Visualizing and Understanding Atari Agents

Sam Greydanus **Anurag Koul** **Jonathan Dodge** **Alan Fern**

Oregon State University

{greydasa, koula, dodgej, alan.fern} @ oregonstate.edu

Abstract

Deep reinforcement learning (deep RL) agents have achieved remarkable success in a broad range of game-playing and continuous control tasks. While these agents are effective at maximizing rewards, it is often unclear what strategies they use to do so. In this paper, we take a step toward explaining deep RL agents through a case study in three Atari 2600 environments. In particular, we focus on understanding agents in terms of their visual attentional patterns during decision making. To this end, we introduce a method for generating rich saliency maps and use it to explain 1) what strong agents attend to 2) whether agents are making decisions for the right or wrong reasons, and 3) how agents evolve during the learning phase. We also test our method on non-expert human subjects and find that it improves their ability to reason about these agents. Our techniques are general and, though we focus on Atari, our long-term objective is to produce tools that explain any deep RL policy.

1 Introduction

Deep learning algorithms have achieved state-of-the-art results in image classification [8, 13], machine translation [14], image captioning [9], drug discovery [3], and deep reinforcement learning [15, 21]. Yet while these models can achieve impressive performance on such tasks, they are often perceived as black boxes. In real-world applications, such as self-driving cars and medical diagnosis, explaining the decision processes of these models is a key concern.

While an abundance of literature has addressed techniques for explaining deep image classifiers [6, 18, 22, 27] and deep sequential models [10, 17], very little work has been done to explain deep RL agents. These agents are able to learn strong policies on a wide range of challenging tasks, often using only sparse rewards and noisy, high-dimensional inputs. Simply observing the behavior of these agents is one way to understand their policies. However, explaining their decision-making process in more detail requires better tools.

Visualizing deep RL policies for explanation purposes is difficult. Past methods include t-SNE embeddings [15, 25], Jacobian saliency maps [24, 25], and reward curves [15]. These tools generally sacrifice interpretability for explanatory power or vice versa. Our work is motivated by trying to strike a favorable balance between the two extremes.

In this paper, we introduce a perturbation-based technique for generating high-quality saliency videos of deep reinforcement learning agents. The introduction of this technique was motivated by observing the generally poor quality of Jacobian saliency, which has been primarily used to visualize deep RL agents in prior work (see Figure 1). For the sake of thoroughness, we limit our experiments to three Atari 2600 environments: Pong, SpaceInvaders, and Breakout. Our long-term goal is to visualize and understand the policies of *any* deep reinforcement learning agent that uses visual inputs. To this end, we use an approach that can be adapted to environments beyond Atari.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

After introducing our technique for generating saliency maps, we take an investigative approach to understanding Atari policies. First, we use our visualizations to identify the key strategies of three agents that exceed human baselines in their environments. Second, we visualize agents at various points in training to see how their policies evolve. Next, we use these strong agents to add "hint pixels" into the environment and train "overfit" agents. These overfit agents "cheat" by learning policies that depend on the hint pixels rather than the original environment, but still obtain high rewards. In other words, they make the right decisions, but for the wrong reasons. We use survey results to show that our method helps non-experts differentiate between strong agents and overfit agents, even when these agents earn similar rewards. This provides evidence that our visualizations 1) correctly reflect the attention of the agents, and 2) can help non-experts understand and trust deep RL agents.

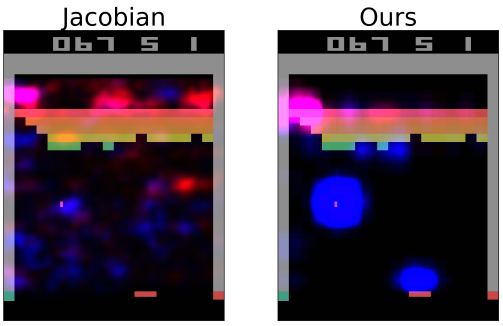


Figure 1: Comparison of Jacobian saliency to our perturbation-based approach. We are visualizing an actor-critic model [16]. Red indicates saliency for the critic; blue is saliency for the actor.

