Transparency in Deep Reinforcement Learning Networks

Ramitha Sundar

CMU-RI-TR-18-48

Aug 2018

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee:
Dr. Katia Sycara, Chair
Dr. Jean Hyaejin Oh
Wenhao Luo

For my parents

Abstract

In the recent years there has been a growing interest in the field of explainability for machine learning models in general and deep learning in particular. This is because, deep learning based approaches have made tremendous progress in the field of computer vision, reinforcement learning, language related domains and are being increasingly used in application areas such as medicine and finance. But before we fully adopt these models, it is important for us to understand the motivations behind network decisions. This helps us to gain trust in the network, to verify that network decisions are fair to those affected by it and to debug the network model. Moreover, it helps us to gain insights about underlying mechanisms learned by the network and understand the limitations of the network i.e. the domain in which the network performs well and conditions when the network fails.

In this particular work, we explore transparency in deep reinforcement learning networks. We focus on answering the question- why a particular decision was taken by a value based deep reinforcement learning agent and identify attributes in the input space that positively or negatively influence its future actions in a human interpretable manner. Particularly, we discuss an approach "object saliency" at length and demonstrate that it can be used as a simple and effective computational tool for this purpose. We compare and contrast it with existing saliency approaches using a quantitative measure, discuss results from a pilot human experiment to study intuitiveness of object saliency and show how object saliency can provide insights into differences in value function learned by different RL architectures or training approaches, that is not highlighted by existing methods. We also show that it is possible to develop rule based textual descriptions of object saliency maps for easy interpretability by humans - which is difficult to do with existing approaches.

Acknowledgements

I am extremely grateful to my advisor, Dr. Katia Sycara, who introduced me to the field of multi-agent systems and Explainable AI. Her invaluable support, vast experience and guidance lead to the development of this thesis. I am thankful to Prof. Michael Lewis (University of Pittsburgh) and my committee members Dr. Jean Oh and Wenhao Luo for their feedback and insightful comments on my work.

I would also like to thank Huao Li, Raghuram Mandyam Annasamy, Xinzhi Wang, Swaminathan Gurumurthy and Akshat Agarwal for their intellectual contributions and help in charting the course of this project.

To my labmates, Sasanka Nagavalli, Navyata Sanghvi, Anqi Li, Changjoo Nam, Yuezhang Li, Rahul Iyer, Meghan Chandarana, Vigneshram Krishnamoorthy, Yi Sha, Sumit Kumar, Yifan Ding, Rui Liu and Dana Hughes, thank you for all the encouragement, fun and amazing memories.

Last but not the least, I must express heartfelt gratitude to my family for their unconditional love and for being my source of strength at every stage.

Contents

1 Introduction **1**
1.1 Contributions . . . 1

2 Saliency Analysis for Reinforcement Learning **2**
2.1 Reinforcement Learning - Preliminaries . . . 2
2.2 Existing Saliency Methods . . . 2
2.2.1 Background . . . 2
2.2.2 Saliency maps for Reinforcement Learning . . . 3
2.2.3 Drawbacks of Current Approaches . . . 5
2.3 Importance of Objects in Neural Network's Decisions . . . 5
2.4 Object Saliency Computation . . . 7
2.4.1 Object Segmentation and Masking . . . 7
2.4.2 Object Saliency . . . 8
2.5 Comparison of Saliency Methods . . . 10
2.5.1 Background - AOPC Metric . . . 10
2.5.2 Discussion . . . 11
2.6 Studying Intuitiveness of Object Saliency through Human Experiments 13
2.6.1 Results : . . . 14
2.6.2 Discussion : . . . 15
2.7 Building rule based descriptions for object saliency maps . . . 17
2.7.1 Related Work . . . 17
2.7.2 Describing the rules . . . 17
2.7.3 Descriptions generated from Saliency Maps . . . 19
2.8 Merits of Object Saliency - Detecting Adversarial Noise . . . 19
2.9 Merits of Object Saliency - Identifying Differences in Value Functions 21
2.9.1 Rainbow Architecture Overview . . . 21
2.9.2 Saliency Maps on Rainbow . . . 22
2.9.3 Value Prediction Network Overview . . . 23
2.9.4 Saliency Maps on VPN agent . . . 23
2.9.5 Discussion . . . 24

