Learn to Interpret Atari Agents

Zhao Yang [1] **Song Bai** [1] **Li Zhang** [1] **Philip H.S. Torr** [1]

Abstract

Deep Reinforcement Learning (DeepRL) models surpass human-level performance in a multitude of tasks. Standing in stark contrast to the stellar performance is the obscure nature of the learned policies. The direct mapping from states to actions makes it hard to interpret the rationale behind the decision making of agents.

In contrast to previous *a-posteriori* methods of visualising DeepRL policies, we propose an end-to-end trainable framework based on Rainbow, a representative Deep Q-Network (DQN) agent. Our method automatically detects important regions in the input domain, which enables characterization of general strategy and explanation for non-intuitive behaviors. Hence, we call it Region Sensitive Rainbow (RS-Rainbow). RS-Rainbow utilises a simple yet effective mechanism to incorporate innate visualisation ability into the learning model, not only improving the interpretability, but enabling the agent to leverage enhanced state representations for improved performance. Without extra supervision, specialised feature detectors focusing on distinct aspects of gameplay can be learned. Extensive experiments on the challenging platform of Atari 2600 demonstrates the superiority of RS-Rainbow. In particular, our agent achieves state of the art at just 25% of the training frames without massive large-scale parallel training.

1. Introduction

Understanding deep neural networks (DNN) has been a long-standing goal of the community. Many efforts have focused on explaining CNN-based classification models (Krizhevsky et al., 2012; He et al., 2016) with human-interpretable visualisations (Simonyan et al., 2014; Zeiler & Fergus, 2014; Springenberg et al., 2015; Shrikumar et al., 2017; Fong & Vedaldi, 2017; Dabkowski & Gal, 2017). These methods exploit the class discriminative nature of the models to produce meaningful and concise class-specific visualisations.

With the advent of Deep Reinforcement Learning (DeepRL) (Mnih et al., 2013), there is an increasing interest in understanding deep representations in these new models. DeepRL combines the strong representation and approximation power of DNNs with the traditional RL objectives framed in the deep learning setting. In modern applications where a state is defined by high-dimensional data input, *e.g.*, Atari 2600 (Bellemare et al., 2013), the task of DeepRL divides into two essential sub-tasks, *i.e.*, generating low-dimensional representations on states and policy learning using such representations.

In this case, a CNN-based feature extractor no longer optimises for class discriminative visual patterns. Consequently, the visualisation techniques which capitalise on discriminative responses to visual variations no longer has suitably-defined, human-interpretable semantics. The approximation of an optimal policy not only maps low-dimensional features to decision suggestions in a black-box manner, but implicitly incorporates state information and modeling environment dynamics over several time steps. The black-box and sequential nature of the policy function makes explaining the policy of a DeepRL system inherently difficult.

Recently, many efforts have been devoted to explaining DeepRL models. Most of these methods (Mnih et al., 2015; Wang et al., 2016; Zahavy et al., 2016; Greydanus et al., 2018) are *a-posteriori*, only capable of explaining model outputs without inducing policy improvements during training with deduced knowledge, and heavily rely on human interventions. Some of them (Mnih et al., 2015; Zahavy et al., 2016) require carefully selected game-specific attributes and expert knowledge in the RL domain, and have limited accessibility for non-experts. Other vision-inspired methods (Mnih et al., 2015; Greydanus et al., 2018) utilise traditional saliency methods. Specifically, (Greydanus et al., 2018) adopts a data-driven approach for illustrating policy responses to a fixed input masking function, which requires hundreds of forward passes for visualising one frame.

In this work, we provide another perspective on tackling inscrutability, and propose Region Sensitive Rainbow (RS-

[1]University of Oxford, Oxford, UK. Correspondence to: Zhao Yang <zhao.yang@eng.ox.ac.uk>, Song Bai <songbai.site@gmail.com>, Li Zhang <lz@robots.ox.ac.uk>, Philip H.S. Torr <philip.torr@eng.ox.ac.uk>.

Preliminary work.

Rainbow) to improve *both the interpretability and performance* of DeepRL models. To this end, RS-Rainbow leverages a region-sensitive module to estimate the importance of different sub-regions on the screen, which is used to direct policy learning in end-to-end training. The selected regions involve distinctive patterns and objects, which when combined, complement each other in forming a representation of the state. When analysing across the time dimension, consistency in a particular detected pattern or type of objects describes a prominent aspect of the game, which illustrates one line of reasoning in the decision making of agents. With the region-sensitive module, we produce intuitive visualisations (see Fig. 1) in a single backward pass without human interventions or repetitive, costly propagations through the network.

The primary contribution of this work is to provide, to our best knowledge, the first learning-based approach for automatically interpreting DeepRL models. It requires no extra supervision and is end-to-end trainable. Moreover, it also possesses three advantages as

1) In contrast to previous methods (Zahavy et al., 2016; Greydanus et al., 2018), RS-Rainbow provides intuitive visualisations and interpretations at little extra cost without human interventions in the loop. Its region sensitivity sheds light on globally-coherent, high-level DeepRL policies.

2) Besides possessing interpretation capacity, extensive experimental results on the Atari 2600 platform (Bellemare et al., 2013) demonstrate that RS-Rainbow effectively improves policy learning. In comparison, previous a-posteriori methods are unable to bring performance enhancements.

3) Region-sensitive module, the core component of RS-Rainbow, is a simple and efficient plug-in. It can be potentially applied to most DQN-based models, and brings performance gains and a built-in visualisation advantage.

2. Background

In this section, we provide a brief review on Deep Q-Network (DQN) (Mnih et al., 2013; 2015), with a special focus on understanding and interpreting deep reinforcement learning models.

