⁴. The action matching loss between two trajectories collected respectively under the original states and the attentive states for the same lane pattern is calculated as follows:

$$L_{\text{KL}} = D_{\text{KL}}(A_s, A_{\bar{s}}), \quad (11)$$

where D_{KL} refers to the KL divergence and A is the action trajectory within 500 interactions with the environment. As shown in Figure 8b, compared to actions under the primitive pattern (*i.e.*, lane⁰), actions under the lane¹ have larger divergence while those under the lane⁴ are less divergent. That is to say, extra attention on the yellow lane indeed raises the action matching degree although it’s irrelevant to the pursuit of high rewards for the agent. Therefore, the redundant attention caused by action matching can be seen as a kind of “overfitting”, which is dedicated to identical actions but neglects the reward-related goal of the policy.

Discussion. What we know so far is that using action matching to explain RL agents would lead to redundant feature attribution. It results from the fact that actions only indirectly represent the agent’s reward-related goal. On the other hand, we have to admit that interpretable features discovered by action matching methods indeed help users predict the agent’s actions, which is a natural benefit of matching actions. Meanwhile, the interpretable features discovered with reward consistency are more like the “safety boundary”, which are the key elements that guarantee the agent’s expected performance. Hopefully, reward and action matching principles can complement each other and provide a more comprehensive understanding of deep RL in real-world applications, for both users and researchers.

VII. CONCLUSION

In this paper, we discussed the limitations of the commonly used assumption, the action matching principle, in RL interpretation methods. It is suggested that action matching cannot truly interpret the agent since it differs from the reward-oriented goal of RL. Hence, the proposed method firstly leverages reward consistency during feature attribution and models the interpretation problem as a new RL problem, denoted as RL-in-RL. Moreover, it provides an adjustable observation length for one-step reward or multi-step reward (or return) consistency, depending on the requirements of behavior analyses. Extensive experiments validate the proposed model and support our concerns that action matching would lead to redundant and non-causal attention during interpretation since it is dedicated to exactly identical actions and thus results in a sort of “overfitting”. Nevertheless, although RL-in-RL

shows superior interpretability and dispenses with redundant attention, further exploration of interpreting RL tasks with explicit causality is left for future work.

APPENDIX A EXPERIMENTAL DETAILS

A. Network Structure

In our experiments, the PPO algorithm is adopted for training. In PPO, the critic network is used to predict the state-value function and shares a common feature extractor with the actor. The actor and critic networks are each connected with a linear layer after the feature extractor. In the proposed method, the critic’s linear layer of the pretrained policy is retrained because the state value changes in the interpretation RL problem. The RL-in-RL model, illustrated in Figure 2, uses an encoder-decoder architecture with DNNs for a universal semantic segmentation task. It takes state images as inputs and produces learned saliency masks, a dense prediction task common in vision applications such as semantic segmentation [56], [57] and scene depth estimation [58]. To mitigate task-irrelevant attention that may arise from complex architectures, we’ve adopted the simpler U-Net architecture with minor modifications, following its proven efficacy in RL explanations [14], [41]. As shown in Figure 11, firstly, the Sigmoid layer is added to project the mask from $\mathbb{R}^{H \times W \times C}$ to $[0, 1]^{H \times W \times C}$, which can better reflect the relative feature importance and ensure the scale of attentive states. Secondly, the $ReLU(f(x))$ layer is added to speed up convergence. It can be formulated as $ReLU(\frac{x-\beta}{1-\beta})$, where $\beta \in [0, 1)$ is a hyperparameter adjusting its effects on the speed of model converging.

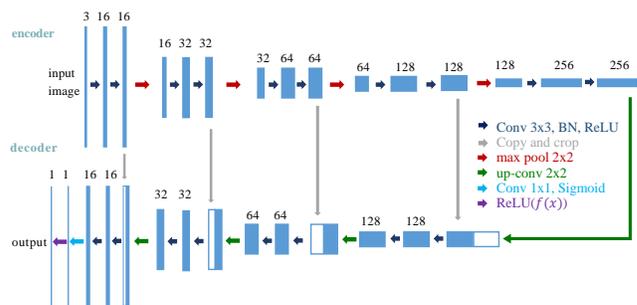


Fig. 11. The architecture of the encoder-decoder network in RL-in-RL.

B. Implementation

We use the following software versions: Python 3.7, Pytorch 1.10.0 [68], Gym 0.15.4 [69], and Duckietown 6.2.24 [64]. Our implementations of gradient-based [13], perturbation-based [12], and supervision-based [14] methods follow their respective author-provided implementations from GitHub or published papers. The Optimal hyperparameters are adopted for each method.

