Neural Networks 161 (2023) 228–241

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Achieving efficient interpretability of reinforcement learning via policy distillation and selective input gradient regularization

Jinwei Xing [a,*], Takashi Nagata [b], Xinyun Zou [b], Emre Neftci [a], Jeffrey L. Krichmar [a,b]

[a] Department of Cognitive Sciences, University of California, Irvine, 92697, CA, USA
[b] Department of Computer Science, University of California, Irvine, 92697, CA, USA

ARTICLE INFO

Article history:
Received 29 May 2022
Received in revised form 18 November 2022
Accepted 19 January 2023
Available online 24 January 2023

Keywords:
Efficient interpretability
Interpretable reinforcement learning
Saliency map

ABSTRACT

Although deep Reinforcement Learning (RL) has proven successful in a wide range of tasks, one challenge it faces is interpretability when applied to real-world problems. Saliency maps are frequently used to provide interpretability for deep neural networks. However, in the RL domain, existing saliency map approaches are either computationally expensive and thus cannot satisfy the real-time requirement of real-world scenarios or cannot produce interpretable saliency maps for RL policies. In this work, we propose an approach of Distillation with selective Input Gradient Regularization (DIGR) which uses policy distillation and input gradient regularization to produce new policies that achieve both high interpretability and computation efficiency in generating saliency maps. Our approach is also found to improve the robustness of RL policies to multiple adversarial attacks. We conduct experiments on three tasks, MiniGrid (Fetch Object), Atari (Breakout) and CARLA Autonomous Driving, to demonstrate the importance and effectiveness of our approach.

1. Introduction

Reinforcement learning (RL) systems have achieved impressive performance in a wide range of simulated domains such as games (Mnih et al., 2015; Silver et al., 2016; Vinyals et al., 2019; Wang et al., 2020), robotics (Fujimoto, Hoof, & Meger, 2018; Haarnoja, Zhou, Abbeel, & Levine, 2018; Lillicrap et al., 2015), automatic control (Li, Liu, & Wang, 2017; Wang, Ha, & Zhao, 2022) and computer vision tasks (Le, Rathour, Yamazaki, Luu, & Savvides, 2021). However, the interpretability of an agent's decision making and robustness to attacks need to be addressed when applying RL to real-world problems. For instance, in a self-driving scenario, real-time interpretability could explain how an RL agent produces a decision in response to its observed states and enable a safer deployment under real-world conditions and adversarial attacks (Bojarski et al., 2018; Ferdowsi, Challita, Saad, & Mandayam, 2018; McAllister, Kahn, Clune, & Levine, 2019).

Saliency maps in deep learning are used to interpret input features that are believed to be important for the neural network output (Fong & Vedaldi, 2017; Nguyen, Yosinski, & Clune, 2019; Selvaraju et al., 2017; Simonyan, Vedaldi, & Zisserman, 2013; Smilkov, Thorat, Kim, Viégas, & Wattenberg, 2017; Sundararajan, Taly, & Yan, 2017; Zhang et al., 2018). As the issue of interpretability in RL gets more attention, a number of methods have been proposed to generate saliency maps to explain the decision making of RL agents. Existing saliency map methods in RL either used gradients to estimate the influence of input features on the output (Wang et al., 2016) (gradient-based methods) or computed the saliency of an input feature by perturbing it and observing the change in output (Greydanus, Koul, Dodge, & Fern, 2018; Iyer et al., 2018; Puri et al., 2020) (perturbation-based methods). Gradient-based methods can compute saliency maps efficiently with backpropagation. However, the quality of these gradient-based saliency maps was generally poor (Rosynski, Kirchner, & Valdenegro-Toro, 2020). Perturbation-based methods are effective in highlighting the important features of the input, but at a significant computational cost, which can make them ineffective when deployed on systems with real-time constraints. As a result, existing RL agents cannot provide high interpretability in a computation-efficient manner.

Different from previous work proposing new saliency calculation methods, we focus on improving the natural interpretability of RL policies. Given an RL policy, we propose an approach of Distillation with selective Input Gradient Regularization (DIGR) that uses policy distillation and input gradient regularization to retrain a new policy. In our approach, input gradient regularization selectively regularizes gradient-based saliency maps of the policy to imitate its interpretable perturbation-based saliency maps. This allows the new RL policy to generate high-quality saliency

* Correspondence to: 2232 Social & Behavioral Sciences Gateway, University of California, Irvine, CA 92697, USA.
E-mail addresses: jinweix1@uci.edu (J. Xing), takashin@uci.edu (T. Nagata), xinyunz5@uci.edu (X. Zou), eneftci@uci.edu (E. Neftci), jkrichma@uci.edu (J.L. Krichmar).

https://doi.org/10.1016/j.neunet.2023.01.025
0893-6080/

maps with gradient-based methods and thus achieve both high interpretability and computational efficiency. At the same time, to ensure that input gradient regularization does not cause task performance degradation, we use policy distillation (Czarnecki et al., 2019) to constrain the output of the new RL policy to remain close to the original RL policy.

We evaluate our method in three different tasks, which include an object fetching task from MiniGrid (Chevalier-Boisvert, Willems, & Pal, 2018), Breakout from Atari games and CARLA Autonomous Driving (Dosovitskiy, Ros, Codevilla, Lopez, & Koltun, 2017). The results show that RL policies trained with our approach are able to achieve efficient interpretability while maintaining good task performance. Selective input gradient regularization also improves the robustness of RL policies to adversarial attacks. These two desired properties allow the RL policy to better adapt to real-world scenarios.

To summarize, we demonstrate a novel approach to improve the efficient interpretability and robustness on attacks on RL policies based on the utilization of saliency maps. Our approach increases the applicability of RL to real-world problems.

2. Background and motivation

2.1. Reinforcement learning

In reinforcement learning, agents learn to take actions in an environment that maximize their cumulative rewards. The environment is typically stated in the form of a Markov Decision Process (MDP), which is expressed in terms of the tuple (S, A, T, R) where S is the state space, A is the action space, T is the transition function and R is the reward function. At each time step t in the MDP, the agent takes an action a_t in the environment based on the current state s_t and receives a reward r_{t+1} and next state s_{t+1}. The goal of the agent is to find a policy $\pi(s)$ to select actions that maximize the discounted cumulative future rewards $r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$, where γ is the discount factor ranging from 0 to 1.

2.2. Policy distillation

Policy distillation (Czarnecki et al., 2019; Rusu et al., 2015) transfers knowledge from one teacher policy π_t to a student policy π_s by training the student policy to produce the same behavior as the teacher policy. This is normally achieved by supervised regression to minimize the following objective:

$$J = \mathbb{E}_{s\sim\pi_c}[D(\pi_t(s), \pi_s(s))], \tag{1}$$

where π_c is the control policy that interacts with the environment to produce states for training, and D is a distance metric. There are multiple choices for both π_c and D. For example, the control policy π_c could take the form of the teacher policy π_t or student policy π_s or even a combination of them. Suitable distance metrics could be mean squared error or Kullback–Leibler divergence (KL divergence).

2.3. Saliency map in RL

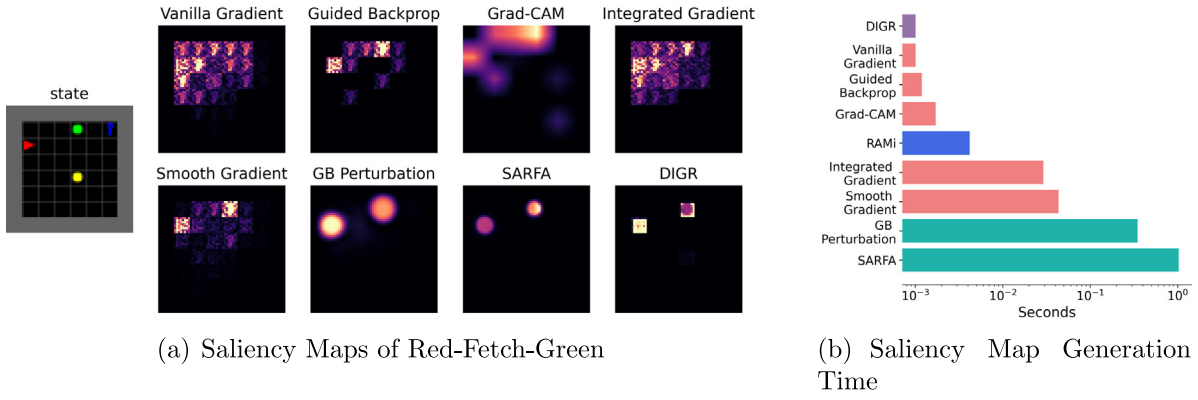
Addressing the interpretability of RL has attracted considerable attention in recent years. One common category of methods used visualization techniques such as saliency maps (Greydanus et al., 2018; Wang et al., 2016), attention mechanism (Mott, Zoran, Chrzanowski, Wierstra, & Jimenez Rezende, 2019) and object detection (Iyer et al., 2018) to explain deep neural network policies. Some other methods aimed to learn intrinsically interpretable policies in the formats of decision tree (Liu, Sun, Schulte, & Poupart, 2021; Silva, Gombolay, Killian, Jimenez, & Son, 2020), programming language (Verma, Murali, Singh, Kohli, & Chaudhuri, 2018) or logic formulations (Zhang, Li, Wang, & Tian, 2021). Besides the methods above, researchers also enhanced the understanding of RL decision making by using evidence-driven interpretation (Dao, Huff, & Lee, 2021; Dao, Mishra, & Lee, 2018), contrastive explanations (Lin, Lam, & Fern, 2020), counterfactual analysis (Atrey, Clary, & Jensen, 2020; Rupprecht, Ibrahim, & Pal, 2019) and state abstraction (Topin & Veloso, 2019). In this work, we focus on saliency map explanations.