Most of this paper is focused on understanding how an agent's current observation affects its current decision. However, since we use recurrent policy networks, we acknowledge that memory is also important to their behavior. A simple example is an agent which has learned to reason about the velocity of a ball; it needs information about previous frames *in addition to* information from the current frame to do this. In response to these concerns, we present preliminary experiments on the role of memory.

2 Related Work

Explaining traditional RL agents. Prior work has considered generating natural language and logic-based explanations for decisions made by policies in Markov Decision Process (MDP) [4, 5, 11]. These methods all assume access to an exact MDP model (e.g. represented as a dynamic Bayesian network) and that policies are mappings from interpretable, high-level state features to actions. Neither assumption is valid in our vision-based RL domains.

More recently there has been work on explaining RL agents that does not require an MDP model [7]. Rather, policy execution traces are analyzed to extract interpretable policy behavior patterns. This approach, however, relies heavily on the presence of hand-crafted features of states that are semantically meaningful to humans. This is impractical in our vision-based applications, where deep RL agents learn directly from pixels. Explaining agents trained in these environments requires developing a new set of tools.

Explaining deep RL agents. Recent work by Zahavy et al. (2017) [25] has developed tools for explaining deep RL policies in visual domains. Similar to our work, the authors use the Atari 2600 environment as an interpretable testbed. Their key contribution is a method of approximating the observed behavior of deep RL policies via "Semi-Aggregated" Markov Decision Processes (SAMDPs), which are used to gain insights about the deep RL policies. While this process of distilling a simple, explainable policy from a complex, uninterpretable one produces valuable insights, it is unclear how well this truly reflects the internal decision-making of the RL agent. From a user perspective, an issue with the explanations themselves is that they emphasize t-SNE clusters and state-action statistics which are uninformative to those without a background in machine learning.

Ideally, explanations should be obtained directly from the underlying policy and they should be interpretable to an untrained eye. Our contribution compliments the results of Zahavy et al. by addressing a different set of questions with a different set of tools. Whereas they primarily take a black box approach (using SAMDPs to analyze high-level policy behavior), we aim to obtain visualizations that give insight into how inputs influence individual decisions. In order to do so, we turned to previous literature on explaining Deep Neural Networks (DNNs). A wealth of previous works seek to explain DNNs. The objective is often a human-interpretable saliency map. While

techniques vary from work to work, most fall into two main categories: gradient methods and perturbative methods.

Gradient-based saliency methods. Gradient methods produce saliency maps for DNN models by using variants of backpropagation to compute gradients on the input image with respect to an output. The simplest approach is to take the Jacobian with respect to the category of interest [22]. Unfortunately, the Jacobian does not always produce the most (qualitatively) human-interpretable saliency maps. Thus several variants have emerged, aimed at modifying gradients to obtain more meaningful saliency maps. These variants include Guided Backpropagation [23], Excitation Backpropagation [27], and DeepLIFT [20].

Gradient methods are efficient to compute and have clear semantics ($\frac{\partial f(x)}{\partial x_i}$ is a mathematical definition of saliency), but their saliency maps are often difficult to interpret. This is partly because, when answering the question *"What perturbation to the input maximizes a particular output?"*, gradient methods choose perturbations which lack physical interpretability. In other words, changing an input in the direction of the gradient will often move it off the manifold of realistic input images.

Perturbative-based saliency methods. The idea behind perturbative methods is to measure how a model's output changes when some of the input information is removed. For a simple example (borrowed from [6]), consider a classifier which predicts +1 if the image contains a robin and -1 otherwise. Removing information from the part of the image which contains the robin should change the model's output, whereas doing so for other areas should not. However, choosing a perturbation which removes information without introducing any *new* information can be difficult.

The simplest perturbation is to replace part of an input image with a gray square [26] or gray region [18]. One problem with this approach is that replacing pixels with a constant color can introduce unwanted information. Adding a gray square might increase a classifier's confidence that the image contains a gray object, such as an elephant. More recent approaches by [2] and [6] use masked interpolations between the original image I and some other image A, where A is chosen to introduce as little new information as possible.