2.1. DQN and Rainbow

As an RL algorithm, DQN seeks to find a policy which maximises the long-term return of an agent acting in an environment with the convergence guarantee of a Bellman equation. DQN combines deep learning with the traditional off-policy, value-based Q-learning algorithm by employing DNN as a value approximation function and mean-squared error minimization as an equivalent form of temporal difference updating. Target network and experience replay are two key engineering inventions for stabilizing the training of DQN. Q value refers to the expected discounted return for executing a particular action in a given state and following the current policy. Given the Q values of all actions in a state, the optimal policy follows as taking the action with the highest Q value.

Rainbow (Hessel et al., 2018) incorporates many extensions over the original DQN (Mnih et al., 2013; 2015), each of which enhances a different aspect of the model. Such extensions include double DQN (van Hasselt et al., 2016), dueling DQN (Wang et al., 2016), priority experience replay (Schaul et al., 2016), multi-step learning (Sutton, 1988), distributional RL (Bellemare et al., 2017), and noisy nets (Fortunato et al., 2018). Double DQN addresses the over-estimation of Q in the target function. Dueling DQN decomposes the estimation of Q into separate estimations for state value V and action advantage A. Priority experience replay follows a prioritized sampling strategy that favors training data with high learning errors. Multi-step learning looks multiple steps ahead by replacing one-step rewards and states with their multi-step counterparts. Noisy net injects adaptable noise to linear layer outputs to introduce state-dependent exploration, which replaces the ϵ-greedy exploration in DQN. In distributional RL, Q is modeled by a distribution over a fixed support set of discrete values. With the duelling structure, this corresponds to modelling V and A respectively. The intuition is that learning a distribution over a support set works better than directly learning the expected value. The resulting Kullbeck-Leibler divergence loss enjoys convergence guarantee as the return distributions satisfy a Bellman equation.

2.2. Understanding DeepRL

Interpretation work in the traditional RL setting typically generates language explanations via first-order logic (Dodson et al., 2011; Elizalde et al., 2008; Khan et al., 2009; Hayes & Shah, 2017). These approaches rely on small state spaces and high-level state variables with human interpretable semantics. As such, they are not applicable to most DeepRL domains with large state spaces defined by high-dimensional, low-semantic features, such as the Atari 2600 domain (Bellemare et al., 2013).

In the context of DeepRL, (Mnih et al., 2015) and (Zahavy et al., 2016) propose to understand DQN policies via a Semi-Aggregated Markov Decision Process (SAMDP), which visualises hierarchical spatio-temporal abstractions in a DQN policy via t-SNE (Maaten & Hinton, 2008). In order to understand these abstractions, suitable game-specific attributes must be manually selected and recorded at each frame. It requires good judgment on suitable attributes beforehand. Extracting these attributes without API support on the Atari 2600 platform adds significant complexity. While

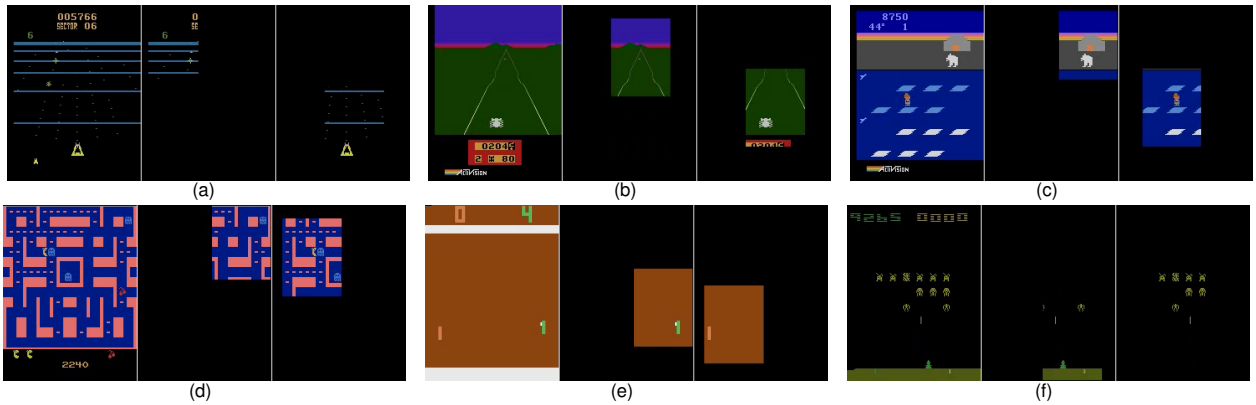


Figure 1. Visualising policies in games beam_rider (a), enduro (b), frostbite (c), ms_pacman (d), pong (e), and space_invaders (f). For each game, the left is the original screen, and the middle and the right each shows a salient region learned by RS-Rainbow, which captures the objects of interest.

high-level abstractions are informative to RL experts, a user without the relevant theoretical background may find them hard to interpret. The above aspects make SAMDP a less convenient visualisation tool for Atari agents.

(Greydanus et al., 2018) adopts perturbation-based saliency (Shrikumar et al., 2017) to visualise pixel importance in an asynchronous advantage actor-critic (A3C) model (Mnih et al., 2016). It applies a masking function on the input frame and observes the impact on the target output quantified by Euclidean distance. Perturbation-based methods can be computationally inefficient as each perturbation requires a separate forward pass through the network. With a stride of 5 in (Greydanus et al., 2018), hundreds of forward passes are still required for producing saliency on a single frame. As outlined by (Shrikumar et al., 2017), traditional saliency methods underestimate pixel importance when the pixel saturates its contribution to the target function. As discussed in Sec. 1, DeepRL models are not optimised for class discriminant features, which further limits the visualisation power of saliency (Simonyan et al., 2014; Zeiler & Fergus, 2014; Springenberg et al., 2015).