3 Conclusion and Future Work **25**
3.1 Conclusion . . . 25
3.2 Future Work . . . 25

1 Introduction

In recent years, extensive progress has been made in the field of Deep Reinforcement Learning (DRL) including the ability to achieve human level scores in video games from raw pixel data such as Atari [23] and Doom [18], gain super human proficiency in challenging games such as Go [31], learn control for robots [12],[17] and complex motion such as walking or running for neuromusculoskeletal environments [19]. Policies for such tasks are learned from scratch through a large number of interactions with the environment in an attempt to maximize the expected reward for a DRL agent. Despite these successes, reinforcement learning network policies are opaque i.e. they are not explainable. With the current approaches, no evidence of high level strategies or structured reasoning about the environment is immediately apparent. Humans must be able to verify that network decisions are based on correct input attributes and this means DRL agents must be able to explain their decisions and causes for chosen actions in human intelligible terms. Apart from improving user trust, transparency in DRL networks can also help to debug trained models and provide insights into reasons why a network performs well for some states as well as identify states where a network would potentially not perform well.
True policy explainability would ideally require an ability to learn causal models about the world, an ability to use these models to understand what is present in the environment and imagine what could happen in order to plan actions [21]. Learning such models however is very difficult and typically model free networks or a combination of model free and model based approaches is used in deep reinforcement learning applications. In this thesis, we focus on addressing the problem of transparency in deep reinforcement learning, primarily for model free value based RL networks trained on Atari 2600 games. We focus on developing post-hoc, model agnostic and local explanation of causes of network decisions in these Atari game states. We mainly attempt to answer the question - "Why did the network choose this action". Because reinforcement learning is a sequential decision making process, we also want to identify what factors in the input space affect the future network decision the most.

1.1 Contributions

We will discuss an approach called "Object Saliency" at length and present its advantages compared to existing methods. We will show that objects in an input frame are most important for affecting a neural network's decision, that salient regions identified by object saliency are human interpretable and that it is possible to construct rule based textual descriptions from saliency maps for easy interpretation by untrained users. We will compare object saliency with existing approaches using a quantitative metric to show that it identifes regions of high relevance and is robust. We will also highlight some merits of object saliency including the ability to visually identify the presence of adversarial noise and the ability to identify differences in value functions learned by agents trained with different DRL architectures.

The work presented in this thesis builds on the concept of object saliency which was first introduced in the following publication :

[1]. Li, Yuezhang, Katia Sycara, and Rahul Iyer. "Object-sensitive deep reinforcement learning." In 3rd Global Conference on Artificial Intelligence.(2017).

Pilot human experiments to study intuitiveness of object saliency maps appeared in the following publication :

[2]. Iyer, Rahul, Yuezhang Li, Huao Li, Michael Lewis, Ramitha Sundar, and Katia Sycara. "Transparency and Explanation in Deep Reinforcement Learning Neural Networks." (2018).

2 Saliency Analysis for Reinforcement Learning

2.1 Reinforcement Learning - Preliminaries

We consider the standard RL setting where an agent learns to solve a sequential decision problem that is modelled as a Markov Decision Process $\Big(\mathcal{S}, \mathcal{A}, \mathcal{T}(s, s'), r_a(s, s'), \gamma\Big)$ [35]. Here, $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{T}$ is the unknown state transition probability function, $r_a(s, s')$ is the immediate reward associated with taking action $a \in \mathcal{A}$ while transitioning from state $s \in \mathcal{S}$ to $s' \in \mathcal{S}$ and $\gamma \in [0, 1]$ is the discount factor that represents a tradeoff between maximizing immediate returns versus future returns. The goal of the agent is to identify a policy π to maximize its expected reward where cumulative return at each time step is $R_t = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau$.