3 Visualizing Saliency for Atari Agents

In this work, we focus on agents trained via the Asynchronous Advantage Actor-Critic (A3C) algorithm, which is known for its ease of use and strong performance in Atari environments. A3C trains agents that have both a policy (actor) distribution π and a value (critic) estimate V^π. In particular, letting $I_{1:t}$ denote the sequence of image frames from time 1 to time t, $\pi(I_{1:t})$ returns a distribution over actions to take at time t and $V^\pi(I_{1:t})$ estimates the value of the (hidden) world state at time t. We use a deep neural architecture for both π and V^π as detailed in Section 4.

We are interested in understanding these deep RL agents in terms of the information they use to make decisions and the relative importance of visual features. To do this, we found it useful to construct and visualize saliency maps for both π and V^π at each time step. In particular, the saliency map for $\pi(I_{1:t})$ is intended to identify the most important information in frame I_t used by the policy to select an action at time t. Similarly, the saliency map for $V^\pi(I_{1:t})$ is intended to identify the most important information in frame I_t for assigning a value to the world state at time t.

In our initial work, we found that gradient-based saliency methods produced results that were difficult to interpret when visualized for entire games. This led us to develop a perturbative method, which we found to produce very rich and insightful saliency videos[1].

Perturbation-Based Saliency. Given an image I_t at time t, we let $\Phi(I_t, i, j)$ denote our perturbation of I_t centered at image coordinate (i, j). $\Phi(I_t, i, j)$ is given by Equation 1 and represents a spatially-weighted blur centered around (i, j). In this definition $A(I_t, \sigma_A)$ is a Gaussian blur of the original input (with variance $\sigma_A = 3$) and $M(i, j) \in (0, 1)^{m \times n}$ is an image mask of dimensions $m \times n$ corresponding to a two-dimensional Gaussian with center at $\mu = (i, j)$ and $\sigma^2 = 25$.

$$\Phi(I_t, i, j) = I_t \odot (1 - M(i, j)) + A(I_t, \sigma_A) \odot M(i, j) \tag{1}$$

[1] We found that making videos from these saliency maps produced very rich visualizations. Videos and code (PyTorch) available at `github.com/greydanus/visualize_atari`

This perturbation can be interpreted as adding spatial uncertainty to a region around (i, j). For example, if location (i, j) coincides with the location of the ball in a frame from the Pong environment, our perturbation will diffuse the ball's pixel values, making the policy less certain about the ball's absolute position.

We are interested in answering the question, *"How much does removing information from the region around location (i, j) change the policy distribution or value estimate?"* Focusing first on the policy π, let $\pi_u(I_{1:t})$ denote the vector of unnormalized values that are computed as inputs to the final softmax layer[2] of π. With these quantities, we define our saliency metric for image location (i, j) at time t as follows:

$$\mathcal{S}_\pi(t, i, j) = \frac{1}{2}\|\pi_u(I_{1:t}) - \pi_u(I'_{1:t})\|^2 \quad \text{where} \quad I'_k = \begin{cases} \Phi(I_k, i, j) & \text{if } k = t \\ I_k & \text{otherwise} \end{cases} \tag{2}$$

Qualitatively, the difference $\pi_u(I_{1:t}) - \pi_u(I'_{1:t})$ can be thought of as a finite differences approximation of the directional gradient $\nabla_{\hat{v}}\pi_u(I_{1:t})$ where the directional unit vector $\hat{v}$ denotes the gradient in the direction of $I'_{1:t}$. Our saliency metric is proportional to the squared magnitude of this quantity. It is this intution that suggests how our perturbation method may improve on gradient-based methods. In particular, the unconstrained gradient need not point in a directly that is visually meaningful to a human. By constraining the direction of change to more meaningful and visually coherent choices, we obtained saliency maps that tended to be much more interpretable.

Saliency in practice. With these definitions, we can construct a saliency map for policy π at time t by computing $\mathcal{S}(t, i, j)$ for every pixel in I_t. In practice, we found that computing a saliency score for $i \mod k$ and $j \mod k$ (we used $k = 5$) produced acceptable saliency maps at lower computational cost. For visualization purposes, we upsampled these maps to the full resolution of the Atari input frames and added them to one of the three (RGB) color channels.