Next we describe RS-Rainbow, which is capable of learning region-discriminative insights for improving training and providing interpretations for learned policies.

3. Proposed Approach

In this section, we introduce our motivation in Sec. 3.1, then describe the architecture of RS-Rainbow in Sec. 3.2, and finally present its capability for visualisation in Sec. 3.3.

3.1. Motivation

Via gradient-based saliency (Simonyan et al., 2014), we first notice that different areas on screen weigh differently to the Q function. It connects to the fairly easy observation that humans play a game using sub-regions on the screen depending on the game state. Granted, an object with a functional role is more critical for earning rewards than the background. Furthermore, the same object can bear different levels of importance in different game states.

Thus, we are interested in two questions, *i.e.*, "can DQN end-to-end learn a region selection strategy for inference," and "how such a strategy affects policy learning." If the model learns a state-dependent distribution over regions (thus the objects within), then it can leverage more information from the right places and reduce interference from irrelevant areas.

3.2. Architecture

The complete architecture of RS-Rainbow is illustrated in Fig. 2, which consists of an image encoder, a region-sensitive module, and policy layers with a value stream and an advantage stream.

As in Rainbow (Hessel et al., 2018), our image encoder Φ is a three-layer CNN interleaved with ReLU nonlinearities (Nair & Hinton, 2010). At each time step t, a stack of four consecutive frames $\mathbf{S}$ of shape $(4, 84, 84)$ is drawn with priority from the replay memory. The image decoder takes $\mathbf{S}$ as input and outputs image embeddings $\mathbf{I} \in \mathbb{R}^{64\times7\times7}$, where 64 denotes the size of dimension along the channel axis and the rest denote the height and width dimensions respectively. We L_2 normalise $\mathbf{I}$ along the channel dimension to ensure scale invariance.

In the region-sensitive module, we employ two layers of 1×1 convolutions with ELU activation (Clevert et al., 2016) inspired by (Kazemi & Elqursh, 2017; Bahdanau et al., 2015). The region-sensitive module takes $\mathbf{I}$ as input and outputs score maps $\mathbf{A} \in \mathbb{R}^{N\times7\times7}$. N is the number of score maps each of size 7×7, corresponding to spatial locations

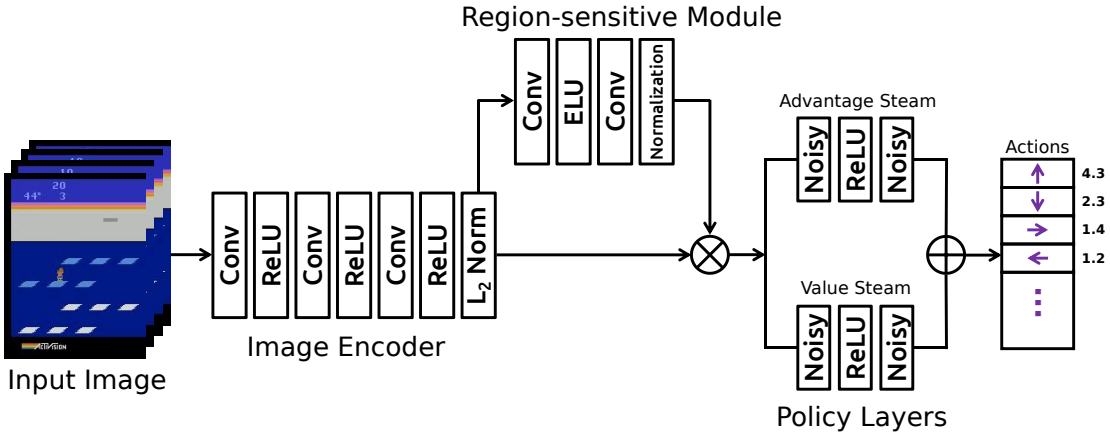


Figure 2. The architecture of the proposed RS-Rainbow.

on the image feature maps. Each score map describes the importance of the image feature vector at the corresponding spatial location. Then score maps $\mathbf{A}$ are passed to a normalisation layer to generate meaningful probability distributions. In our experiments, we implement the normalisation layer using the softmax function or the sigmoid function. The final probability distributions after normalising $\mathbf{A}$ are denoted as $\mathbf{P} = [\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_N]$, where $\mathbf{P}_n \in \mathbb{R}^{1\times 7\times 7}$ is the n-th counterpart of the learned probability distributions.

Each $\mathbf{P}_n$ highlights a unique criterion of the agent in selecting important regions. As discussed in the analysis Sec. 4, each $\mathbf{P}_n$ assigns high importance to a unique category of patterns or type of objects in a game. The most important area given by $\mathbf{P}_n$ summarises a particular aspect in the policy of an agent. Diversely distributed $\mathbf{P}$ describe a holistic strategy of the policy.

For each $\mathbf{P}_n$, we generate the corresponding image embeddings $\mathbf{F}_n$ as a unique representation of the state. $\mathbf{F}_n$ is defined as the element-wise product of $\mathbf{P}_n$ and $\mathbf{I}$ by broadcasting along the channel dimension, as

$$\mathbf{F}_n = \mathbf{P}_n \otimes \mathbf{I}. \tag{1}$$

Hence, $\mathbf{F}_n$ is of the same shape as $\mathbf{I}$. To obtain the final state representation, we aggregate $\mathbf{F}_n$ $(1 \leq n \leq N)$ as

$$\mathbf{F} = \sum_{n=1}^{N} \mathbf{F}_n. \tag{2}$$

In summary, the original image embeddings $\mathbf{I}$ are scaled at each spatial location with the corresponding estimated importance, and N independent estimations are aggregated to form the final representation of the state.