Saliency map techniques are popular in computer vision and RL communities for interpreting deep neural networks. Gradient-based methods calculate the gradient of some function f with respect to inputs s based on the chain rule and then use the gradients to estimate the influence of input features on the output. In RL, one common approach is the Jacobian saliency map (Wang et al., 2016) which computes the saliency of input feature s_i as $|\frac{\partial f(s)}{\partial s_i}|$ where function f could be calculated from either the state–action value $Q(s, a)$ in Q-learning or the action distribution $\pi(s)$ in actor-critic methods. Other gradient-based visualization methods from the field of image classification are also explored (Greydanus et al., 2018; Rosynski et al., 2020) but most of them did not work well in the RL domain.

Perturbation-based methods compute the saliency of an input feature by perturbing (e.g. removing, altering or masking) the feature and observing the change in output. Given a state input s, a perturbed state s' could be generated by inducing a perturbation on input feature s_i. The approach of computing the change in output caused by the perturbation may vary based on the form of RL agent. For example, in Q-learning, the network output is a scalar and thus the saliency of s_i could be defined as $|Q(s, a) - Q(s', a)|$. In actor-critic methods, the saliency of s_i could be defined as $D_{KL}(\pi(s) \parallel \pi(s'))$ which is the KL divergence between action distributions before and after the perturbation. Alternatively, Greydanus et al. (2018) considered the output of actor as a vector and computed the saliency as $\frac{1}{2}\|\pi(s) - \pi(s')\|^2$. Puri et al. (2020) further proposed an approach of Specific and Relevant Feature Attribution (SARFA) to address the specificity and relevance in perturbation-based saliency maps.

2.4. Motivation

We first introduce a simple fetching-object task in MiniGrid and demonstrate the results of different saliency map methods on this task to motivate our method. In the fetching-object task in MiniGrid, the environment is a room composed of 8×8 grids and 4 entities with unique colors. The red agent needs to locate and pick up the green object, while the yellow and blue objects are distractors. Based on the task rule, we name this task as Red-Fetch-Green. We first use PPO (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017) to train an RL policy to solve the task and then investigate the interpretability and computation efficiency of different saliency map methods to explain the policy. Examples of gradient-based (Vanilla Gradient (Simonyan et al., 2013), Guided Backprop (Springenberg, Dosovitskiy, Brox, & Riedmiller, 2014), Grad-CAM (Selvaraju et al., 2017), Integrated Gradient (Sundararajan et al., 2017), Smooth Gradient (Smilkov et al., 2017)) and perturbation-based (Gaussian-Blur Perturbation (Greydanus et al., 2018) and SARFA (Puri et al., 2020)) saliency maps for Red-Fetch-Green are shown in Fig. 1(a). We also include an example of saliency maps generated by our DIGR approach for comparison. In general, perturbation-based saliency maps mainly demonstrate high saliency on task-relevant features (e.g. red agent and green target object) while gradient-based saliency maps are noisier and harder to interpret. However, the high quality of perturbation-based saliency maps is achieved with an increased cost of computation time. As shown in Fig. 1(b), perturbation-based saliency map takes more time to



(a) Saliency Maps of Red-Fetch-Green

(b) Saliency Map Generation Time

Fig. 1. (a). Different saliency maps on Red-Fetch-Green. All gradient-based saliency maps (Vanilla Gradient, Guided Backprop, Grad-CAM, Integrated Gradient and Smooth Gradient) produced by the PPO policy are noisy and show noticeable saliency on task-unrelated features. Gaussian-Blur Perturbation (GB Perturbation), SARFA saliency maps and saliency maps produced by DIGR approach demonstrate saliency on the red agent and green target object only. (b). The average time for each method to explain one action selection for states of Red-Fetch-Green during policy deployment with a CPU of Intel i7-9750H and a GPU of GeForce RTX 2080 Ti. We mark DIGR with purple and use red and green colors to represent normal gradient-based and perturbation-based saliency map methods. Blue color represents RAMi which is a decision-tree based policy and added as a baseline for comparing the computation efficiency of decision making explanation.

generate compared to gradient-based saliency maps and counterfactual analysis of 'Represent And Mimic'(RAMi) (Liu et al., 2021). The computation time of perturbation-based saliency maps is highly affected by the input size and policy network architectures. This makes it incompatible with many real-world tasks that require real-time interpretability such as autonomous driving. Thus, based on the result in Fig. 1, we find that normal gradient-based saliency maps are computationally more efficient but hard to interpret while perturbation-based saliency maps are more interpretable but come with a higher computation cost during deployment. This finding motivates us to think about how we can keep the computation efficiency of gradient-based methods and the high interpretability of perturbation-based methods while avoiding their limitations, and thus propose DIGR.

How does DIGR generate interpretable saliency maps like perturbation-based methods while only requiring a short generation time as the most efficient Vanilla Gradient saliency maps? Is it possible for us to use gradient-based methods such as Vanilla Gradient method to generate high-quality saliency maps as those from perturbation-based methods? We answer these questions in the next section.

3. Method

Our approach to achieving both computational efficiency and high interpretability in RL is to produce a policy whose gradient-based saliency maps are comparable to those of perturbation-based methods. To achieve this, given a trained RL policy, we set its perturbation-based saliency maps as supervisory signals and update the weights of the policy so that its gradient-based saliency maps match the perturbation-based saliency maps. Since the computations involved in gradient-based saliency maps are differentiable, we can use stochastic gradient descent to conduct the training. The idea of optimizing gradient-based saliency maps has a close connection with input gradient regularization which imposes constraints on how input gradients behave. For example, Ross and Doshi-Velez (2018) penalized input gradients based on an expert annotation to prevent the network from "attending" to certain parts of the input in an image classification task. Inspired by this, the training of the gradient-based saliency map in our approach is conducted by selectively penalizing the gradients of input features that have low perturbation-based saliency.

One challenge of selective input gradient regularization is that optimizing gradient-based saliency maps may also affect the policy output and thus degrade the task performance. To avoid this, we conduct policy distillation to ensure that the new policy maintains the same task performance. We give a more formal introduction of our method below.

Given an RL policy π and input s, we define the function g as the method used in generating gradient-based saliency map M_g and function f as the method used in generating perturbation-based saliency map M_p. Both M_g and M_p have the same size as input s. Each element in the saliency map, M_{g_i} and M_{p_i}, are computed as

$$g(s,i,\pi) = \left| \sum_a \pi(a|s) \frac{\partial \pi(a|s)}{\partial s_i} \right| \qquad M_{g_i} = \frac{g(s,i,\pi)}{\max_{0\le j\le N} g(s,j,\pi)} \tag{2}$$

$$f(s,i,\pi) = D_{KL}(\pi(s) \parallel \pi(m(s,i))) \qquad M_{p_i} = \frac{f(s,i,\pi)}{\max_{0\le j\le N} f(s,j,\pi)} \tag{3}$$

where $g(s,i,\pi)$ and $f(s,i,\pi)$ compute the gradient-based and perturbation-based saliency values of input feature s_i given policy π. These saliency values are then normalized between 0 and 1 to form saliency maps that contain N elements in each map. In this work, perturbation function m induces a Gaussian blur on the input with the input feature of interest s_i as the center (Greydanus et al., 2018). It is worth mentioning that, besides perturbation-based saliency maps, DIGR could be easily extended to utilize other saliency data (e.g. saliency maps from expert annotation) as supervisory signals. In this work, we focus on using perturbation-based saliency maps for input gradient regularization as they show high interpretability and can be computed as long as we have access to the policy and states.