An identical approach is used to construct saliency maps for the value estimate V^π. The only difference is that saliency is defined in terms of the squared difference between the value estimate of the original sequence and the perturbed sequence. That is,

$$\mathcal{S}_V(t, i, j) = \frac{1}{2}\|V^\pi(I_{1:t}) - V^\pi(I'_{1:t})\|^2. \tag{3}$$

We chose to display policy saliencies with blue pixels and value saliencies with red.

4 Experiments

Below we first describe implementation details for our method. Next we present a series of experimental results that use our saliency technique for multiple explanatory purposes.

4.1 Implementation Details

All Atari agents in this paper have the same recurrent architecture. The input at each time step is a preprocessed version of the current frame. The input is processed by 4 convolutional layers (each with 32 filters, kernel sizes of 3, strides of 2, and paddings of 1), followed by an LSTM layer with 256 hidden units, and a fully-connected layer with $n + 1$ units, where n is the dimension of the Atari action space. We applied a softmax activation to the first n neurons of the fully-connected layer to obtain $\pi(I_{1:t})$ and used the last neuron to predict the expected reward, $V^\pi(I_{1:t})$ for images $I_{1:t}$.

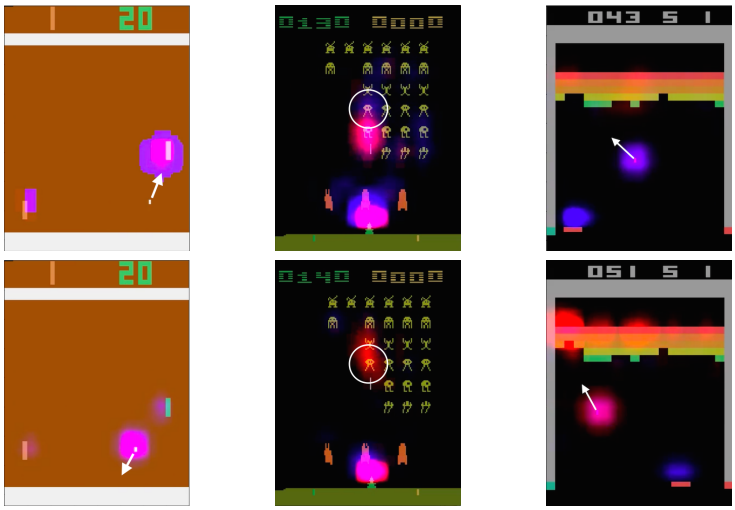
We trained agents on Pong, Breakout, and SpaceInvaders using the OpenAI Gym API [1]. We chose these environments because each poses a different set of challenges and deep RL algorithms have historically exceeded human-level performance in them [15]. Preprocessing consisted of gray-scaling, down-sampling by a factor of 2, cropping the game space to an 80×80 square and normalizing the values to $(0, 1)$. We used the A3C RL algorithm with a learning rate of $\alpha = 10^{-4}$, a discount factor of $\gamma = 0.99$, and computed loss on the policy using Generalized Advantage Estimation with $\lambda = 1.0$ [19]. Each policy was trained asynchronously for a total of 40 million frames with 20 CPU processes and a shared version of the Adam optimizer [3] [12].

[2] We found that working with π_u rather than the softmax output π resulted in sharper saliency maps.

[3] Code (PyTorch) available at `github.com/greydanus/visualize_atari`

4.2 Understanding Strong Policies

Our first objective was to use saliency videos to explain strategies learned by strong Atari agents. These agents all exceeded human baselines in their environments by a significant margin. First, we generated saliency videos for three episodes (2000 frames each). Next, we conducted a qualitative investigation of these videos and noted strategies or features that stood out.



(a) Pong: kill shot (b) SpaceInvaders: aiming (c) Breakout: tunneling

Figure 2: Visualizing strong Atari 2600 policies. We use an actor-critic network; the actor's saliency map is blue and the critic's saliency map is red. White arrows denote motion of the ball.

The strong Pong policy. Our deep RL Pong agent learned to beat the hard-coded AI over 95% of the time, often by using a "kill shot" which the hard-coded AI was unable to return. Our initial understanding of the kill shot was that the RL agent had learned to first "lure" the hard-coded AI into the lower region of the frame and then track and aim the ball towards the top of the frame, where it was most difficult for the hard-coded AI to return.