Our work is related to the broader concept of *attention* popularised by (Bahdanau et al., 2015; Vaswani et al., 2017) in the task of neural machine translation and further extended in areas such as visual question answering (Yang et al., 2016; Xu et al., 2015) and image caption generation (Xu et al., 2015). Different from attention, our region-sensitive module does not assume the role as a mapping function from a query and a key-value pair to an aggregated output.

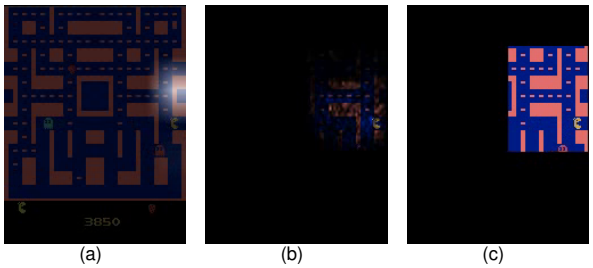


Figure 3. Three alternatives of visualisation strategy. (a) Importance weights overlay. (b) Soft saliency mask. (c) Binary saliency mask.

As in Rainbow, the policy layers split into an advantage stream and a value stream, the outputs of which are aggregated to estimate the state-action value Q. Each stream is implemented with two noisy linear layers and ReLU. The noisy linear layer incorporates extra parameters into a linear function to learn adaptable noise in its output, thereby inducing state-dependent exploration which replaces ϵ-greedy exploration. And finally, Q values are calculated as the mean of a learned distribution over a fixed support set of discrete return values, which are used to derive the policy.

3.3. Visualisation

Based on the region-sensitive module, we explore in this section how to interpret and visualise the learned salient regions, which are most important to decision making.

The first alternative (see Fig. 3(a)) is the direct overlay of upsampled $\mathbf{P}_n$ onto the original screen. The intensity corresponds to the importance weight. As $\mathbf{P}_n$ is of 7×7, this alternative effectively divides the original screen into a 7×7 grid and assumes the receptive field of each element in $\mathbf{P}_n$

corresponds to a grid cell. As we can see the visualisation is neither accurate nor intuitive.

In the second and the third alternatives, we apply *soft* and *binary* saliency masks to the original screen, respectively. We first calculate the gradient-based saliency (Simonyan et al., 2014) of the largest importance score from each score map, $G_n = \frac{\partial \max_l(\mathbf{A}_{nl})}{\partial \mathbf{S}}$ for $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_N]$ where l is spatial location. We take the absolute value of the gradients and normalise between 0 and 1 as saliency. The original saliency corresponds to a soft mask, and we also binarise it to generate a binary mask.

As shown in Fig. 3(b) and Fig. 3(c), we multiply the soft saliency mask and the binary saliency mask with the original frame, respectively. Both approaches can accurately locate the salient object and there is no fundamental difference between them. However, we observe that the soft saliency mask appears fuzzy and rugged, while the binary saliency mask produces clear and intuitive visualisation.

Based on the above analysis, we adopt the binary saliency approach shown in Fig. 3(c) in our following experiments to help interpret challenging games on the Atari platform. Note that our visualisation is automatically learned, which is different from existing a-posteriori methods. Interested readers can refer to (Zahavy et al., 2016; Greydanus et al., 2018) for more details.

4. Atari Gameplay Analysis

4.1. Enduro

In this racing game, the total return for the player is the total number of cars that has been passed. On each day, the maximum available return is the target number of cars the player has to pass to qualify for the next day. Passing more cars than the target does not bring extra return. Variations in weather and time add extra difficulty for avoiding collisions.

We utilise our region-sensitive module as described in Sec. 3.3 to visualise individual *gazes* of the agent. A gaze is a region assigned the highest importance by the region-sensitive module and contributes the most to the Q value prediction. We first define the normal patterns of specialization in each of the two gazes, which appear under the most common game conditions, and use them to explain the general policy. Then we focus on special cases when characteristics of the gazes shift to new patterns, which we discover explaining interesting changes in the policy.

General strategy. We illustrate from Fig. 4(a) to Fig. 4(c) the normal patterns discovered by RS-Rainbow. We first notice that both the left and right gazes consistently attend to the race track. This illustrates that during inference, the agent considers the race track as the most important general area, and makes the most use of features from the race track when predicting Q values (making a decision).

From Fig. 4(a) to Fig. 4(c), the left gaze seems to focus on a distant car near horizon, three mid-distance cars and two close cars, respectively. In the same figures, the right gaze always falls upon the player and a short road segment ahead. We name this gaze the player tracker. The player tracker also leaves room at the top half of its field of vision for detecting upcoming cars.

By far we can summarise the general policy employed by RS-Rainbow in this game. Firstly, the agent ignores irrelevant features from mountains, sky, the mileage board and empty grounds, and relies on information from the race track to make decisions. Specifically, the agent keeps separate two categories of objects on the race track, *i.e.*, cars and the player. On the one hand, the agent locates the player and a local area around it for avoiding immediate collisions with cars. On the other hand, the agent locates the next potential collision targets at different locations, particularly the remote ones. The agent combines these two perspectives in making its decisions.

We highlight three properties of our interpretations. First, the gazes are automatically learned during end-to-end training. Second, our analysis does not treat the model as a black box or operate externally to the policy as (Greydanus et al., 2018) and (Zahavy et al., 2016) do. Instead, what we describe is part of the policy, the learned region-discriminative information used in inference. Third, these explanations are also the reasons for the observed performance improvements in Sec. 5.