APPENDIX B FURTHER STUDIES

A. Validity of RL-in-RL

The proposed RL-in-RL model is verified on more tasks of the Atari2600, as shown in Figure 12. In the Enduro task for

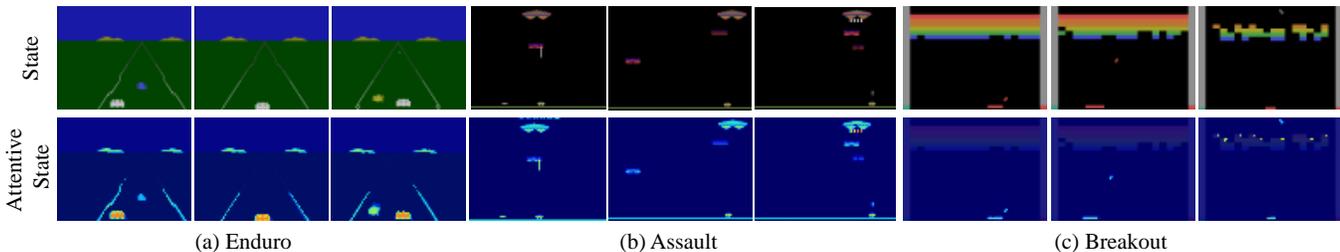


Fig. 12. More results of the RL-in-RL model on Atari2600. The attentive states are visualized as in Figure 4.

avoiding obstacles while driving, the attention is mainly on the driving boundary, obstacles, and the agent’s own position. It is clear that the distant boundary is less attended to since it has a smaller impact on the current behavior. Similarly, the most distant obstacles are not noteworthy as shown in the third scenario. In the Assault task for offense and defense, attentive areas are generally the player, spaceships, and bullets. In the Breakout task for hitting colored squares with a ball, the agent mainly focuses on the small ball and the board, since whether the board catches the ball or not directly determines the reward. Another thing worth noting is the top of the rainbow block in the third scenario. It is suggested that for the pretrained policy, the actor prefers to send the ball to the top of the rainbow block to get a higher reward.

B. Ablation Studies on Hyperparameters

The effects of hyperparameters α and β are explored on the Atari AirRaid task. α is the weight of the auxiliary task loss as illustrated in Equation (8), and β is to speed up the training process as described in Appendix A-A. Results are demonstrated in Figure 13. In the first experiment, α is fixed to 0.1, and β is adjusted between 0 and 0.3. Visualized results are mostly the same since β only affects the converging speed. In the second experiment, β is fixed to 0.1 and α is adjusted between 0 and 0.4. When α is set to 0, the attention almost highlights all of the observations. As α increases, the mask becomes more sparse and emphasizes more noteworthy parts. Specifically, the background’s and the buildings’ attention importance gradually becomes lower because of their relatively small impact on the agent’s decision. The player’s and aircraft’s attention importance has become smaller but remains the most salient in the whole observation, as they directly affect the agent’s obtained rewards.

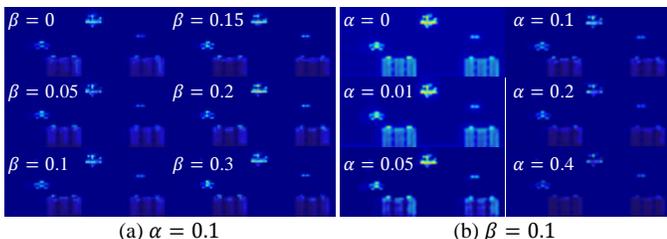


Fig. 13. Two ablation studies on the hyperparameters. α is the weight of the auxiliary task loss as in Equation (8), while β is to speed up the training process as described in Appendix A.

Furthermore, we compare the effect of α to the action matching method [14]. This hyperparameter controls attention



Fig. 14. Comparative results of the RL-in-RL method and the action matching method’s ablation studies on the hyperparameter α .

sparseness. As shown in Figure 14, the attentive patterns and their differences maintain as α varies. It coincides with our conclusion in Section VI that the redundant feature discovery is caused by the action matching principle instead of the optimization method or the hyperparameter.