Saliency visualizations told a different story. It appears that the deep RL agent is exploiting the deterministic nature of the Pong environment; it has learned that, upon executing a precise series of actions, it can obtain a reward with very high certainty. In the top frame of Figure 2a, the agent is positioning its own paddle in order to return the ball at a specific angle. Note that the agent attends to very little besides its own paddle, probably because the movements of the ball and opponent are fully deterministic. Similarly, once the agent has executed the kill shot (lower frame), saliency centers entirely around the ball because there is nothing that either paddle can do to alter the outcome.

The strong Space Invaders policy. When we observed our SpaceInvaders agent without saliency maps, we noted that it had learned a strategy that resembled aiming. However, we were not certain of whether it was "spraying" shots towards dense clusters of enemies, or whether it was picking out individual targets.

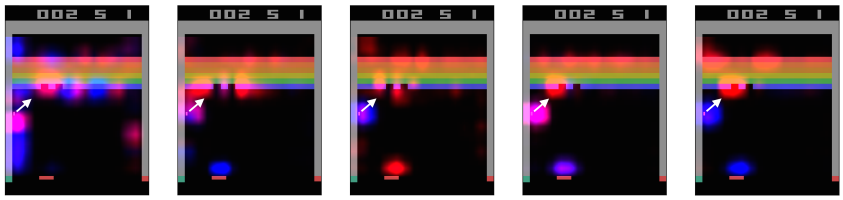
Applying saliency videos to this agent revealed that it had learned a sophisticated aiming strategy, during which first the actor and then the critic would "track" a target. Aiming begins when the actor highlights a particular alien in blue (circled enemy in Figure 2b). This is somewhat difficult to see because the critic network is also attending to a recently-vanquished opponent below. Aiming ends with the agent shooting at the new target. At this point the critic highlights the target in anticipation of an upcoming reward (lower frame). Notice that both the actor and the critic tend to monitor the area above the ship at the bottom of the screen. This may be useful for determining whether the ship is protected from enemy fire and/or has a clear shot at enemies.

The strong Breakout policy. Previous works have noted that strong Breakout agents often develop tunneling strategies [15, 25]. During tunneling, an agent repeatedly directs the ball at a region of the brick wall in order to tunnel through it. The strategy is desirable because it allows the agent to obtain dense rewards by bouncing the ball between the ceiling and the top of the brick wall.

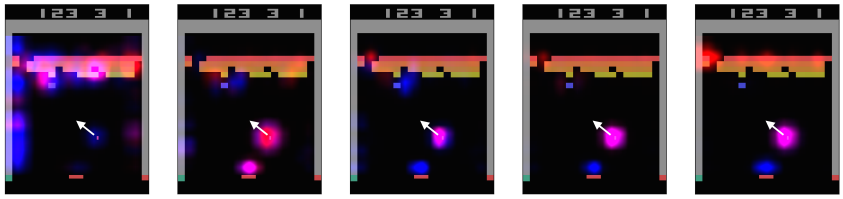
Our impression was that possible tunneling locations would become (and remain) salient from early in the game. Instead, we found that the agent enters and exits a "tunneling mode" over the course of a single frame. Once the tunneling location became salient, it remained so until the tunnel was finished. In the top frame of Figure 2c, the agent has not yet initiated a tunneling strategy and the value network is relatively inactive. Just 20 frames later, the value network starts attending to the far left region of the brick wall, and continues to do so for the next 70 frames (lower frame).

4.3 Policies during Learning

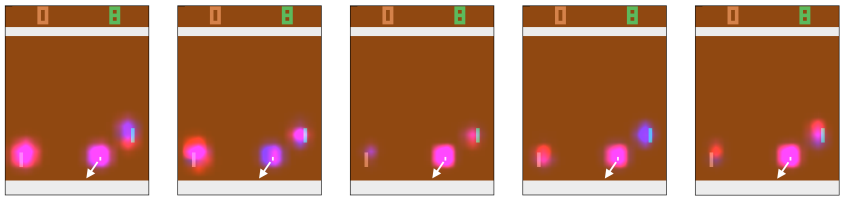
During learning, deep RL agents are known to transition through broad spectrum of strategies. Some of these strategies are eventually discarded in favor of better ones. While training AlphaGo Zero [21], for example, researchers observed first an increase in the frequency of moves made by professional players and then a decrease, as the model discovered strategies unknown to humans. Does an analogous process occur in Atari agents? We explored this question by saving several models during training and visualizing them with our saliency method.