After introducing the process of generating two types of saliency maps given an RL policy and state input, we introduce how they are used in DIGR. Given a trained RL policy π_t, DIGR aims to produce a new policy π_θ with parameters θ that can generate interpretable saliency maps using a gradient-based method. Given a state input s, the saliency map could differ based on the generation method (gradient-based vs perturbation-based) and the policy (π_t vs π_θ) used to generate them. For clarity, we define these 4 types of saliency maps as ${M_g}^t$, ${M_g}^\theta$, ${M_p}^t$, ${M_p}^\theta$ where the subscript of g represents gradient-based saliency maps and p represents perturbation-based saliency maps. The superscript of t represents the saliency map is generated by the original teacher policy π_t and θ represents the saliency map is

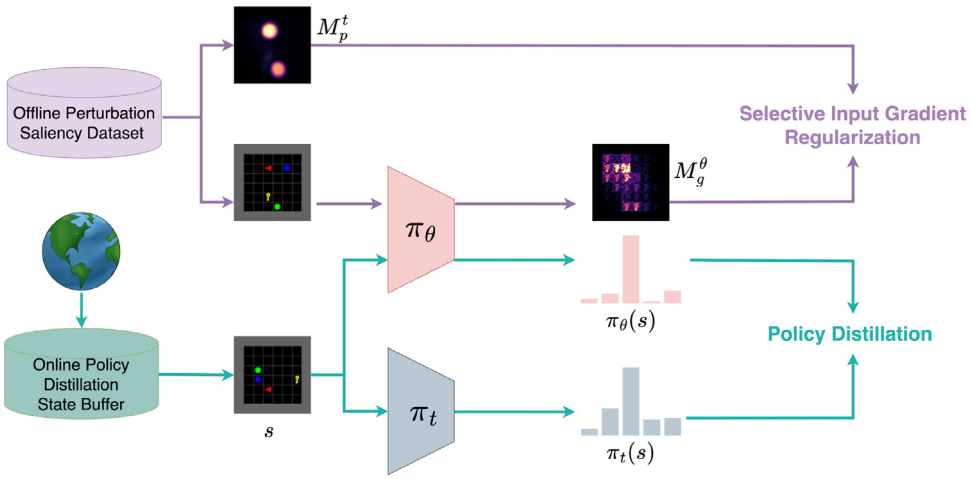


Fig. 2. Framework of our approach. Policy π_θ is used as the control policy and interacts with the environment. The experienced states are saved into a replay buffer and then sampled later for policy distillation. The training includes two objectives. The first objective is using input gradient regularization to regularize gradient-based saliency map M_g^θ based on the perturbation-based saliency map M_p^t. The second objective is using policy distillation to make sure the learning policy π_θ has the same behavior as the trained policy π_t.

generated by policy trained with DIGR (π_θ). In DIGR, we use the perturbation-based saliency maps generated by the teacher policy (${M_p}^t$) to provide supervisory signals to regularize the gradient-based saliency maps generated by the DIGR policy (${M_g}^\theta$). Then the loss function for input gradient regularization is

$$L = \mathbb{E}_{s\sim d_{\pi_\theta}}[\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}_{[0,\infty)}(\lambda - M_{p_i}^t) \times M_{g_i}^\theta] \tag{4}$$

where d_{π_θ} is the state distribution following policy π_θ and N is the number of input features in the saliency map. ${M_p}^t$ and ${M_g}^\theta$ have the same size and are both indexed by i. Threshold λ is used in the indicator function $\mathbb{1}$ to determine whether one input gradient should be penalized. The indicator function $\mathbb{1}$ returns 1 if $\lambda - M_{p_i}^t \geq 0$ and 0 otherwise. In other words, if the perturbation-based saliency for an input feature is below threshold λ, the loss penalizes its gradient-based saliency. This selective penalization allows the model to only keep high saliency on task-relevant features selected by the perturbation-based saliency maps.

The final loss function in our approach is a weighted combination of selective input gradient regularization and policy distillation. In practice, generating perturbation-based saliency maps online for input gradient regularization could be time-consuming and slow down the overall training. To address this, we build an offline perturbation saliency dataset D which contains states sampled from d_{π_t} and the corresponding perturbation-based saliency maps generated in advance. Because of the policy similarity brought by policy distillation, we use D to approximate d_{π_θ} for input gradient regularization. As a result, the loss function for DIGR is

$$\begin{aligned} L_{DIGR} = \underbrace{\mathbb{E}_{s\sim D}[\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}_{[0,\infty)}(\lambda - M_{p_i}^t) \times M_{g_i}^\theta]}_{\text{Input Gradient Regularization}} \\ + \alpha \underbrace{\mathbb{E}_{s\sim d_{\pi_\theta}}[D_{KL}(\pi_t(s) \parallel \pi_\theta(s))]}_{\text{Policy Distillation}} \end{aligned} \tag{5}$$

where α is a weighting parameter used to balance the loss of input gradient regularization and policy distillation. We show the complete architecture of our approach in Fig. 2.

4. Experimental results

We conduct experiments on three tasks including Red-Fetch-Green in MiniGrid, Breakout in Atari games and CARLA Autonomous Driving to demonstrate the effectiveness of our approach. In Red-Fetch-Green, the red agent needs to locate and pick up the green object while avoiding picking up other distractors in a room composed of 8 × 8 grids. In Breakout, the paddle is controlled to move at the bottom to ricochet the ball against the bricks and eliminate them for rewards. Besides these two tasks, we designed a CARLA Autonomous Driving task in which the agent needs to control an autonomous car driving on a highway while avoiding collisions. Since CARLA's simulation clock can be matched with the real time, we use it to show how the high quality and computation efficiency of our approach in interpreting RL policies could be important in real-world scenarios.

4.1. Setup

4.1.1. RL training

In our experiments, we first use PPO algorithm to train RL policies on Red-Fetch-Green, Breakout and CARLA Autonomous Driving. The trained RL policies, which are used to generate offline perturbation saliency datasets for input gradient regularization, also serve as the teacher policy in policy distillation and generate saliency maps for comparison. In all three tasks, we used similar network architectures composed of 3 convolutional layers and 2 linear layers but with different layer sizes. The trained RL policies achieved reasonably good performance in each task: The policy in Red-Fetch-Green solves the task with a success rate of 100%; the policy in Breakout achieves an average score of 320; the policy in CARLA Autonomous Driving could drive smoothly and learned to steer to avoid collision with other vehicles. We include more details of RL training in the Appendix.

4.1.2. Offline perturbation saliency dataset

To conduct selective input gradient regularization, we generate an offline perturbation saliency dataset by sampling states experienced by the trained RL policy π_t and generating the corresponding Gaussian-Blur perturbation saliency maps (Greydanus et al., 2018). The perturbation saliency datasets of Red-Fetch-Green, Breakout, and CARLA Autonomous Driving 1k, 10k, and 2.5k pairs of states and saliency maps. Although our method still

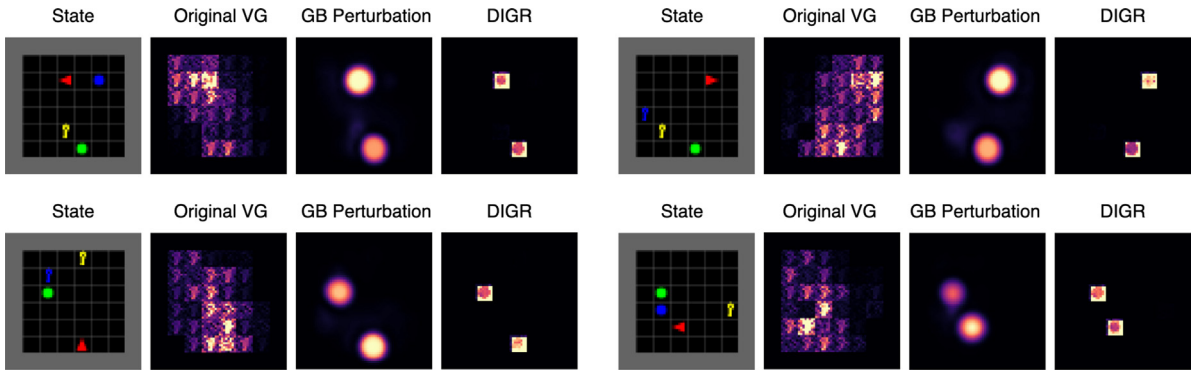


Fig. 3. Demonstration of our approach on Red-Fetch-Green. There are four sets of examples and each set includes a state, a Vanilla Gradient saliency map generated by the original policy (Original VG), a Gaussian-Blur perturbation-based saliency map (GB Perturbation) generated by the original policy and a Vanilla Gradient saliency map generated by the policy trained with DIGR. The annotation of DIGR on the figure refers to Vanilla Gradient saliency maps generated by the policy trained with DIGR. In all examples, GB Perturbation and DIGR saliency maps show high saliency on the red agent and green target while Original VG saliency maps are noisy and hard to interpret.

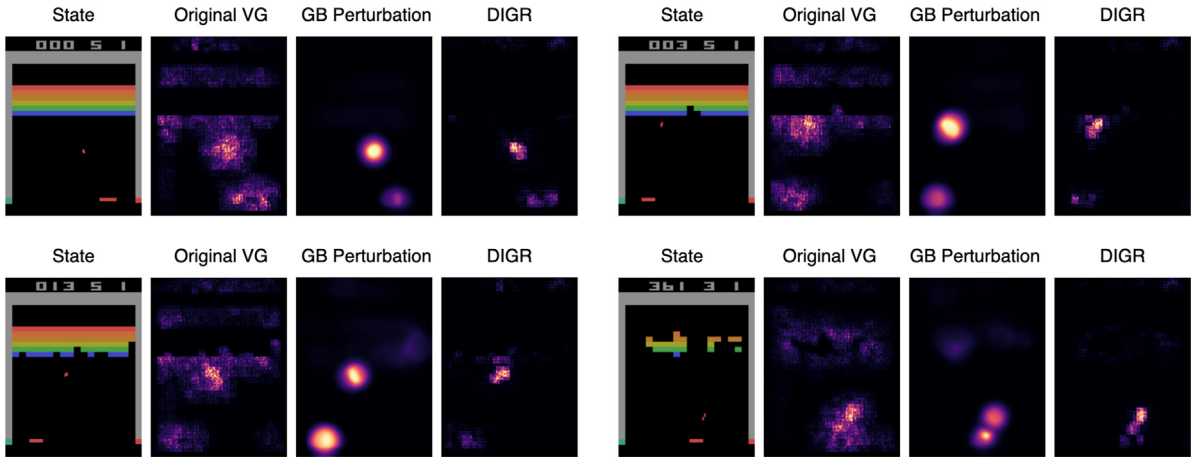


Fig. 4. Demonstration of our approach on Breakout. VG and GB Perturbation stand for Vanilla Gradient and Gaussian-Blur Perturbation. Both DIGR and Gaussian-Blur perturbation-based saliency maps demonstrate high saliency mainly on the paddle and ball while the Vanilla Gradient saliency maps generated by the original policy (Original VG) are noisier.

needs to generate perturbation-based saliency maps, the computation happens in the training stage without affecting the computation efficiency during deployment. Also, the computation problem could be mitigated by the limited size of the dataset (e.g. 1k, 10k, and 2.5k states in Red-Fetch-Green, Breakout, and CARLA respectively) and the potential utilization of parallel computing with multiple machines.