Counting down. Near the completion of the current goal, the agent "celebrates" in advance. As shown from Fig. 4(d) to Fig. 4(f), the left gaze loses its focus on cars and diverts to the mileage board starting when only 13 cars remain before completion. The previous "car tracker" now picks up on the "important" information that it is close to victory, and fixates on the countdown. In the short remaining race, there is no noticeable policy shift despite the change in the left gaze, while the player tracker functions normally. Such behavior oddly resembles the premature celebration of a runner in a race. The discovery demonstrates the robustness of adopting multiple independent gazes. We cannot gain such insights only by observing the policy output.

Slacking. Upon reaching the target, the agent does not receive reward signals until the next day starts. During this period the agent learns to output no-op actions, corresponding to not playing the game. We refer to this stage as "slacking." We are interested in what leads to the decision of not playing. Fig. 4(g) shows that both gazes fixate on the mileage board, where flags indicate task completion. When slacking happens, the agent considers the flag signs as important and the road not. The complete reverse in focus as compared

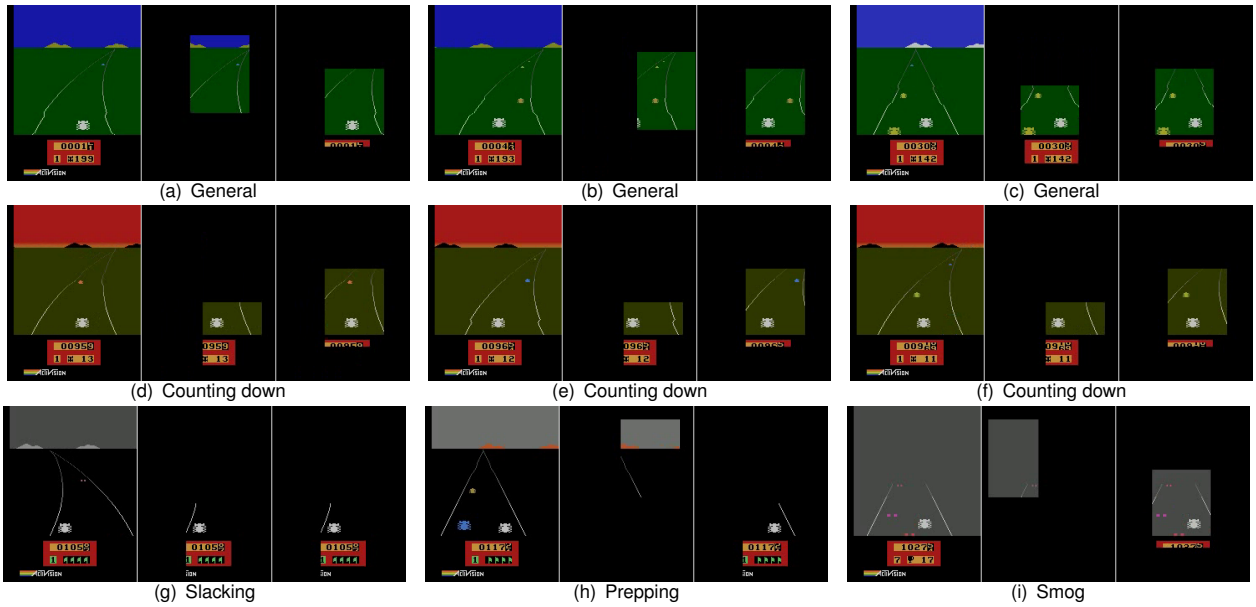


Figure 4. Visualising policies in enduro. (a)-(c) correspond to the general game state. (d)-(f) correspond to the state of counting down. (g), (h) and (i) each represents a special game state.

to the normal case explains this shift in policy. The flags outweigh the road in importance, since they are signs of absolute zero return. By downgrading the importance of the road, the agent is not collecting information for driving therein, thus it outputs no-ops.

Prepping. Shortly before a new day, the agent begins "prepping" for the new race in advance. By being up to speed when the race starts, the player earns more rewards. We are interested in the causes of the change from slacking to prepping as there are still no rewards for driving. As shown in Fig. 4(h), the left gaze focuses on the mountain and sky in the background. As it turns out, the agent recognizes that the time is dawn (right before morning when race starts) from the unique colours of the light gray sky and orange mountains, therefore the agent gets ready early for a head start in the new race.

Smog. When smog partially blocks the forward view, the left gaze loses its focus on cars. It strays off the road into some empty area. The degradation of focus results in minor performance decrease under smog weather.

4.2. Ms_pacman

In this game, ms_pacman accumulates points by collecting pellets while avoiding ghosts. Eating power pellets makes the ghosts vulnerable. Eating fruits and vulnerable ghosts adds bonus points. Ms_pacman proceeds to the next level after eating all pellets.

Fig. 5 illustrates the learned gazes of RS-Rainbow in this game. The right gaze stays focused on ms_pacman to track its position and detect any local threats or benefits. The left gaze detects different objects or locations depending on the current state. The moving objects, which are essential for high return, are ghosts, vulnerable ghosts, and fruits. The left gaze can properly detect these objects in relevant game states and we interpret the visualisations accordingly.

Moving objects detection. In Fig. 5(a), the left gaze detects the two ghosts on the upper-right corner of the map. Therefore, ms_pacman, as located by the right gaze, stays in the mid-left section to safely collect dense rewards. In Fig. 5(b), the left gaze locks in on all three vulnerable ghosts in the mid-right section, as ms_pacman chases after them. In Fig. 5(c), the left gaze detects a newly-appeared cherry at the lower-left warp tunnel entrance. Ms_pacman immediately enters the closest opposite tunnel entrance in the shortest path to the cherry.