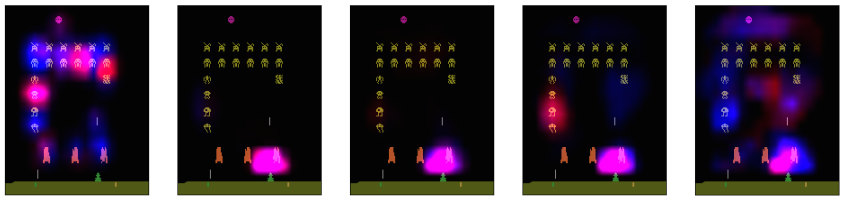
(a) Breakout: learning what features are important.



(b) Breakout: learning a tunneling strategy.



(c) Pong: learning a kill shot.



(d) SpaceInvaders: learning what features are important and how to aim.

Figure 3: Visualizing learning. Each row corresponds to a selected frame from a game played by a fully-trained agent. Leftmost agents are untrained, rightmost agents are fully-trained. Each column is separated by ten million frames of training. White arrows denote motion of the ball.

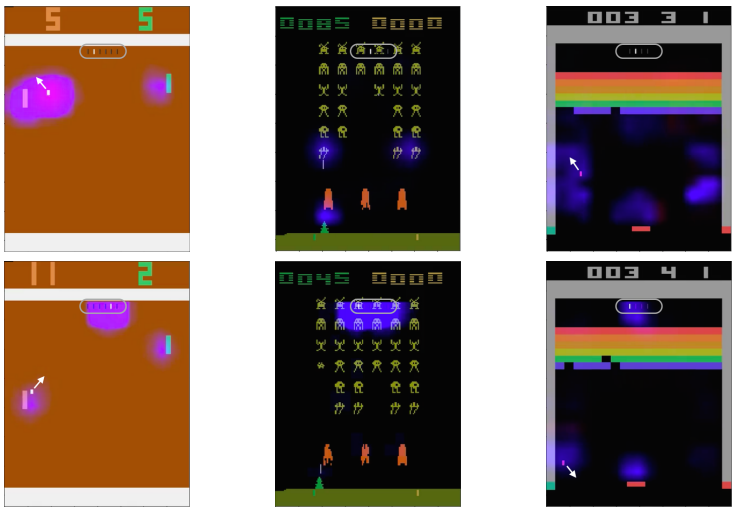
Learning policies. Figure 3 shows how attention changes during the learning process. We see that Atari agents exhibit a significant change in their attention as training progresses. In general, the "most salient" regions/objects to the actor and critic networks vary dramatically during training which suggests that they make decisions for very different reasons. For example, in Breakout, Figure 3b shows how the critic appears to learn about the value of tunneling the ball through the bricks as

depicted by the clear focus on the tunnel in the upper left part of the screen. As another example, in Space Invaders, we noticed that early in training the agent began by "spraying bullets," during which the actor-critic saliencies focused on the spaceship at the bottom of the frame and occasionally a cluster of opponents above. As training progressed, though, the agent shifted to an aiming-based policy.

4.4 Detecting Overfit Policies

Here our objective was to understand the difference between a strong policy and a overfit policy. By "overfit", we refer to a policy that obtains high rewards, but "for the wrong reasons". A secondary objective of this experiment is to provide a more controlled setting for validating whether our saliency maps truly reflect the attention of our agents. We constructed a toy example where we encouraged overfitting by adding "hint pixels" to the raw Atari frames. For "hints" we chose the most probable action selected by a strong ("expert") agent and coded this information as a one-hot distribution of pixel intensities at the top of each frame (see Figure 4 for examples).