C. Tasks with Sparse Rewards

This paper hasn’t particularly discussed the sparse reward problem as most of the explainable RL research [6], [15], [70], [71]. However, our experiments actually validated the ability of RL-in-RL to deal with such tasks. Specifically, in the Atari Pong, Breakout, and SpaceInvaders environments (cf. Figure 6 and 12), agents only get non-zero (1 or -1) rewards when the round is over. As shown in Table II, during the game, only 1.36%, 4.37%, and 3.59% of rewards are non-zero respectively (evaluated on 100 random seeds). Furthermore, the RL-in-RL^K presented in Section IV-C is compatible with the sparse reward problem and can fully leverage the sparse reward information.

TABLE II
TASKS WITH SPARSE REWARDS

Task	Pong	Breakout	SpaceInvaders
Horizon Steps	1633.93	1356.50	921.09
Non-zero Reward Steps	22.27	59.29	33.03
Non-zero Reward Percentage	1.36%	4.37%	3.59%

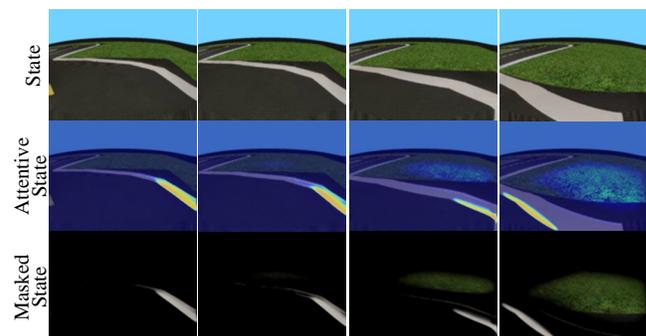


Fig. 15. Visualization of attention shift in the zigzag turn. Three rows correspond to original states, attentive states, and masked states, respectively.

D. Analysis of Failure Case

RL-in-RL provides an explanation for why RL agents may not perform robustly in novel situations different from their training scene from the viewpoint of *attention shift*, in a similar manner to [14]. As visualized in Figure 15, the agent fails to judge decisive features and mistakenly attends to the task-irrelevant grassland, because the zigzag turn has never been encountered during training. Catastrophic cumulative attention shifts would lead to problematic situations and cause significant performance degradation for practical RL agents.