For a stochastic policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, the value of a state is given by $V^\pi(s) = \mathbb{E}(\sum_{t \geq 0} \gamma^t r_t)$ and the Q value of a state-action pair is given by $Q^\pi(s_t, a_t) = \mathbb{E}[R_t | s = s_t, a = a_t, \pi]$. Fundamental to all value based RL approaches is a recursive Bellman update equation shown in Eq.[1], that states that the maximum future reward at any state s_t and for current action a_t is the sum of immediate reward r_t and maximum future reward in the next state s_{t+1} discounted by γ :

$$Q^\pi(s_t, a_t) = E_{s_{t+1}}[r_t + \gamma E_{a_{t+1} \sim \pi(s_{t+1})}[Q^\pi(s_{t+1}, a_{t+1})]] \tag{1}$$

2.2 Existing Saliency Methods

2.2.1 Background

Approaches for understanding deep learning networks in the supervised learning domain mainly focus on image saliency. They aim to highlight parts of the input image that are most relevant for deciding the neural network output. One of the first such methods was introduced in [32] where the authors suggest computing pixel saliency based on backpropagating gradients to the image layer as a measure of sensitivity for a model trained on ILSVRC-2013 dataset. We refer to this approach as Pixel Saliency throughout this document. The authors also developed an optimization procedure to numerically generate an image that is best representative of each class in terms of the ConvNet scoring model. Other notable sensitivity approaches include a layer wise pixel decomposition method [1], visual backpropagation [4], guided backpropagation

[34], SmoothGrad [33], Gradient-weighted Class Activation Mapping [30] and a deconvolution based approach as discussed in [41]. A related paper [28] develops a quantitative measure to assess saliency heatmap quality for applications where humans may not be available to intuitively assess these based on a prior of what is regarded relevant. Deriving from these sensitivity based methods, [7] formulates explanations as meta-predictors and computes the best explanation (a "mask") through empirical risk minimization. The authors use a constant, noise or blur based perturbation to the image region under these masks and play a maximally informative deletion game i.e. attempt to identify which perturbed region when turned off causes the biggest difference in network predictions. We refer to this approach as mask based optimization in the rest of the document. In a related work that studies saliency in reinforcement learning, Greydanus et al. [11] use standard jacobian pixel saliency in images perturbed by a gaussian blur to understand and visualize agent policy in Atari games.

2.2.2 Saliency maps for Reinforcement Learning

In traditional supervised learning frameworks and for the specific task of classification for which most of the saliency methods discussed in Sec.[2.2.1] were developed, it is easier to qualitatively judge the heatmaps. Figure [1] shows the output of Vanilla Pixel Saliency [32], Guided Backpropogation [34], GradCAM [30], and Mask based Optimization approach [7] as applied to an image from the ImageNet dataset on a pretrained VGG-16 model. In reinforcement learning tasks and for our specific domain

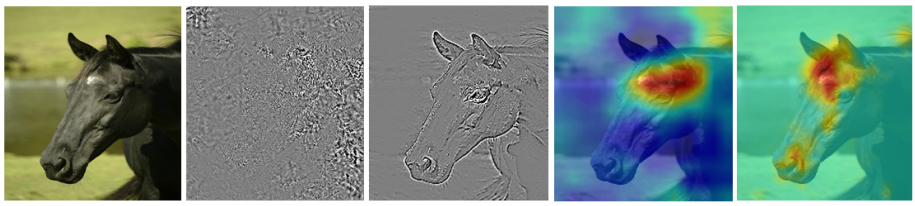


Figure 1: Image of a horse from ImageNet and saliency masks on trained VGG-16 network in order (a) Pixel Saliency (b) Guided Backprop (c) GradCAM (d) Mask based Optimization

of Atari games, there are several objects of interest and different means of network analysis generate vastly different results. More importantly, they also require substantial efforts for human interpretation. Fig.[2] shows the masks obtained from different saliency approaches for one input frame from Atari 2600 benchmark game MsPacman on a pretrained DQN model. These approaches lead to vastly different interpretations for the reasons that cause a network to take a particular action. A short summary of the approaches used to compute other saliency masks is as follows :

- Pixel Saliency : The non-linear network class specific decision score $S_c(I)$ is approximated by a first order Taylor Expansion, $S_c(I) = w^T(I) + b$ where I is the input image. Weight w is the derivative of the network output $S_c(I)$

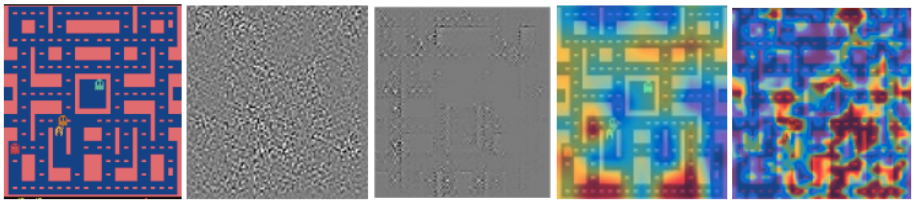


Figure 2: Input state for MsPacman game and saliency masks on trained DQN network in order (a) Pixel Saliency (b) Guided backprop (c) GradCAM (d) Mask based Optimization

with respect to the input point I_0 in the input image I. Pixel saliency is thus computed as $\frac{\partial S_c(I)}{\partial I}|_{I_0}$. For the Q value based reinforcement learning networks, we compute pixel saliency using network predicted Q value $Q(s, a)$ where s is the input state and a is the chosen action.