Travelling through a warp tunnel. In Fig. 5(d), the right gaze locates ms_pacman entering the upper-right tunnel. In this case, the left gaze no longer detects a moving object, but predicts the upper-left tunnel as the exiting point. In Fig. 5(e) and Fig. 5(f), we observe the same patterns, where the left gaze predicts the destination of the warp tunnel that ms_pacman enters.

Eating the last pellet. As shown in Fig. 5(g), the left gaze locates the last pellet when ms_pacman is in the mid-section of the maze. Therefore ms_pacman moves towards the pellet. In Fig. 5(h), a red ghost appears in the left gaze close to the pellet, causing ms_pacman to deviate to the right. The appearance of the red ghost in the left gaze sheds light on

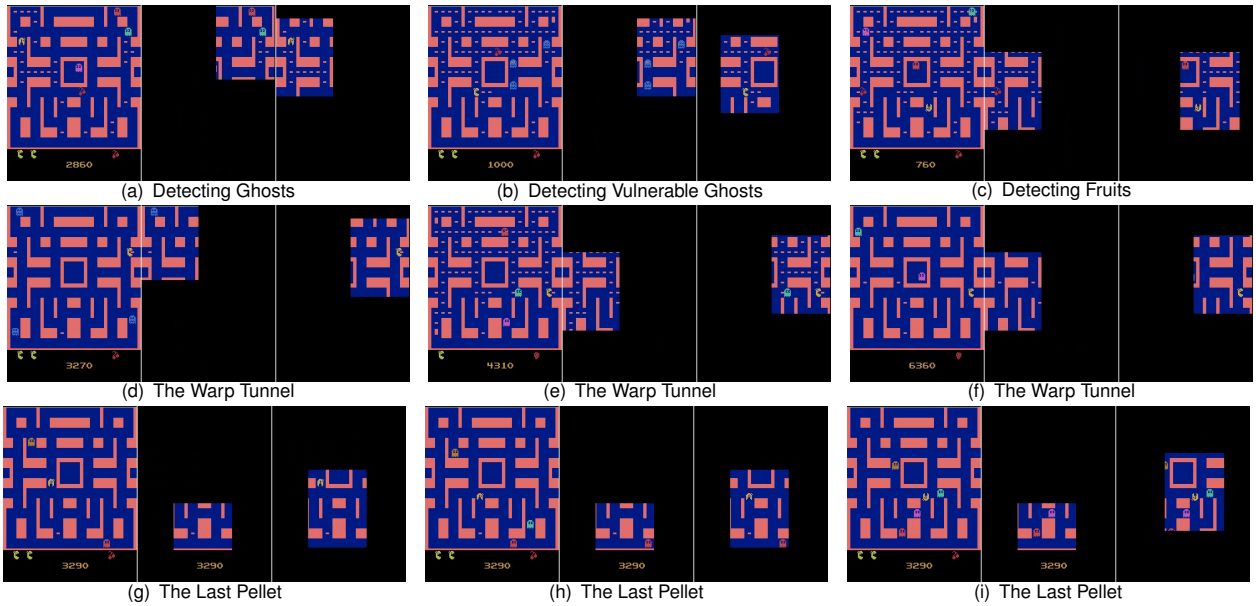


Figure 5. Visualising policies in ms_pacman. (a) Detecting ghosts. (b) Detecting vulnerable ghosts. (c) Detecting fruits. (d)-(f) Travelling through a warp tunnel. (g)-(i) Eating the last pellet in the maze.

the reason of changed direction. After changing course, the ghosts approach ms_pacman from all directions as shown in Fig. 5(i). Even though the agent detects all the ghosts (they appear in the gazes), ms_pacman has no route to escape.

4.3. Frostbite

In Frostbite, the player jumps over ice blocks to build an igloo, upon completion it needs to avoid the bear and enter the igloo. Every collected ice block provides some immediate rewards. Remaining degrees of temperature are also converted into rewards in the end. If the player falls into water or temperature drops to zero, it loses a life. The bear, clams, and birds can all cost the player a life. Fish provides extra bonus.

While the right gaze consistently locates the player, the left gaze takes on the role of a generic target detector. We summarise targets into three general categories, each of which defines an essential sub-task in the game. By visualising each target, we show how the agent specialises in each sub-task.

Jumping. Fig. 6(a) visualises the next jumping target, *i.e.*, uncollected ice blocks (white). The agent must repeatedly jump to the next uncollected ice blocks, not only for immediate rewards, but for building the final igloo. The left gaze often focuses on white ice blocks that are the destinations of jumping.

Entering the igloo. Fig. 6(b) shows the sub-task of the player entering the igloo, after jumping over white ice blocks for building it. The player must avoid the bear when running for the igloo. As it shows, the left gaze tracks the state of the bear and the right gaze locates the player and igloo in this sub-task.

Inspecting progress. Fig. 6(c) shows a frequent pattern in the game, *i.e.*, the left gaze inspecting the building progress of the igloo. As it shows, the igloo is two jumps away from completion, and the left gaze focuses on the igloo in advance for preparing to enter.

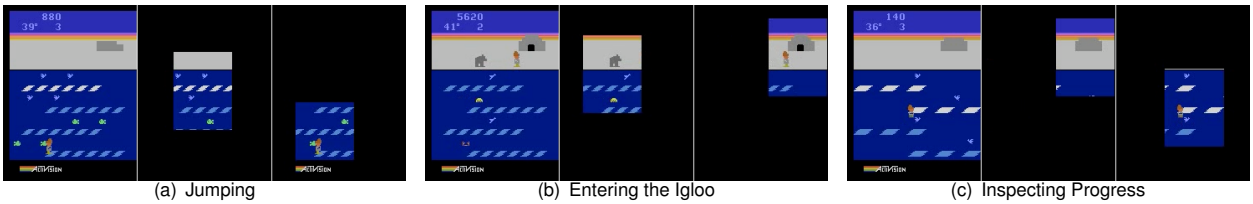
5. Quantitative Evaluation

As emphasized above, RS-rainbow can lead to performance improvements due to a better policy learning paradigm. In this section, we give a quantitative evaluation on benchmark datasets.