4.1.3. DIGR training

DIGR uses selective input gradient regularization and policy distillation to produce a new policy that achieves efficient interpretability while maintaining task performance. In all three experiments, we randomly initiate the new policy π_θ. To further stabilize the training, we consider the training of selective input gradient regularization and policy distillation as a multi-objective optimization problem and used the technique of projecting conflicting gradients (PCGrad) (Yu et al., 2020) to mitigate gradient interference. More hyperparameters of training are included in the Appendix.

4.2. Effectiveness via visual illustrative examples

The main goal of our approach is to allow RL policies to generate interpretable saliency maps with computationally efficient gradient-based methods. To demonstrate the effectiveness of our approach, we provide examples of the most computationally-efficient Vanilla Gradient saliency maps before and after our method, and Gaussian-Blur perturbation saliency maps that work as supervisory guidance in Figs. 3, 4, and 5.

Our results show that Vanilla Gradient saliency maps generated by original RL policies are noisy and hard to interpret. However, after the optimization with our approach, we can use the same saliency map method to generate much more interpretable saliency maps which reduces a large amount of unexplainable saliency and demonstrate high saliency on task-relevant features only. The saliency maps generated by our approach also have a close similarity to Gaussian-Blur perturbation-based saliency maps which demonstrates the successful saliency guidance. We provide more visual examples containing saliency maps produced by other gradient-based methods for comparison in the Appendix.

4.3. Importance of computational efficiency

In this section, we further show the importance of our approach by demonstrating that missing either computation efficiency or high interpretability makes it difficult to achieve interpretable RL in real-world scenarios. We take Autonomous Driving as an example and show the results of utilizing different saliency maps to explain a sequence of RL decision making in Fig. 6. In our experiments, the state of CARLA Autonomous Driving is a 128 $\times$ 128 RGB image taken every 0.05 s by a camera attached

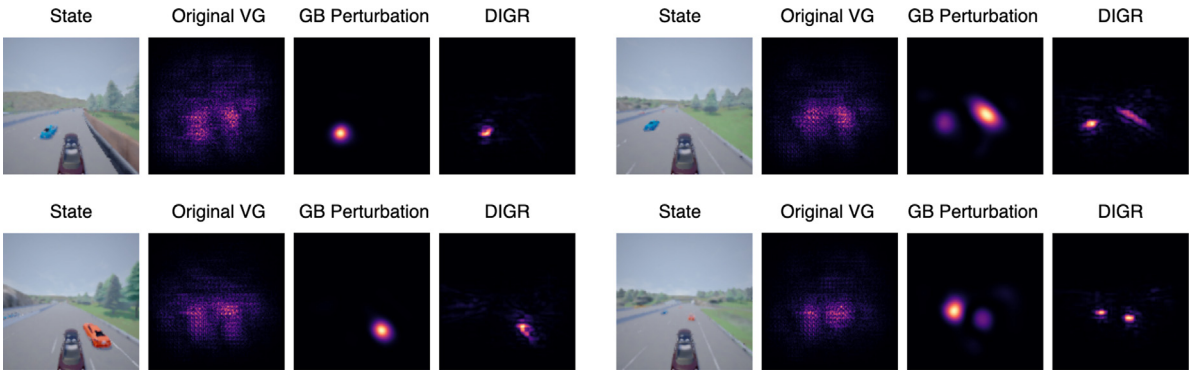


Fig. 5. Demonstration of our approach on CARLA Autonomous Driving. VG and GB Perturbation stand for Vanilla Gradient and Gaussian-Blur Perturbation. In the left two sets of examples, DIGR and GB Perturbation methods demonstrate high saliency on the vehicles that got close to the controlled vehicle. In the top-right example, DIGR and GB perturbation methods show high saliency on the vehicle and road curb. In the bottom-right example, DIGR and GB perturbation methods show high saliency on two vehicles ahead. DIGR and GB perturbation methods did not show saliency on the controlled vehicle because the controlled vehicle is always at the same region of the images for all states and is not salient to the performance. The saliency is demonstrated on other features that may lead to a collision and affect the performance. In all four sets of examples, Vanilla Gradient saliency maps generated by the original policy (Original VG) are very similar and hard to distinguish.

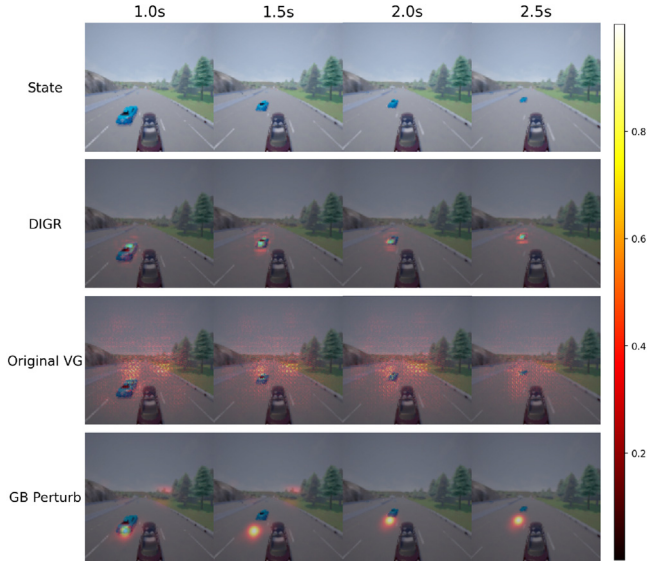


Fig. 6. Different types of saliency maps on a sequence of states in CARLA Driving. Vanilla Gradient saliency maps generated by the policy trained with DIGR always demonstrate high saliency on the traffic vehicles while Vanilla Gradient saliency maps generated by the original policy (original VG) are noisy and just show saliency in the center region of all states. Gaussian-Blur perturbation-based saliency maps show saliency behind the vehicle because of the computation delay. The bar on the right represents the mapping between saliency values and colors.

to the ego vehicle. Although Gaussian-Blur perturbation-based saliency maps show high interpretability as seen in Fig. 5, it takes 0.97 $\pm$ 0.02 s to generate one saliency map with a GPU of RTX 2080Ti. This means there is a delay of almost one second between meeting the state and the availability of the corresponding saliency map and all saliency maps for states experienced during the delay will be missed. In contrast to Gaussian-Blur perturbation-based saliency maps each takes 0.97 s to generate on average, Vanilla Gradient saliency maps are much more efficient to compute and take only 0.0021 $\pm$ 0.0001 s for each state with the same machine. However, Vanilla Gradient saliency maps generated by normal RL policies are hard to interpret and

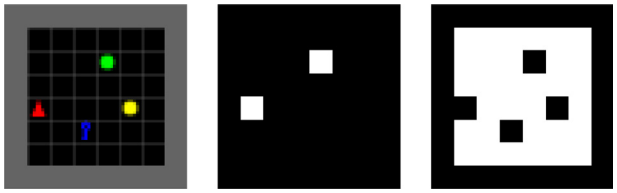


Fig. 7. **a**. An example state in the saliency dataset of Red-Fetch-Green. **b**. Regions whose saliency is important. **c**. Regions whose saliency is unimportant.

only our approach achieves both computation efficiency and high interpretability.

4.4. Saliency dataset and evaluation

Besides illustrative examples, we also aim to provide a quantitative evaluation of saliency maps generated by different approaches and thus introduce a new saliency dataset based on Red-Fetch-Green. Different from previous work that relies on expert annotations and classifies each state element as either an important or unimportant feature (Puri et al., 2020), we focus on features whose saliency importance is certain. There are six types of objects in Red-Fetch-Green including the red agent, the green target object, the blue and yellow distractors, gray walls, and black empty grids. Based on the roles of objects, we assume the red agent and green target are important features as they have the most important information required for optimal decision making and assume the empty tiles as unimportant features since they do not provide any information. The two distractors and gray walls are not included in the dataset because their influence on decision making is either uncertain or only exists in a small subset of state space. We collected 10k states in the saliency dataset and provide an example in Fig. 7.