With these modifications, we trained overfit agents to predict the expert's policy in a supervised manner. We trained "control" agents in the same manner, assigning random values to their hint pixels. We expected that the overfit agents would learn to focus on the hint pixels, whereas the control agents would need to attend to relevant features of the gamespace. We halted training after 3×10^6 frames, at which point all agents obtained mean episode rewards at or above human baselines. We were unable to distinguish the overfit agents from the control agents by observing their behavior.



(a) Pong (b) SpaceInvaders (c) Breakout

Figure 4: Visualizing overfit Atari policies. The top row shows control agents whereas the bottom row shows overfit agents. Grey boxes (near the top of each frame) denote the hint pixels. White arrows denote motion of the ball.

In all three games, our saliency technique made clear the difference between overfit and control agents. This finding helps validate our saliency method, in that it can pinpoint regions of the input that we already know are most important to the agent. Second, it serves as a good, although contrived, example of how saliency maps can identify agents that obtain high rewards for the wrong reasons.

4.5 Visualizations for Non-experts

Convincing human users to trust deep RL is a notable hurdle in most real-world applications. Non-experts should be able to understand what a strong agent looks like, what an overfit agent looks like, and reason about *why* these agents behave the way they do. We surveyed 31 students at Anonymous Institution to measure how our visualization helps non-experts with these tasks.

Our survey consisted of two parts. First, participants watched videos of two agents (one control and one overfit) playing Breakout *without* saliency maps. The policies appear nearly identical in these

clips. Next, participants watched the same videos *with* saliency maps. After each pair of videos, they were instructed to answer several multiple-choice questions.

Table 1: *"Which agent has a more robust strategy?"*

	Can't tell	Agent 1 (overfit)	Agent 2 (control)
Video without saliency	16.1	**48.4**	35.5
Video with saliency	16.1	25.8	**58.1**

Results in Table 1 indicate that saliency maps helped participants judge whether or not the agent was using a robust strategy. In free response, participants generally indicated that they had switched their choice of "most robust agent" to Agent 2 (control agent) after seeing that Agent 1 (the overfit agent) attended primarily to "the green dots."

Another question we asked was *"What piece of visual information do you think Agent X primarily uses to make its decisions?"*. Without the saliency videos, respondents mainly identified the ball (Agent 1: 67.7%, Agent 2: 41.9%). With saliencies, most respondents said Agent 1 was attending to the hint pixels (67.7%) and Agent 2 was attending to the ball (32.3%).

4.6 Importance of Memory

Memory is a key part of recurrent policies that we have not yet addressed. To motivate future directions of research, we modified our perturbation method to measure the saliency of memory over time. Since memory vectors are not spatially correlated, we chose a different perturbation: decreasing the magnitudes of all entries by 1%. This perturbation reduces the relative magnitude of the LSTM memory vector compared to the CNN vector that encodes the input frame; if memory is not important to a decision, it should not have a large impact on the action probabilities.

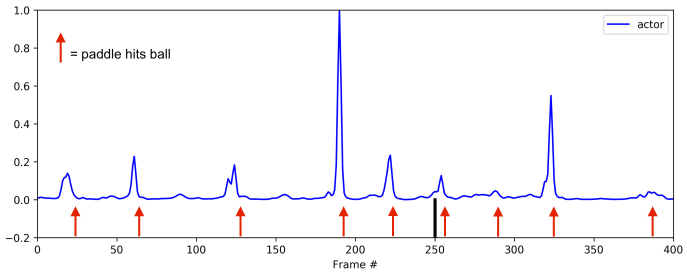


Figure 5: Our saliency metric, applied to the memory vector of the Breakout agent, indicates that memory is most salient immediately before the ball contacts the paddle.

Our preliminary results suggest that memory is most salient to Pong and Breakout agents immediately before the ball contacts the paddle (see Figure 5). The role of memory in SpaceInvaders was less clear. These results are interesting, but we recognize that the policy might be most sensitive to *any* perturbations immediately before the paddle contacts the ball. If this is the case, understanding the contributions of memory to these agents' policies will require a new set of visualization tools.