REFERENCES

- [1] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, “Grandmaster level in starcraft ii using multi-agent reinforcement learning,” *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.
- [2] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, “Solving rubik’s cube with a robot hand,” *CoRR*, vol. abs/1910.07113, 2019.
- [3] S. Li, L. Ding, H. Gao, Y. Liu, N. Li, and Z. Deng, “Reinforcement learning neural network-based adaptive control for state and input time-delayed wheeled mobile robots,” *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 50, no. 11, pp. 4171–4182, 2020.
- [4] X. Xu, L. Zuo, X. Li, L. Qian, J. Ren, and Z. Sun, “A reinforcement learning approach to autonomous decision making of intelligent vehicles on highways,” *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 50, no. 10, pp. 3884–3897, 2020.
- [5] T. Han, S. Nagesh Rao, D. P. Filev, K. A. Redmill, and Ü. Özgüner, “An online evolving method for a safe and fast automated vehicle control system,” *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 52, no. 9, pp. 5723–5735, 2022.
- [6] C. Glanois, P. Weng, M. Zimmer, D. Li, T. Yang, J. Hao, and W. Liu, “A survey on interpretable reinforcement learning,” *CoRR*, vol. abs/2112au.13112, 2021.
- [7] A. Adadi and M. Berrada, “Peeking inside the black-box: A survey on explainable artificial intelligence (XAI),” *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.
- [8] F. K. Doshirovic, M. Brcic, and N. Hlupic, “Explainable artificial intelligence: A survey,” in *MIPRO*. IEEE, 2018, pp. 210–215.
- [9] Y. Qing, S. Liu, J. Song, and M. Song, “A survey on explainable reinforcement learning: Concepts, algorithms, challenges,” *CoRR*, vol. abs/2211.06665, 2022.
- [10] R. M. Annasamy and K. Sycara, “Towards better interpretability in deep q-networks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 4561–4569.
- [11] A. Manchin, E. Abbasnejad, and A. van den Hengel, “Reinforcement learning with attention that works: A self-supervised approach,” in *ICONIP (5)*, ser. Communications in Computer and Information Science, vol. 1143. Springer, 2019, pp. 223–230.
- [12] S. Greydanus, A. Koul, J. Dodge, and A. Fern, “Visualizing and understanding atari agents,” in *ICML*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 1787–1796.
- [13] T. Zahavy, N. Ben-Zrihem, and S. Mannor, “Graying the black box: Understanding dqns,” in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 48. JMLR.org, 2016, pp. 1899–1908.
- [14] W. Shi, G. Huang, S. Song, Z. Wang, T. Lin, and C. Wu, “Self-supervised discovering of interpretable features for reinforcement learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
- [15] A. Heuillet, F. Couthouis, and N. Diaz-Rodríguez, “Explainability in deep reinforcement learning,” *Knowledge-Based Systems*, vol. 214, p. 106685, 2021.
- [16] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, no. 11, 2008.
- [17] 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, pp. 529–533, 2015.
- [18] T. Jaunet, R. Vuillemot, and C. Wolf, “Drlviz: Understanding decisions and memory in deep reinforcement learning,” in *Computer Graphics Forum*, vol. 39, no. 3. Wiley Online Library, 2020, pp. 49–61.
- [19] Y. Engel and S. Mannor, “Learning embedded maps of markov processes,” in *ICML*. Morgan Kaufmann, 2001, pp. 138–145.
- [20] D. Nikulin, A. Ianina, V. Aliev, and S. I. Nikolenko, “Free-lunch saliency via attention in atari agents,” in *ICCV Workshops*. IEEE, 2019, pp. 4240–4249.
- [21] J. Choi, B.-J. Lee, and B.-T. Zhang, “Multi-focus attention network for efficient deep reinforcement learning,” in *Workshops at the thirty-first AAAI conference on artificial intelligence*, 2017.
- [22] A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. J. Rezende, “Towards interpretable reinforcement learning using attention augmented agents,” in *NeurIPS*, 2019, pp. 12 329–12 338.
- [23] Q. Zhang, X. Ma, Y. Yang, C. Li, J. Yang, Y. Liu, and B. Liang, “Learning to discover task-relevant features for interpretable reinforcement learning,” *IEEE Robotics and Automation Letters*, 2021.
- [24] T. Gomez, S. Ling, T. Fréour, and H. Mouchère, “Improve the interpretability of attention: A fast, accurate, and interpretable high-resolution attention model,” *CoRR*, vol. abs/2106.02566, 2021.
- [25] J. Kim and M. Bansal, “Towards an interpretable deep driving network by attentional bottleneck,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7349–7356, 2021.
- [26] K. Simonyan, A. Vedaldi, and A. Zisserman, “Deep inside convolutional networks: Visualising image classification models and saliency maps,” in *ICLR (Workshop Poster)*, 2014.
- [27] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, “On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation,” *PLoS one*, vol. 10, no. 7, p. e0130140, 2015.
- [28] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, “Not just a black box: Learning important features through propagating activation differences,” *CoRR*, vol. abs/1605.01713, 2016.
- [29] D. Smilkov, N. Thorat, B. Kim, F. B. Viégas, and M. Wattenberg, “Smoothgrad: removing noise by adding noise,” *CoRR*, vol. abs/1706.03825, 2017.
- [30] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *ICCV*. IEEE Computer Society, 2017, pp. 618–626.
- [31] K. Vinogradova, A. Dibrov, and G. Myers, “Towards interpretable semantic segmentation via gradient-weighted class activation mapping (student abstract),” in *AAAI*. AAAI Press, 2020, pp. 13 943–13 944.
- [32] V. Petsiuk, A. Das, and K. Saenko, “RISE: randomized input sampling for explanation of black-box models,” in *BMVC*. BMVA Press, 2018, p. 151.
- [33] R. Fong, M. Patrick, and A. Vedaldi, “Understanding deep networks via extremal perturbations and smooth masks,” in *ICCV*. IEEE, 2019, pp. 2950–2958.
- [34] C. Chang, E. Creager, A. Goldenberg, and D. Duvenaud, “Explaining image classifiers by counterfactual generation,” in *ICLR (Poster)*. OpenReview.net, 2019.
- [35] V. Kim, H. Cho, and S. Chung, “One-step pixel-level perturbation-based saliency detector,” 2021.
- [36] P. S. Parvatharaju, R. Doddaiiah, T. Hartvigsen, and E. A. Rundensteiner, “Learning saliency maps to explain deep time series classifiers,” in *CIKM*. ACM, 2021, pp. 1406–1415.
- [37] M. Ivanovs, R. Kadikis, and K. Ozols, “Perturbation-based methods for explaining deep neural networks: A survey,” *Pattern Recognition Letters*, vol. 150, pp. 228–234, 2021.
- [38] S. Yang, W. Wang, C. Liu, and W. Deng, “Scene understanding in deep learning-based end-to-end controllers for autonomous vehicles,” *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 49, no. 1, pp. 53–63, 2019.
- [39] R. C. Fong and A. Vedaldi, “Interpretable explanations of black boxes by meaningful perturbation,” in *ICCV*. IEEE Computer Society, 2017, pp. 3449–3457.
- [40] P. Dabkowski and Y. Gal, “Real time image saliency for black box classifiers,” in *NIPS*, 2017, pp. 6967–6976.
- [41] W. Shi, G. Huang, S. Song, and C. Wu, “Temporal-spatial causal interpretations for vision-based reinforcement learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
- [42] H. Wu, K. Khetarpal, and D. Precup, “Self-supervised attention-aware reinforcement learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 311–10 319.
- [43] J. Pearl, *Causality*. Cambridge university press, 2009.
- [44] B. Schölkopf, “Causality for machine learning,” *CoRR*, vol. abs/1911.10500, 2019.
- [45] T. Herlau, “Causal variables from reinforcement learning using generalized bellman equations,” *CoRR*, vol. abs/2010.15745, 2020.