To evaluate the quality of different saliency maps, we compute the average amount of important saliency and unimportant saliency in each saliency map. Furthermore, we also compare different saliency maps with Area under the Receiver Operating Characteristic Curve (AUC), which is a popular metric used to evaluate saliency maps (Iyer et al., 2018; Puri et al., 2020). As shown in Table 1, our approach keeps a comparable amount of

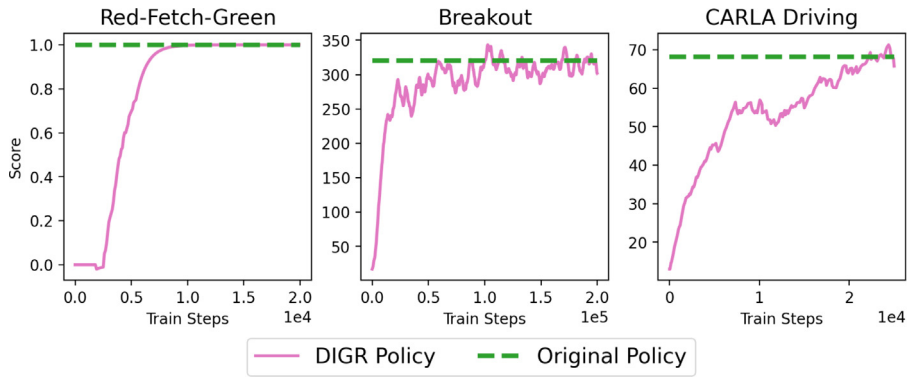


Fig. 8. The performance of DIGR policy could match the performance of the original policy.

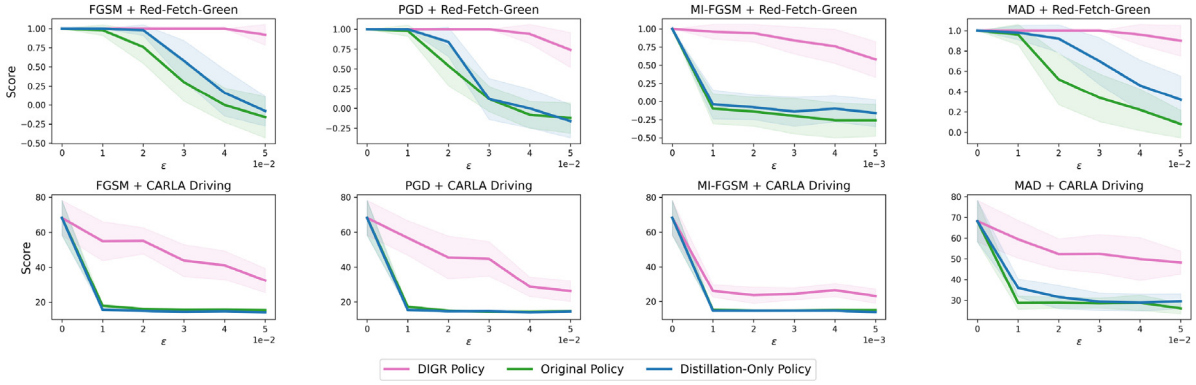


Fig. 9. Policies trained with DIGR achieve much stronger robustness to all four types of adversarial attacks (FGSM, PGD, MI-FGSM and MAD) compared to the policies trained with normal RL algorithms. Although policy distillation also helps robustness slightly, selective input gradient regularization makes the most contribution to the improved robustness. All results are averaged over 50 runs in Red-Fetch-Green and 20 runs in CARLA Autonomous Driving. The shaded area represents one standard deviation.

Table 1
Saliency results of Vanilla Gradient (VG), Guided Backpropagation (Guided BP), Grad-CAM, Smooth Gradient (Smooth G), Integrated Gradient (Integrated G), Gaussian-Blur Perturbation (GB Perturbation), SARFA of the original policy and Vanilla Gradient of DIGR policy on Red-Fetch-Green. Our method keeps a comparable amount of important saliency, reduces all unimportant saliency, and achieves the highest AUC.

	Saliency on Red-Fetch-Green		
	Important	Unimportant	AUC
VG	56.04	278.10	0.840
Guided BP	**82.84**	35.67	0.993
Grad-CAM	43.12	364.97	0.686
Smooth G	**83.05**	84.76	0.991
Integrated G	67.79	232.09	0.900
GB Perturbation	**86.11**	77.81	0.989
SARFA	58.40	42.17	0.895
DIGR	**72.52**	**0.00**	**0.997**

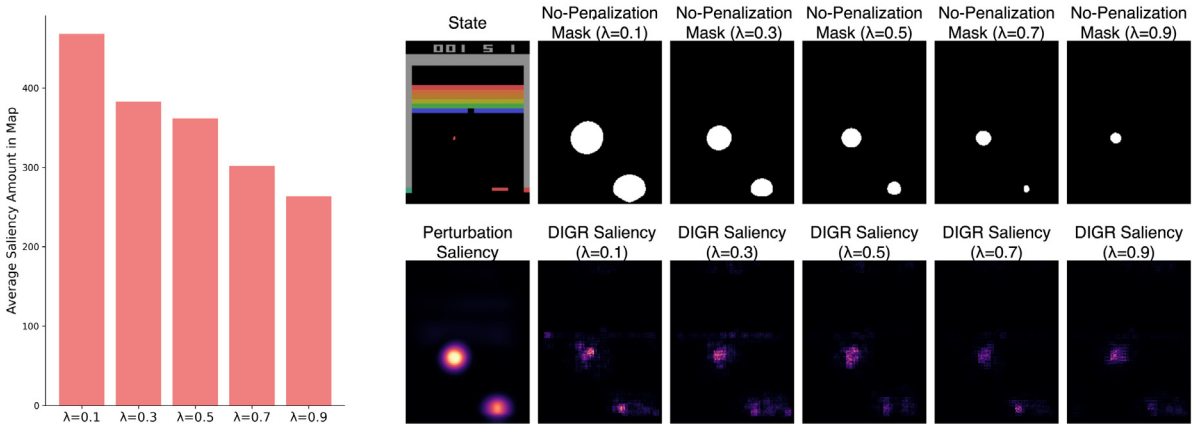
important saliency, reduces all unimportant saliency and achieves the highest AUC compared with other approaches. The decreased amount of unimportant saliency is in line with our expectation since our approach works by penalizing the saliency that is not helpful for interpretation. As a result, our approach utilizes gradient-based and perturbation-based saliency maps for training and finally achieves even better saliency maps.

4.5. Policy performance maintenance

The objective of optimizing gradient-based saliency maps may change the action selection of the original policy and thus cause the policy performance to degrade. In DIGR, we use policy distillation to constrain the output of the new RL policy to remain close to the original policy. To verify its effectiveness, we plot the performance of DIGR policy during training and compare it with the results of the original policy. As seen in Fig. 8, the policy trained with our approach could achieve similar performance as the original policy.

4.6. Improved robustness to attacks

Due to the importance of robustness of neural networks (Carlini & Wagner, 2017; Cheney, Schrimpf, & Kreiman, 2017; Zheng, Song, Leung, & Goodfellow, 2016) and recent research findings of a deep entanglement between adversarial attacks and interpretability of deep neural network (DNN) models (Ignatiev, Narodytska, & Marques-Silva, 2019; Tao, Ma, Liu, & Zhang, 2018), we are also interested in DIGR's influence on policy's robustness to attacks. To study that, we evaluate the robustness of RL policies before and after applying DIGR to four types of adversarial attacks including Fast Gradient Sign Method (FGSM) (Huang, Papernot, Goodfellow, Duan, & Abbeel, 2017), Projected Gradient Descent



(a) Average saliency amount in each DIGR saliency map

(b) The influence of different λ on the range of input gradient penalization and the resulting DIGR saliency maps

Fig. 10. (a). Average saliency amount in each DIGR saliency map. As λ increases, the average saliency amount decreases. (b). No-Penalization masks and DIGR saliency maps generated by choosing λ of 0.1, 0.3, 0.5, 0.7 and 0.9. In No-Penalization Masks, white pixels represent the region of input features whose input gradients will not be penalized. As λ increases, more input gradients will be penalized while the DIGR saliency maps generated with different λ still demonstrate high similarity.

Table A.2
PPO training hyperparameters for three tasks.

Hyperparameters	Red-Fetch-Green	Breakout	CARLA driving
γ	0.99	0.99	0.999
λ in GAE	0.95	0.95	0.95
Entropy bonus coefficient	0.01	0.01	0.01
Value loss coefficient	0.5	0.5	0.5
Gradient clipping	0.5	0.5	0.5
PPO clip range	0.2	0.2	0.2
Learning rate	0.001	0.0002	0.0002
Total timesteps	10M	20M	1M
# environments	16	8	1
# timesteps per rollout	128	128	1000
# epochs per rollout	4	4	4
# minibatches per rollout	8	4	4
Frame stack	1	2	1

Table A.3
DIGR hyperparameters for three tasks.