- Guided Backprop : This approach is very similar to pixel saliency with the difference that any gradients at the hidden layers h^j which become negative are discarded at the ReLU i.e. input regions that contribute negatively to the output are ignored. Here, weights w are computed as $\frac{\partial S_c(I)}{\partial I} = [h^j > 0 \ \forall j = \{0, 1, \ldots\}] \ \frac{\partial S_c(I)}{\partial I}|_{I_0}$. For our reinforcement learning agent, $S_c(I)$ is the approximated $Q(s, a)$ value for state s and action a.

- GradCAM : In this approach, class specific gradient information is exploited for more localized saliency analysis. Importance weights α_k^c of the k^{th} feature map of convolutional layer A is obtained by a global average pooling operation. Here, weights w for the saliency mask are computed through a weighted combination of feature maps, i.e. $w = ReLU(\sum_k \alpha_k^c A^k)$. The mask obtained is the same size as the feature maps of the chosen convolutional layer of the network and $\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial S_c(I)}{\partial A_{ij}^k}$. Similar to the previous two cases, for a Q value reinforcement learning network, we use $S_c(I) = Q(s, a)$ where required class c is analogous to network chosen action a. Since "more" higher level concepts are captured by deeper convolutional layers, last convolutional layer of the Q learning network is used to obtain the saliency mask.

- Mask Based Optimization : In this approach, computation of saliency mask is formulated as a regularized empirical risk minimization problem. A local perturbation operator $[\phi(x_0; m)](u)$ is introduced where $m : \Lambda \rightarrow [0, 1]$ is a mask that associates each pixel $u \in \Lambda$ with a scalar value $m(u)$. Here, $\Lambda = \{1, 2, \ldots, H\} \times \{1, 2, \ldots, W\}$ is a discrete domain for an image of height H and width W. The term $\lambda_1 ||1 - m||_1$ in the objective encourages the smallest deletion mask that causes the network output $f_c(\phi(x_0; m))$ to drop the most, where c is the target class. To obtain a mask that is more representative of natural perturbations, m is regularized in total variation norm and upsampled from a low resolution version, represented by $||\nabla m(u)||_\beta^\beta$ in the objective. $M(v) =$

$\sum_u g_{\sigma_m}(\frac{v}{s-u})m(u)$ is the upsampled mask and g_{σ_m} is a 2D gaussian kernel. Perturbation combines a blurred version of the image, regular image and the upsampled mask. The final optimization objective is :
$m^* = \min_{m \in [0,1]^\Lambda} \lambda_1 ||1 - m||_1 + \lambda_2 \sum_{u \in \Lambda} ||\nabla m(u)||_\beta^\beta + \mathbb{E}_\tau[f_c(\phi(x_0(-\tau), M))]$.

2.2.3 Drawbacks of Current Approaches

Pixel saliency makes linear approximations of the decision boundary and hence computes only local gradients. GradCAM combines class specific gradient information flowing into the last convolutional layer of the CNN. While this is useful for supervised learning, this may not yield good saliency maps for reinforcement learning. This is because the reinforcement learning networks are not trained for an input image to class label mapping and there is a high diversity in the input states for which the network chooses an action label "UP" for instance. For perturbation based methods, such as the mask based optimization approach, identifying perturbations appropriate for a given input domain is important. In the next discussion, we motivate why object level perturbations that form the base for "Object Saliency" may be most useful for describing why a network chooses specific actions. We do this by showing that input pixels that belong to objects are most important for a reinforcement learning network's decisions in Atari games.