5.1. Testing Environment and Prepocessing

Atari 2600 (Bellemare et al., 2013) is a common testbed for DeepRL algorithms, which contains 63 retro games. We employ 6 games to conduct the comparison with other state-of-the-art methods, including beam_rider, pong, ms_pacman, space_invaders, enduro, and seaquest.

As for the preprocessing step, we follow (Mnih et al., 2015; Wang et al., 2016; Schaul et al., 2016; Hessel et al., 2018). In more detail, each frame is converted from RGB format into single-channel grayscale and downsampled from the resolution of 210×160 to 84×84 via bilinear interpolation. At each time step, the input is four consecutive preprocessed frames stacked along the channel dimension.



(a) Jumping (b) Entering the Igloo (c) Inspecting Progress

Figure 6. Visualising policies in frostbite. (a) corresponds to the sub-task of jumping. (b) corresponds to the sub-task of entering the igloo. (c) corresponds to the sub-task of inspecting progress on building igloo.

Table 1. Comparison of performance with other state-of-the-art methods under the no-op testing condition.

Method	beam_rider	pong	ms_pacman	space_invaders	enduro	seaquest
DQN	8,627.5	19.5	3,085.6	1,692.3	729.0	5,860.6
DDQN	13,772.8	20.9	2,711.4	2,525.5	1,211.8	16,452.7
Prior. DDQN	22,430.7	20.7	4,751.2	7,696.9	2,155.0	44,417.4
Duel. DDQN	12,164.0	**21.0**	6,283.5	6,427.3	2,258.2	50,254.2
Dist. DQN	13,213.4	20.8	3,769.2	6,869.1	2,259.3	4,754.4
Noisy DQN	12,534.0	**21.0**	2,501.6	2,145.5	1,129.2	2,495.4
Rainbow	16,850.2	20.9	5,380.4	18,789.0	2,125.9	15,898.9
Rainbow*	17,656.8	20.9	6,686.3	3,001.2	**2,344.9**	73,601.4
RS-Rainbow (ours)	**26,722.3**	20.9	**7,219.3**	**19,670.0**	2,245.7	**245,307.3**

5.2. Implementation Details

We re-implement Rainbow[1] with the same hyperparameters and model details as in (Hessel et al., 2018). For each game, each episode is capped at 108K steps, which is approximately 30-minute gameplay. We clip rewards to the range of $[-1, 1]$. Training is paused every $100K$ environment steps to evaluate the agent for 10 episodes and record the average score. We adopt an action repeat of 4. When testing, the agent executes an ϵ-greedy policy with $\epsilon = 0.001$. During training, the exploration strategy is implemented via noisy linear layers. In the region-sensitive module, we set $N = 2$. For the selection of normalisation layers, we employ the sigmoid function in games space_invaders and seaquest, and the softmax function in the rest games.

We evaluate the trained agents under the no-op random start condition. At the beginning of each test episode, a random number (up to 30) of no-ops are executed (as is done in training) before the agent starts playing. For each game, we report the average scores across 200 test episodes.

5.3. Comparison with State-of-the-art

We compare the performance of RS-Rainbow with several other state-of-the-art methods in Table 1. The selected methods include Rainbow (Hessel et al., 2018), Distributional DQN (Bellemare et al., 2017), Noisy DQN (Fortunato et al., 2018), Duelling DDQN (Wang et al., 2016), Prioritized DDQN (Schaul et al., 2016), DDQN (van Hasselt et al., 2016), and DQN (Mnih et al., 2015).

[1]Code is available at https://github.com/Kaixhin/Rainbow.

For the performance of Rainbow, we report both the original ones quoted from (Hessel et al., 2018) and the ones reproduced by us, which are marked with $*$ on the upper right corner. It should be mentioned that the published performances of those variants of DQN (including Rainbow) are obtained after training for 200 million environment steps, and the reported results of RS-Rainbow and Rainbow* are obtained after training with just 50 million environment steps, due to limited computational resources.

A first glance at Table 1 shows that RS-Rainbow outperforms Rainbow* by a large margin in 4 games, including beam_rider, ms_pacman, space_invaders, and seaquest. For instance, compared with Rainbow*, RS-Rainbow achieves a 51% improvement on beam_rider, 333% improvement on seaquest, and 655% improvement on space_invaders. RS-Rainbow also achieves comparable results in pong and enduro. For the game pong, the maximum score is 21, therefore there is relatively little room for further improvement.

Compared with the other state-of-the-art methods, RS-Rainbow also enjoys solid performance gains. It achieves the best performance among the compared methods on beam_rider, ms_pacman, space_invaders, and seaquest. For example, it beats the second-best methods, prioritized DDQN on beam_rider by 19%, duelling DDQN on ms_pacman by 14%, and Rainbow on space_invaders by 4%. On the rest two games, *i.e.*, pong and enduro, RS-Rainbow also achieves competitive results with a minor decrease of no more than 0.6%. The results are especially encouraging, as RS-Rainbow is trained only with 25% of training frames employed by other state-of-the-art methods.

Considering the relatively large performance improvement

obtained with significantly less training data, it can be expected that more performance gains can be obtained when RS-Rainbow is trained with more data or is deployed on a massively parallel platform (Horgan et al., 2018).