Hyperparameters	Red-Fetch-Green	Breakout	CARLA driving
Saliency threshold	0.1	0.1	0.1
Weighting parameter α	0.01	0.01	0.01
Learning rate	0.001	0.001	0.0002
Optimizer	Adam	Adam	RMSprop
Online state buffer size	10k	10k	10k
# perturbation-based saliency maps	1k	10k	2.5k

(PGD) (Madry, Makelov, Schmidt, Tsipras, & Vladu, 2018), Momentum Iterative Fast Gradient Sign Method (MI-FGSM) (Dong et al., 2018) and Maximum Action Difference (MAD) (Zhang et al., 2020) in Red-Fetch-Green and CARLA Autonomous Driving tasks. Since both policy distillation and input gradient regularization in our approach could affect the robustness of RL policies, we further include an ablation study by conducting policy distillation only to understand their own influence on robustness. As shown in Fig. 9, our approach significantly improves the robustness of RL policies. Although policy distillation also improves the robustness slightly, selective input gradient regularization contributes the most to the significant robustness gains.

4.7. Sensitivity to saliency threshold

The choice of a saliency threshold is an important hyperparameter that determines which input features the gradients should penalize. To investigate the sensitivity of DIGR to saliency threshold λ, we conduct a comparison analysis on the Atari game of Breakout by choosing different λ of 0.1, 0.3, 0.5, 0.7 and 0.9. As shown in Fig. 10(a), the average amount of saliency in each saliency map decreases as λ increases. This is because increasing λ will cause more input gradients to be penalized as shown by the No-Penalization Masks in Fig. 10(b). DIGR saliency maps generated by different λ demonstrate high structural similarity. We conjecture the reason is that although more input gradients are penalized as λ increases, the input gradients on the ball and paddle are not completely affected when choosing λ of 0.1, 0.3, 0.5 and 0.7. When increasing λ to 0.9, the input gradients on the paddle is completely penalized and thus the resulting DIGR saliency map shows less saliency on the paddle compared to other DIGR saliency maps. Also, policy distillation could play a role in restoring input gradients on input features that are essential for action selection. This could mitigate the influence of penalizing too many input gradients with a very high λ. As a result, as shown in Fig. 10, DIGR is robust to the hyperparameter of saliency threshold.

5. Conclusion

We propose an approach called DIGR to improve the efficient interpretability of RL by retraining a policy with selective input gradient regularization and policy distillation. Our approach allows RL policies to generate highly interpretable saliency maps with computationally efficient gradient-based methods. We further show that our approach is able to improve the robustness of RL policies to multiple adversarial attacks. Interpretable decision-making and robustness to attacks are two challenges in deploying RL to real-world systems. We believe our approach could help to build trustworthy agents and benefit the deployment of RL policies in practice.

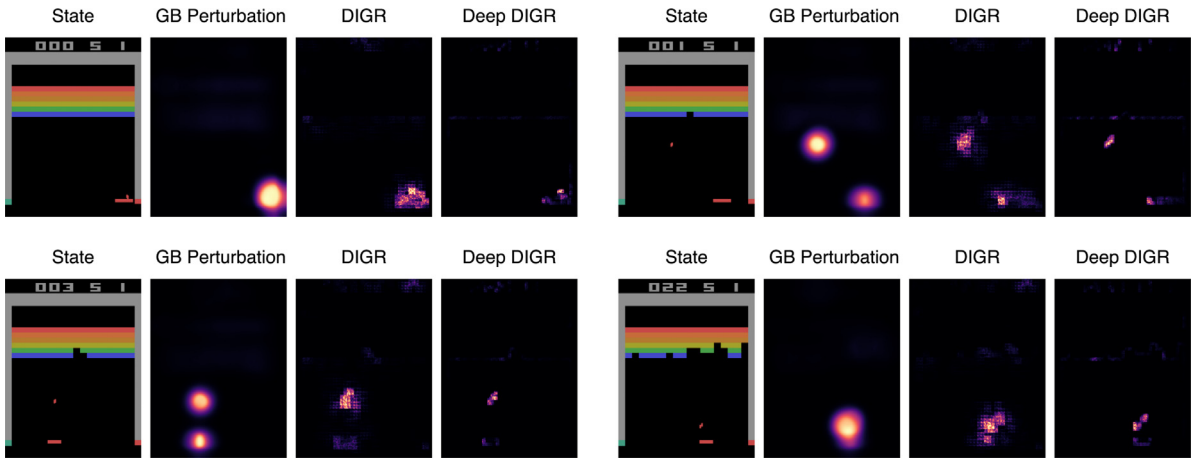


Fig. B.11. Examples of states, perturbation-based saliency maps and DIGR saliency maps with original and deep policy architectures. GB Perturbation stand for Gaussian-Blur perturbation. DIGR represent the saliency maps generated by the original policy architecture. Deep DIGR represent the saliency maps generated by the policy with deep architecture.

Table B.4
Architecture details of the original policy and deeper policy in Breakout.

Layers	Original policy	Deep policy
Convolutional layers (kernel_size, stride, padding, dilation)	(8,4,0,1) (4,2,0,1) (3,1,0,1)	(4,2,0,1) (4,2,0,1) (4,2,0,1) (3,1,0,1) (3,1,0,1)
Linear layers (input_size, output_size)	(22 528, 512) (512, 1)	(17 920, 1024) (1024, 512) (512, 1)

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This material is partially based upon work supported by the United States Air Force and DARPA under Contract No. FA8750-18-C-0103, and Air Force Office of Scientific Research under Contract No. FA9550-19-1-0306. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force and DARPA. Authors are also thankful to computing resources provided by CHASE-CI under NSF, USA Grant CNS-1730158.

Appendix A. Experiment details and hyperparameters

We conduct experiments on three tasks including Red-Fetch-Green in MiniGrid, Breakout in Atari games and CARLA Autonomous Driving to demonstrate the effectiveness of our approach. To generate the trained reinforcement learning (RL) policies, we use Proximal Policy Optimization (PPO) as the training algorithm and list hyperparameters in Table A.2.

When applying our approach, we need to choose the saliency threshold to select saliency that will be penalized, weighting parameter α to balance selective input gradient regularization and policy distillation, learning rate and online state buffer size. Furthermore, in practice, since we conduct input gradient regularization based on perturbation-based saliency maps collected in advance instead of producing them from the state buffer, we also need to choose the number of perturbation-based saliency maps in the offline perturbation saliency dataset. We list these hyperparameters in Table A.3.

In this work, we use multiple saliency map methods including Vanilla Gradient, Guided Backpropagation, Grad-CAM, Integrated Gradient, Smooth Gradient, Gaussian-Blur Perturbation and SARFA. For Grad-CAM, we report the saliency maps extracted from the last convolutional layer. For Integrated Gradient method, we use 50 interpolation steps to calculate the saliency maps. In our experiments, Smooth Gradient saliency maps are produced by applying SmoothGrad on Guided Backprop saliency maps. For SmoothGrad, we set the noise scale σ as 0.15 and the number of samples as 20. One important hyperparameter in Gaussian-Blur perturbation-based method is the radius size of the perturbation. Based on the size of the state images and features, we set the radius as 4, 8, 5 in Red-Fetch-Green, Breakout and CARLA Autonomous Driving. SARFA is based on Gaussian-Blur perturbation and thus shares the same hyperparameters.

Appendix B. Deeper RL policies

To verify the effectiveness of DIGR on deeper RL policies, we test to use a deeper neural network architecture for DIGR policy in Breakout. The teacher policy π_t trained with PPO is still composed of 3 convolutional layers and 2 linear layers while the DIGR policy π_θ consists of 5 convolutional layers and 3 linear layers (Table B.4). As shown in Fig. B.11, DIGR saliency maps generated by original policy and deeper policy have high similarity. This demonstrates that DIGR works with deeper policy architectures. We also find that saliency maps generated by deeper policy have more fine-grained saliency which could be caused by the smaller convolutional kernel size.

Appendix C. Additional experiment results

In this section, we provide more examples to demonstrate the effectiveness of DIGR. Besides Vanilla Gradient and Gaussian-Blur perturbation-based saliency maps, we also provide Guided Backprop, Grad-CAM, Integrated Gradient and Smooth Gradient saliency maps for comparison. All these saliency maps except DIGR are produced by the policy trained with PPO algorithm. The results on Red-Fetch-Green, Breakout and CARLA Autonomous Driving are shown in Figs. C.12–C.14.