2.3 Importance of Objects in Neural Network's Decisions

In this section we illustrate that objects in the input image are most crucial in determining a reinforcement learning network's decision. To show this, we first introduce the concept of adversarial noise injection. Adversarial noise is an input signal designed to drastically change the output of a trained neural network but still remain imperceptible to a human. This intriguing vulnerability of deep networks has been discussed by Goodfellow et al. [9] and several others [10],[36], [27]. Huang et al. [16] and Behzadan et al. [2] showed that adversarial noise can be injected for reinforcement learning agents such that an input state remains visually indistinguishable to a human but the network would choose different actions from the standard non-adversarial input. Fig.[3] shows noise injected by a simple process- Fast Gradient Sign Method [9] to one game frame of Atari2600 Freeway environment. For the original image without noise, the network predicts the action "UP" and for the state with noise added, the network predicts an action "DOWN".

Fast Gradient Sign Method generates an adversarial input x^* by maximizing a network loss function $J(\theta, x, y)$ with the one step update shown in Eq.[2]. The adversarial input is bound by L_p norm constraint, $p = \{1, 2, \infty\}$ of the form $||x^* - x||_p \leq \epsilon$. The bound ϵ is chosen to be small enough to ensure that the noise in the adversarial input to the network is imperceivable to the human eye.

$$x^* = x + \epsilon \text{sign}(\nabla_x J(x, y)) \tag{2}$$

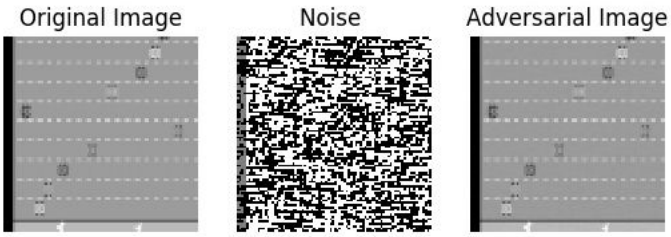


Figure 3: Original image without noise, FGSM crafted adversarial noise and the new adversarial state

Here, $J(\theta, x, y)$ is the cross entropy loss between the network prediction vector $Q(s)$ for given input state s and a distribution that maximizes the weight of the chosen action $a \in \pi_\theta : \mathcal{S} \rightarrow \mathcal{A}$. Other simple approaches for generating adversarial noise in images include iterative FGSM introduced by Kurakin et al. [20] and optimization based approach introduced by Szegedy et al. [36] and a method for blackbox attack on networks is presented in [26].
Parameter ϵ controls the amount of noise which can be introduced to pixels in the input image. In Fig.[5], we show the adversarial noise that is injected by the FGSM process with decreasing values of ϵ. As the allowed perturbation in pixels decreases, we find that noise is injected to only pixels belonging to specific objects of interest in each frame and not to any background pixels. RL agent decisions can thus be well explained by observing the effect of individual objects on the network output.

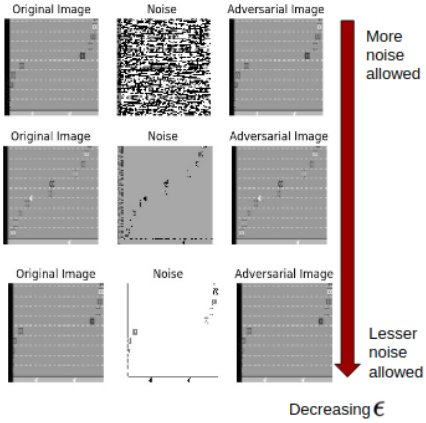


Figure 4: Original frame from Atari game Freeway, Injected noise and Adversarial frame generated with Decreasing ϵ

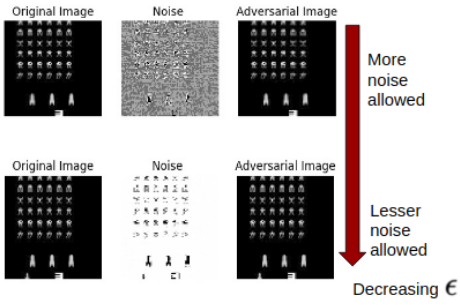


Figure 5: Original frame from Atari game Space Invaders, Injected noise and Adversarial frame generated with Decreasing ϵ

2.4 Object Saliency Computation

In this section, we introduce the concept of object saliency which yields more human interpretable heatmaps. These saliency maps highlight both relative influence and positive or negative valence of objects in the game environment for the current network decision. The first step in computing object saliency maps is identifying objects in the game environment. We introduce the segmentation algorithm used to identify distinct objects in Sec.[2.4.1] and introduce the computation of object saliency in Sec.[2.4.2].