- [46] M. Edmonds, J. Kubricht, C. Summers, Y. Zhu, B. Rothrock, S.-C. Zhu, and H. Lu, "Human causal transfer: Challenges for deep reinforcement learning," in *CogSci*, 2018.
- [47] J. Y. Halpern and J. Pearl, "Causes and explanations: A structural-model approach. part i: Causes," *The British journal for the philosophy of science*, 2020.
- [48] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere, "Explainable reinforcement learning through a causal lens," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 03, 2020, pp. 2493–2500.
- [49] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere, "Distal explanations for explainable reinforcement learning agents," *CoRR*, vol. abs/2001.10284, 2020.
- [50] G. Liu, X. Sun, O. Schulte, and P. Poupart, "Learning tree interpretation from object representation for deep reinforcement learning," in *NeurIPS*, 2021, pp. 19 622–19 636.
- [51] N. Tishby, F. C. N. Pereira, and W. Bialek, "The information bottleneck method," *CoRR*, vol. physics/0004057, 2000.
- [52] R. S. Sutton, A. G. Barto *et al.*, "Introduction to reinforcement learning," 1998.
- [53] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *CoRR*, vol. abs/2005.01643, 2020.
- [54] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.
- [55] R. Guo, L. Cheng, J. Li, P. R. Hahn, and H. Liu, "A survey of learning causality with data: Problems and methods," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–37, 2020.
- [56] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI (3)*, ser. Lecture Notes in Computer Science, vol. 9351. Springer, 2015, pp. 234–241.
- [57] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017.
- [58] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *CVPR*. IEEE Computer Society, 2016, pp. 4040–4048.
- [59] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.
- [60] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," in *ICLR*. OpenReview.net, 2022.
- [61] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," in *NeurIPS*, 2020.
- [62] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI (3)*, ser. Lecture Notes in Computer Science, vol. 9351. Springer, 2015, pp. 234–241.
- [63] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.
- [64] M. Chevalier-Boisvert, F. Golemo, Y. Cao, B. Mehta, and L. Paull, "Duckietown environments for openai gym," <https://github.com/duckietown/gym-duckietown>, 2018.
- [65] A. Krajna, M. Brcic, T. Lipic, and J. Doncevic, "Explainability in reinforcement learning: perspective and position," *CoRR*, vol. abs/2203.11547, 2022.
- [66] A. Zyteck, I. Arnaldo, D. Liu, L. Berti-Équille, and K. Veeramachaneni, "The need for interpretable features: Motivation and taxonomy," *SIGKDD Explor.*, vol. 24, no. 1, pp. 1–13, 2022.
- [67] L. He, N. Aouf, and B. Song, "Explainable deep reinforcement learning for uav autonomous path planning," *Aerospace science and technology*, vol. 118, p. 107052, 2021.
- [68] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*, 2019, pp. 8024–8035.
- [69] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016.
- [70] A. Chatzimpampas, R. M. Martins, I. Jusufi, and A. Kerren, "A survey of surveys on the use of visualization for interpreting machine learning models," *Information Visualization*, vol. 19, no. 3, pp. 207–233, 2020.
- [71] A. Alharin, T.-N. Doan, and M. Sartipi, "Reinforcement learning interpretation methods: A survey," *IEEE Access*, vol. 8, pp. 171 058–171 077, 2020.