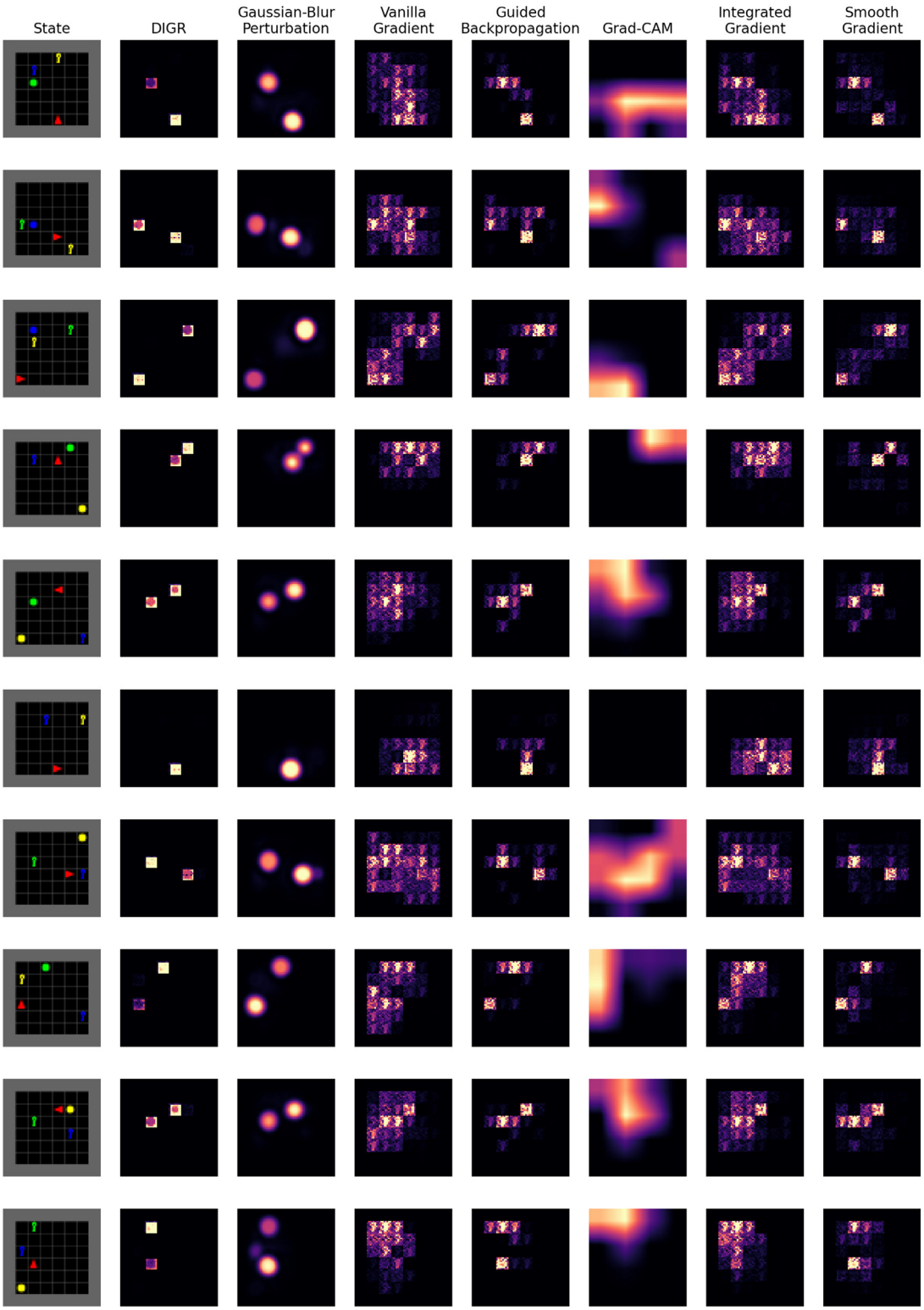


Fig. C.12. Supplementary saliency map examples on Red-Fetch-Green.

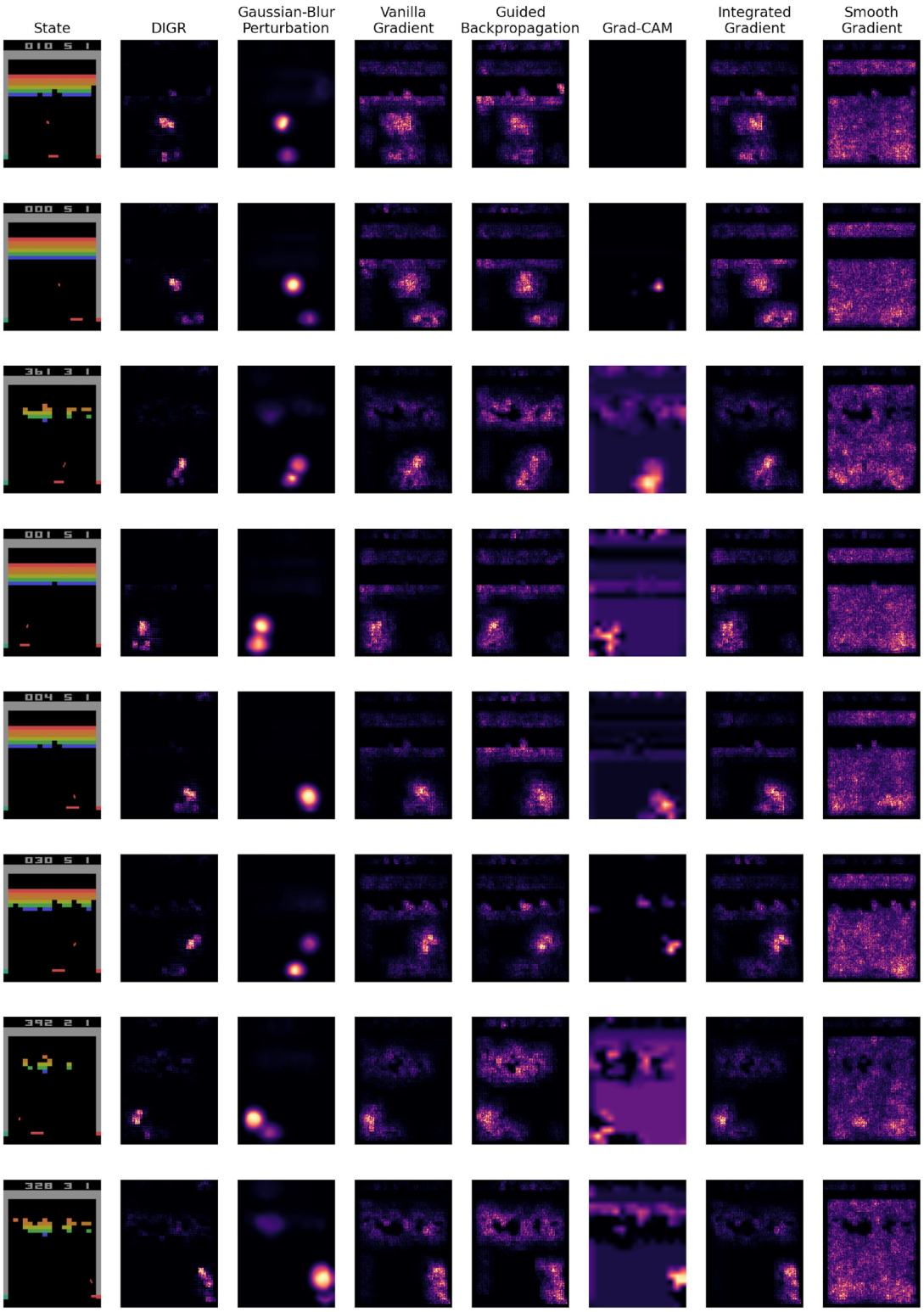


Fig. C.13. Supplementary saliency map examples on Breakout.

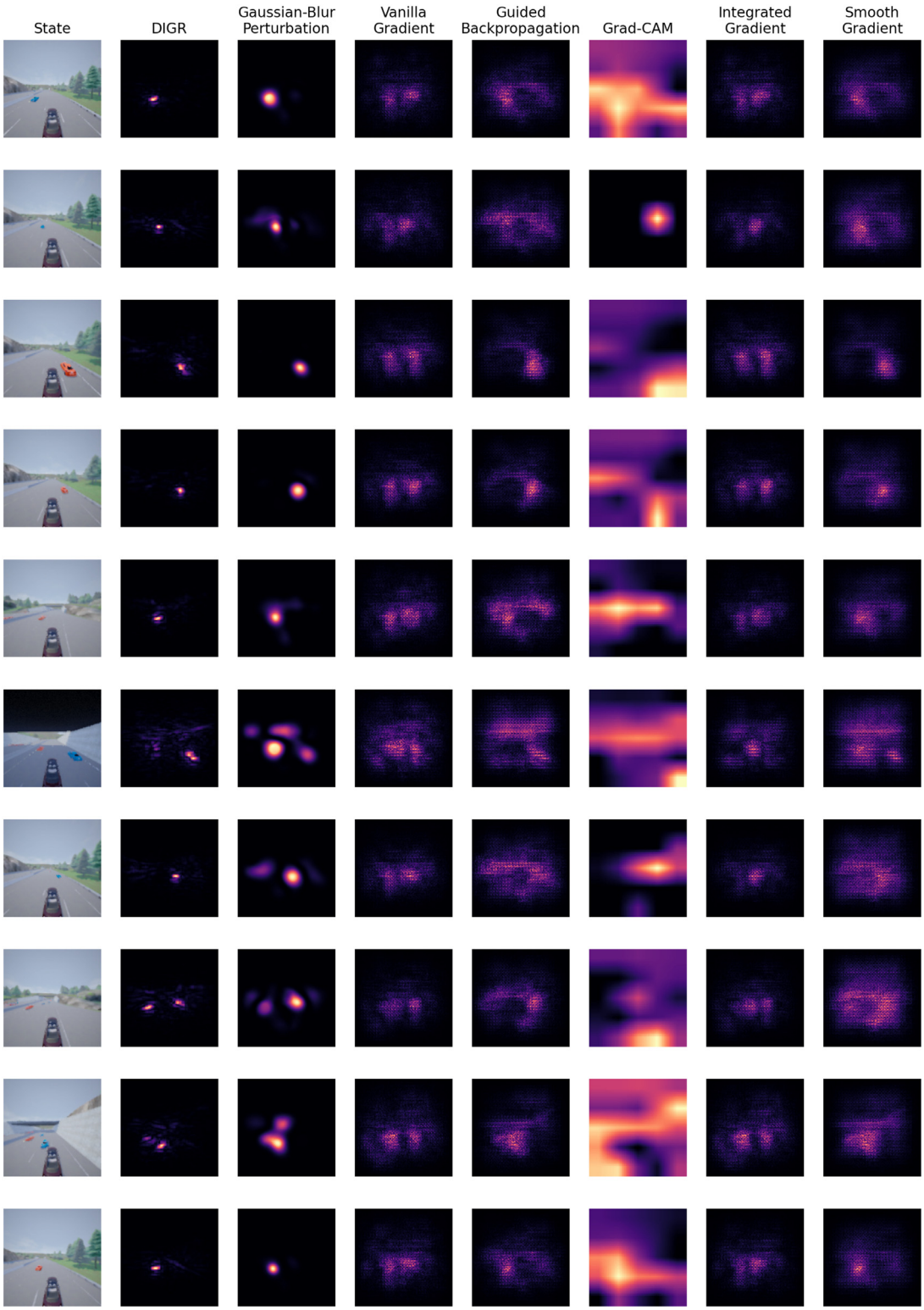


Fig. C.14. Supplementary saliency map examples on CARLA Autonomous Driving.