2.4.1 Object Segmentation and Masking

We use a graph based image segmentation approach introduced in [6] to identify all objects in each input frame of a given Atari game. Here the input image is represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and the output is a segmentation of $\mathcal{V}$ into $\mathcal{S} = \{\mathcal{C}_1, \ldots, \mathcal{C}_r\}$ components. Each vertex of the graph is a pixel in the input image and the corresponding edges between the pixels $(v_i, v_j) \in \mathcal{E}$ has a corresponding weight, $w(v_i, v_j)$, that is a non-negative measure of dissimilarity between the two pixels. For every component in the output segmentation, $Int(\mathcal{C}, \mathcal{E}) = \max_{e \in MST(\mathcal{C},\mathcal{E})} w(e)$ is the largest weight in the minimum spanning tree of the component $\mathcal{C}$. We start with an initial segmentation, $\mathcal{S}^0 = \mathcal{V}$ and for every $q = 1, 2, \ldots, |\mathcal{E}|$, $v_i \in \mathcal{C}_i^{q-1}$, $v_j \in \mathcal{C}_j^{q-1}$ we use a comparison measure, $w(v_i, v_j) < MInt(\mathcal{C}_i^{q-1}, \mathcal{C}_j^{q-1})$ to decide if the two components $\mathcal{C}_i^{q-1}$ and $\mathcal{C}_j^{q-1}$ are to be merged or not. Here, $MInt(\mathcal{C}_1, \mathcal{C}_2) = \min$ ((Int$(\mathcal{C}_1)+\tau(C_1)$, Int$(\mathcal{C}_2)+\tau(C)_2)$) and $\tau(C)$ is a threshold parameter to control the size of each component given by $\tau = k/|\mathcal{C}|$. The edge set $\mathcal{E}$ is constructed in a 8 connected manner with an edge weight function based on absolute intensity of the pixels corresponding to the two vertices p_i, p_j, i.e. $w(v_i, v_j) = |I(p_i) - I(p_j)|$. Color images are treated as three separate monochrome images with the same edge weight function used for each channel individually and components are merged if the condition described in the algorithm is satisfied for all three channels.

This approach allows the graph to be constructed in $\mathcal{O}(n)$ where n is the number of pixels in the input image and segmentation is completed in $\mathcal{O}(n \log n)$.
Fig.[6] shows the output of the segmentation algorithm as applied on Atari game SpaceInvaders alongwith the extracted objects of interest for the corresponding frame. Here, we apply an initial gaussian smoothing determined by parameter σ and use $\sigma = 0.2, k = 300$. We merge all components smaller than a given minimum size $minS$ of 5 pixels as a final post processing step.

For all extracted objects, we mask the presence of an object in the given image with a fast marching method based image inpainting technique introduced by Telea et al. [37]. We use the extracted objects from segmentation as masks for inpainting. The algorithm starts from the boundary of the object and moves inward while replacing each pixel in the object with a normalized weighted sum of all the known pixels in its neigbourhood. More weight is given to pixels lying near near the normal of the boundary, lying on the boundary contours and near the pixel under consideration. Once a pixel is inpainted,

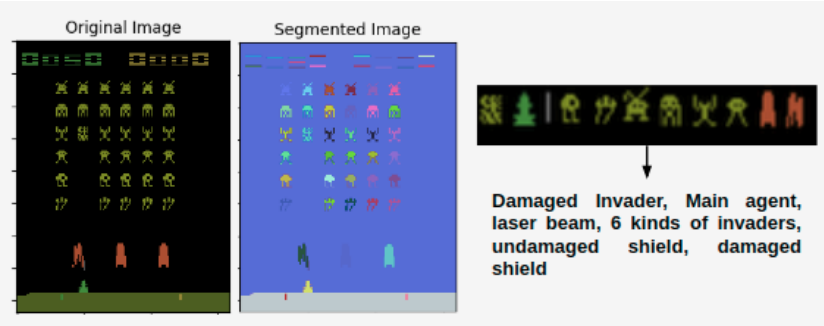


Figure 6: Extracting main objects from Atari SpaceInvaders input state frame

the next nearest pixel is identified using Fast Marching Method. FMM operates as a manual heuristic function and repeatedly ensures that pixels near the known pixels are inpainted first.