6. Conclusion

We approach the problem of understanding DeepRL models from a learning perspective. Our proposed RS-Rainbow jointly learns to interpret and optimises the current policy. In contrast to previous *a-posteriori* methods, our visualisations are clean, intuitive, and computationally efficient. RS-Rainbow leverages the region-sensitive module, which estimates regional importance for sub-regions on the screen for greater interpretation capacity, and guides the policy learning with learned importance for better performance.

The region-sensitive module employed by RS-Rainbow is easy to implement and end-to-end trainable. Therefore it will be interesting to integrate it with other DeepRL models, such as A3C (Mnih et al., 2016) and proximal policy optimisation (Schulman et al., 2017). We leave these issues in our future work.

References

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Bellemare, Marc G., Dabney, Will, and Munos, Rémi. A distributional perspective on reinforcement learning. In *ICML*, 2017.

Clevert, Djork-Arné, Unterthiner, Thomas, and Hochreiter, Sepp. Fast and accurate deep network learning by exponential linear units (elus). In *ICLR*, 2016.

Dabkowski, Piotr and Gal, Yarin. Real time image saliency for black box classifiers. In *NeurIPS*. 2017.

Dodson, Thomas, Mattei, Nicholas, and Goldsmith, Judy. A natural language argumentation interface for explanation generation in markov decision processes. In *Algorithmic Decision Theory*, 2011.

Elizalde, Francisco, Sucar, Luis, Luque, Manuel, Dez, Francisco, and Reyes Ballesteros, Alberto. Policy explanation in factored markov decision processes. In *European Workshop on Probabilistic Graphical Models*, 2008.

Fong, R. and Vedaldi, A. Interpretable explanations of black boxes by meaningful perturbation. In *ICCV*, 2017.

Fortunato, Meire, Azar, Mohammad Gheshlaghi, Piot, Bilal, Menick, Jacob, Hessel, Matteo, Osband, Ian, Graves, Alex, Mnih, Volodymyr, Munos, Remi, Hassabis, Demis, Pietquin, Olivier, Blundell, Charles, and Legg, Shane. Noisy networks for exploration. In *ICLR*, 2018.

Greydanus, Samuel, Koul, Anurag, Dodge, Jonathan, and Fern, Alan. Visualizing and understanding Atari agents. In *ICML*, 2018.

Hayes, Bradley and Shah, Julie A. Improving robot controller transparency through autonomous policy explanation. In *ACM/IEEE International Conference on Human-robot Interaction*, 2017.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *CVPR*, 2016.

Hessel, Matteo, Modayil, Joseph, Van Hasselt, Hado, Schaul, Tom, Ostrovski, Georg, Dabney, Will, Horgan, Dan, Piot, Bilal, Azar, Mohammad, and Silver, David. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018.

Horgan, Dan, Quan, John, Budden, David, Barth-Maron, Gabriel, Hessel, Matteo, van Hasselt, Hado, and Silver, David. Distributed prioritized experience replay. In *ICLR*, 2018.

Kazemi, Vahid and Elqursh, Ali. Show, ask, attend, and answer: A strong baseline for visual question answering. *CoRR*, abs/1704.03162, 2017.

Khan, Omar Zia, Poupart, Pascal, and Black, James P. Minimal sufficient explanations for factored markov decision processes. In *International Conference on Automated Planning and Scheduling*, 2009.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

Maaten, Laurens van der and Hinton, Geoffrey. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, and Riedmiller, Martin. Playing atari with deep reinforcement learning. In *NeurIPS Deep Learning Workshop*. 2013.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A., Veness, Joel, Bellemare, Marc G., Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K., Ostrovski, Georg, Petersen, Stig, Beattie, Charles, Sadik, Amir, Antonoglou, Ioannis, King, Helen, Kumaran, Dharshan, Wierstra, Daan, Legg, Shane, and Hassabis, Demis.

Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Mnih, Volodymyr, Badia, Adria Puigdomenech, Mirza, Mehdi, Graves, Alex, Lillicrap, Timothy, Harley, Tim, Silver, David, and Kavukcuoglu, Koray. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Schaul, Tom, Quan, John, Antonoglou, Ioannis, and Silver, David. Prioritized experience replay. In *ICLR*, 2016.

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Shrikumar, Avanti, Greenside, Peyton, and Kundaje, Anshul. Learning important features through propagating activation differences. In *ICML*, 2017.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR*, 2014.

Springenberg, Jost Tobias, Dosovitskiy, Alexey, Brox, Thomas, and Riedmiller, Martin. Striving for simplicity: The all convolutional net. In *ICLR*, 2015.

Sutton, Richard S. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

van Hasselt, Hado, Guez, Arthur, and Silver, David. Deep reinforcement learning with double q-learning. In *AAAI*, 2016.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Lukasz, and Polosukhin, Illia. Attention is all you need. In *NeurIPS*, 2017.

Wang, Ziyu, Schaul, Tom, Hessel, Matteo, Hasselt, Hado, Lanctot, Marc, and Freitas, Nando. Dueling network architectures for deep reinforcement learning. In *ICML*, 2016.

Xu, Kelvin, Ba, Jimmy, Kiros, Ryan, Cho, Kyunghyun, Courville, Aaron, Salakhutdinov, Ruslan, Zemel, Richard, and Bengio, Yoshua. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

Yang, Zichao, He, Xiaodong, Gao, Jianfeng, Deng, Li, and Smola, Alexander J. Stacked attention networks for image question answering. In *CVPR*, 2016.

Zahavy, Tom, Ben-Zrihem, Nir, and Mannor, Shie. Graying the black box: Understanding dqns. In *ICML*, 2016.

Zeiler, Matthew D. and Fergus, Rob. Visualizing and understanding convolutional networks. In *ECCV*, 2014.