References

Atrey, A., Clary, K., & Jensen, D. (2020). Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. In *International conference on learning representations*. URL: https://openreview.net/forum?id=rkl3m1BFDB.

Bojarski, M., Choromanska, A., Choromanski, K., Firner, B., Ackel, L. J., Muller, U., et al. (2018). Visualbackprop: Efficient visualization of cnns for autonomous driving. In *2018 IEEE international conference on robotics and automation* (pp. 4701–4708). IEEE.

Carlini, N., & Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 IEEE symposium on security and privacy (Sp)* (pp. 39–57). IEEE.

Cheney, N., Schrimpf, M., & Kreiman, G. (2017). On the robustness of convolutional neural networks to internal architecture and weight perturbations. arXiv preprint arXiv:1703.08245.

Chevalier-Boisvert, M., Willems, L., & Pal, S. (2018). Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid.

Czarnecki, W. M., Pascanu, R., Osindero, S., Jayakumar, S., Swirszcz, G., & Jaderberg, M. (2019). Distilling policy distillation. In *The 22nd international conference on artificial intelligence and statistics* (pp. 1331–1340). PMLR.

Dao, G., Huff, W. H., & Lee, M. (2021). Learning sparse evidence-driven interpretation to understand deep reinforcement learning agents. In *2021 IEEE symposium series on computational intelligence* (pp. 1–7). IEEE.

Dao, G., Mishra, I., & Lee, M. (2018). Deep reinforcement learning monitor for snapshot recording. In *2018 17th IEEE international conference on machine learning and applications* (pp. 591–598). IEEE.

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., et al. (2018). Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 9185–9193).

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). CARLA: An open urban driving simulator. In *Conference on robot learning* (pp. 1–16). PMLR.

Ferdowsi, A., Challita, U., Saad, W., & Mandayam, N. B. (2018). Robust deep reinforcement learning for security and safety in autonomous vehicle systems. In *2018 21st international conference on intelligent transportation systems* (pp. 307–312). IEEE.

Fong, R. C., & Vedaldi, A. (2017). Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE international conference on computer vision* (pp. 3429–3437).

Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International conference on machine learning* (pp. 1587–1596). PMLR.

Greydanus, S., Koul, A., Dodge, J., & Fern, A. (2018). Visualizing and understanding atari agents. In *International conference on machine learning* (pp. 1792–1801). PMLR.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning* (pp. 1861–1870). PMLR.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., & Abbeel, P. (2017). Adversarial attacks on neural network policies. arXiv preprint arXiv:1702.02284.

Ignatiev, A., Narodytska, N., & Marques-Silva, J. (2019). On relating explanations and adversarial examples. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems, Vol. 32*. Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper/2019/file/7392ea4ca76ad2fb4c9c3b6a5c6e31e3-Paper.pdf.

Iyer, R., Li, Y., Li, H., Lewis, M., Sundar, R., & Sycara, K. (2018). Transparency and explanation in deep reinforcement learning neural networks. In *Proceedings of the 2018 AAAI/ACM conference on AI, ethics, and society* (pp. 144–150).

Le, N., Rathour, V. S., Yamazaki, K., Luu, K., & Savvides, M. (2021). Deep reinforcement learning in computer vision: a comprehensive survey. *Artificial Intelligence Review*, 1–87.

Li, H., Liu, D., & Wang, D. (2017). Manifold regularized reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(4), 932–943.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Lin, Z., Lam, K.-H., & Fern, A. (2020). Contrastive explanations for reinforcement learning via embedded self predictions. arXiv preprint arXiv:2010.05180.

Liu, G., Sun, X., Schulte, O., & Poupart, P. (2021). Learning tree interpretation from object representation for deep reinforcement learning. *Advances in Neural Information Processing Systems*, *34*, 19622–19636.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *International conference on learning representations*. URL: https://openreview.net/forum?id=rJzIBfZAb.

McAllister, R., Kahn, G., Clune, J., & Levine, S. (2019). Robustness to out-of-distribution inputs via task-aware generative uncertainty. In *2019 international conference on robotics and automation* (pp. 2083–2089). IEEE.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Mott, A., Zoran, D., Chrzanowski, M., Wierstra, D., & Jimenez Rezende, D. (2019). Towards interpretable reinforcement learning using attention augmented agents. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems, Vol. 32*. Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper/2019/file/e9510081ac30ffa83f10b68cde1cac07-Paper.pdf.

Nguyen, A., Yosinski, J., & Clune, J. (2019). Understanding neural networks via feature visualization: A survey. In *Explainable AI: Interpreting, explaining and visualizing deep learning* (pp. 55–76). Springer.

Puri, N., Verma, S., Gupta, P., Kayastha, D., Deshmukh, S., Krishnamurthy, B., et al. (2020). Explain your move: Understanding agent actions using specific and relevant feature attribution. In *International conference on learning representations*. URL: https://openreview.net/forum?id=SJgzLkBKPB.

Ross, A., & Doshi-Velez, F. (2018). Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the AAAI conference on artificial intelligence*.

Rosynski, M., Kirchner, F., & Valdenegro-Toro, M. (2020). Are gradient-based saliency maps useful in deep reinforcement learning? In *"I Can'T Believe It's Not Better!"neurIPS 2020 workshop*. URL: https://openreview.net/forum?id=ZF4KyC2zz6x.

Rupprecht, C., Ibrahim, C., & Pal, C. J. (2019). Finding and visualizing weaknesses of deep reinforcement learning agents. arXiv preprint arXiv:1904.01318.

Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., et al. (2015). Policy distillation. arXiv preprint arXiv:1511.06295.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision* (pp. 618–626).

Silva, A., Gombolay, M., Killian, T., Jimenez, I., & Son, S.-H. (2020). Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In *International conference on artificial intelligence and statistics* (pp. 1855–1865). PMLR.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., & Wattenberg, M. (2017). Smoothgrad: removing noise by adding noise. arXiv preprint arXiv:1706.03825.

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806.

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. In *International conference on machine learning* (pp. 3319–3328). PMLR.

Tao, G., Ma, S., Liu, Y., & Zhang, X. (2018). Attacks meet interpretability: Attribute-steered detection of adversarial samples. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems, Vol. 31*. Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper/2018/file/b994697479c5716eda77e8e9713e5f0f-Paper.pdf.

Topin, N., & Veloso, M. (2019). Generation of policy-level explanations for reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 2514–2521).

Verma, A., Murali, V., Singh, R., Kohli, P., & Chaudhuri, S. (2018). Programmatically interpretable reinforcement learning. In *International conference on machine learning* (pp. 5045–5054). PMLR.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., et al. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, *575*(7782), 350–354.

Wang, D., Ha, M., & Zhao, M. (2022). The intelligent critic framework for advanced optimal control. *Artificial Intelligence Review*, *55*(1), 1–22.

Wang, R., Lehman, J., Rawal, A., Zhi, J., Li, Y., Clune, J., et al. (2020). Enhanced poet: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *International conference on machine learning* (pp. 9940–9951). PMLR.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning* (pp. 1995–2003). PMLR.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., & Finn, C. (2020). Gradient surgery for multi-task learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems, Vol. 33* (pp. 5824–5836). Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper/2020/file/3fe78a8acf5fda99de95303940a2420c-Paper.pdf.

Zhang, J., Bargal, S. A., Lin, Z., Brandt, J., Shen, X., & Sclaroff, S. (2018). Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, *126*(10), 1084–1102.

Zhang, H., Chen, H., Xiao, C., Li, B., Liu, M., Boning, D., et al. (2020). Robust deep reinforcement learning against adversarial perturbations on state observations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, H. Lin (Eds.), *Advances in neural information processing systems* (pp. 21024–21037). Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper/2020/file/f0eb6568ea114ba6e293f903c34d7488-Paper.pdf.

Zhang, L., Li, X., Wang, M., & Tian, A. (2021). Off-policy differentiable logic reinforcement learning. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 617–632). Springer.

Zheng, S., Song, Y., Leung, T., & Goodfellow, I. (2016). Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4480–4488).