2.4.2 Object Saliency

We define object saliency as a coarse estimate of the influence of objects on a reinforcement learning network's decision. It is computed by identifying the effect of objects in a given state s based on their impact on $Q(s, a)$. However, calculating the derivative of $Q(s, a)$ with respect to the objects is nontrivial. Therefore, we use a simpler method. For each object O found in s, we mask the object with the background color to form a new state s_o. We calculate the Q-values for both the masked and unmasked states, and the difference of the Q-values gives an estimate of object influence. Thus object saliency map is computed as $w = Q(s_o, a) - Q(s, a)$. Here, if masking the object gives a higher future expected reward to the agent, the difference will be positive. That means lighter objects do not support the agent's action in the given state and darker objects in the saliency map support the agent's current action. Fig. [7] shows an example object saliency corresponding to a given input state from Atari 2600 Pacman game. In this game state, Pacman chooses an action "LEFT" and we observe from the saliency map that :

- The pellets towards the up and left direction of Pacman are darker and hence support the action of the agent. The pellets on the top are lighter and do not support the current action of the agent.
- The row of pellets in the down and right direction row also support the chosen-action left. Two close future paths for the agent are to move left and eat the row of pellets in the up direction or move down and eat the row of pellets in the right direction.

Similarly, Fig. [8] shows object saliency maps corresponding to three input states from Atari 2600 Frostbite game. In the first game frame, the agent can choose to jump up or down to collect frozen tiles in either row 2 or row 4 and we observe from the saliency

map that the object with greatest attractive influence is in the lower most row i.e 4 and the agent jumps down to move to that particular tile. In the second frame, the agent chooses to jump to the row of tiles above it and from the object saliency map, we can observe that the collectible frozen tile in the row immediately above the agent has a high positive influence in this current state. We also observe that the influence of already collected tiles (non-white) in the lower row is negligible in the current state. In the third game frame, we observe that the agent moves towards the completed igloo and expects very low aggregate rewards from any of the frozen tiles in the three rows.

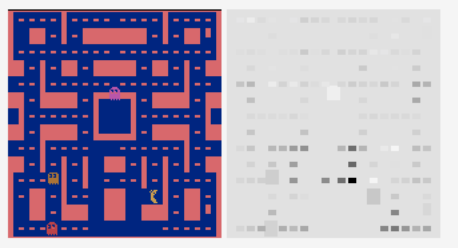


Figure 7: MsPacman game frame and corresponding object saliency

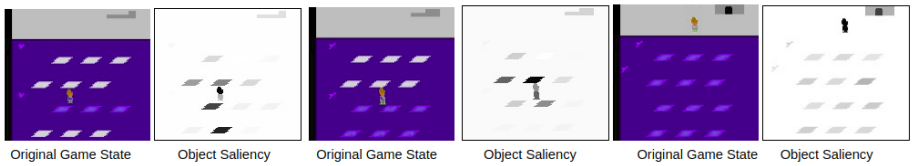


Figure 8: Frostbite game frames and corresponding object saliency

2.5 Comparison of Saliency Methods

Different techniques for saliency analysis as applied to different RL agent architectures lead to visualizations that differ in multiple metrics including robustness, computational time and human intuitiveness. In this section, we compare object saliency with other existing approaches using a quantitative metric called Area Over MoRF Perturbation Curve (AOPC metric).

2.5.1 Background - AOPC Metric

In the supervised learning domain Samek et al.[28] introduce a quantitative metric to evaluate the quality of a given saliency analysis method called AOPC - Area Over MoRF Perturbation Curve. This approach uses a greedy iterative technique (MoRF - Most Relevant First) to repeatedly remove information from local regions around the most relevant pixels identified by the saliency approach in a given input image. The steps of the algorithm and important terminology used are summarized as follows :

1. Repeat for k random seeds of perturbation function $g(x, p)$
   2. Repeat for saliency mask $= 1, 2, \ldots, |\mathcal{S}|$
      3. Repeat for $p = 1, 2, \ldots |\mathcal{P}|$
         4. $x^{(p)} = g(x^{(p-1)}, p)$
            5. AOPC(s) = $\frac{1}{|\mathcal{P}|+1} \left\langle \sum_{k=0}^{L} (f(x^{(0)}) - f(x^{(p)})) \right\rangle_s$