5 Conclusions

In this paper, we addressed the growing need for human-interpretable explanations of deep RL agents by introducing a new saliency method and using it to visualize and understand Atari agents. We found that our saliency method can yield effective visualizations for a variety of Atari agents. We also found that these visualizations can help non-experts understand what deep RL agents are doing. Finally, we obtained preliminary results for the role of memory in these policies.

Understanding deep RL agents is difficult because they are black boxes that can learn nuanced and unexpected strategies. To produce explanations that satisfy human users, researchers will need to use not one, but many techniques for extracting the "how" and "why" from these agents. This work compliments previous efforts, taking the field a step closer to producing truly satisfying explanations.

Acknowledgments

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract N66001-17-2-4030. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the DARPA, the Army Research Office, or the United States government.

References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv Preprint*, 2016.

[2] Piotr Dabkowski and Yarin Gal. Real Time Image Saliency for Black Box Classifiers. *Neural Information Processing Systems*, 2017.

[3] George E Dahl, Navdeep Jaitly, and Ruslan Salakhutdinov. Multi-task Neural Networks for QSAR Predictions. *arXiv preprint*, 2014.

[4] Thomas Dodson, Nicholas Mattei, and Judy Goldsmith. A Natural Language Argumentation Interface for Explanation Generation in Markov Decision Processes. *Algorithmic Decision Theory*, pages 42–55, 2011.

[5] Francisco Elizalde, L Enrique Sucar, Manuel Luque, Francisco Javier Díez, and Alberto Reyes. Policy Explanation in Factored Markov Decision Processes. *Proceedings of the 4th European Workshop on Probabilistic Graphical Models (PGM 2008)*, pages 97–104, 2008.

[6] Ruth C Fong and Andrea Vedaldi. Interpretable Explanations of Black Boxes by Meaningful Perturbation. *International Conference on Computer Vision*, 2017.

[7] Bradley Hayes and Julie A Shah. Improving Robot Controller Transparency Through Autonomous Policy Explanation. *ACM/IEEE International Conference on Human-Robot Interaction*, pages 303–312, 2017.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *Computer Vision and Pattern Recognition*, 2015.

[9] Andrej Karpathy and Li Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions. *CVPR*, 2015.

[10] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and Understanding Recurrent Networks. *International Conference on Learning Representations*, 2016.

[11] Omar Zia Khan, Pascal Poupart, and James P Black. Minimal Sufficient Explanations for Factored Markov Decision Processes. *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 194–200, 2009.

[12] Diederik P Kingma and Jimmy Lei Ba. Adam: A Method for Stochastic Optimization. *ArXiv Preprint (1412.6980)*, 2014.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*, pages 1097–1105, 2012.

[14] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, and Sanjeev Khudanpur. Recurrent neural network based language model. *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association*, pages 1045–1048, 2010.

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518, 2015.

[16] Volodymyr Mnih, Adrià Puigdomènech, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *International Conference on Machine Learning*, pages 1928–1937, 2016.

[17] W James Murdoch and Arthur Szlam. Automatic Rule Extraction from Long Short Term Memory Networks. *International Conference on Machine Learning*, 2017.

[18] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[19] John Schulman, Philipp Moritz, Sergey Levine, Michael I Jordan, and Pieter Abbeel. High Dimensional Continuous Control Using Generalized Advantage Estimation. *International Conference on Learning Representations*, 2016.

[20] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning Important Features Through Propagating Activation Differences. *Proceedings of Machine Learning Research*, 70:3145–3153, 2017.

[21] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature Publishing Group*, 550, 2017.

[22] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv Preprint*, 2014.

[23] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. *International Conference on Learning Representations*, 2015.

[24] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, and Marc Lanctot. Dueling Network Architectures for Deep Reinforcement Learning. *International Conference on Machine Learning*, 48:1995–2003, 2016.

[25] Tom Zahavy, Nir Baram, and Shie Mannor. Graying the black box: Understanding DQNs. *International Conference on Machine Learning*, pages 1899–1908, 2016.

[26] Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. *European conference on computer vision*, pages 818–833, 2014.

[27] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down Neural Attention by Excitation Backprop. *Zhang, Jianming, et al. "Top-down neural attention by excitation backprop." European Conference on Computer Vision*, pages 543–559, 2016.