- Set $\mathcal{S}$ is a collection of saliency masks obtained from different techniques (Pixel saliency, Grad CAM etc) for a given input image x^0.
- Set $\mathcal{P}(s)$ represents the pixels in a specific saliency mask $s \in \mathcal{S}$ arranged in descending order of importance.
- Perturbation function $g(x, p)$ removes information by replacing all pixels in a 9×9 neighborhood around pixel $p \in \mathcal{P}$ in an input image x.
- $f(x)$ is the RL Network decision function value for an input x, i.e. $\sigma(Q(x, a))$ for input state x and network predicted action a for the input state where σ is the softmax function.

The algorithm computes difference in certainty of network decision for predicting a specific label as increasingly larger perturbations are applied to the original image vs a random baseline. In a random baseline, an arbitrary importance of pixels in the input image is chosen. If an increase in the amount of perturbation increases the difference in network prediction certainty for a label, it means that pixels very relevant to the decision of the network were identified by the approach. If the difference in network prediction certainty for a label with increasing perturbation vs a random baseline is low, it means that the pixels identified were not very important to the network decision. Thus, methods with a greater area under the MoRF perturbation curve perform better.

To make the evaluation of AOPC metric applicable to a reinforcement learning domain we average our results over all frames (after the default “no-op” steps) in one

entire test episode played by a fully trained DQN network. This is done in lieu of averaging over all images in a specific dataset as was done in the original paper. In Fig.[9], we show the AOPC plots for different saliency approaches averaged over one episode of MsPacman. For the AOPC measure, a high value of difference $f(x^{(0)}) - f(x^{(p)})$ indicates an important pixel was disturbed and consequently a larger value of AOPC indicates that on average, the given saliency method identified more sensitive regions first.

2.5.2 Discussion

1. We observe from the plot that object saliency has the highest set of AOPC values that consistently increase with increasing perturbation steps.

2. We observe that for the initial set of perturbation steps, which are the most important, optimization based mask method has a high AOPC value that is comparable to object saliency. Optimization based mask approach aims to identify an explanation that is sparse, maximally informative and smooth. The fact that the approach has high AOPC values corresponding to the initial set of perturbation steps indicates that the method successfully identifies a minimal mask to explain the network decision that ranks high on the AOPC metric.

3. Pixel saliency and Guided Backprop are similar approaches that operate on raw gradients. Guided Backprop can be seen to achieve marginally higher AOPC score than Pixel saliency. Interestingly, both methods receive AOPC scores below zero in the intermediate perturbation steps and this could mainly be due to the highly local nature of explanation generated.

4. GradCAM has the lowest AOPC values and even receives a negative score through some later perturbation steps. This could be because GradCAM uses class specific gradient information flowing into the last convolutional layer of the CNN for explaining a decision of interest. This method is useful for supervised learning domains such as classification where the network is optimized for an input image to output label mapping. In this supervised learning scenario, the network learns to identify specific "high-level" features corresponding to given output labels and GradCAM qualitatively identifies these well. Poor performance in RL networks could be because the network weights in RL are optimized for reducing the Bellman error and not trained to map a given input state to a specific action label. The diversity in input states is high corresponding to a given output label (action) such as "UP" or "DOWN". Hence, GradCAM may not be a suitable approach for saliency in RL networks.

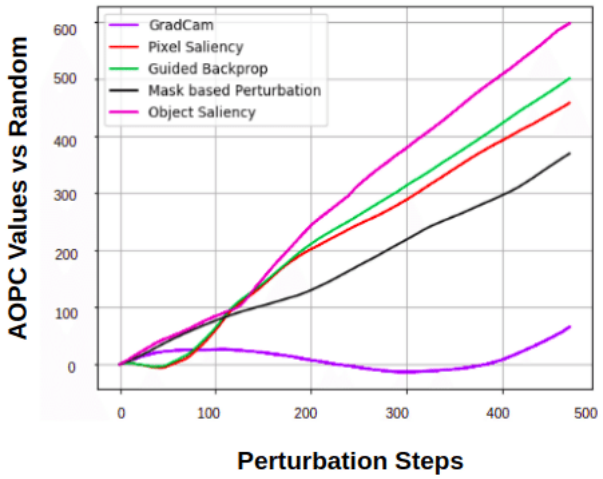
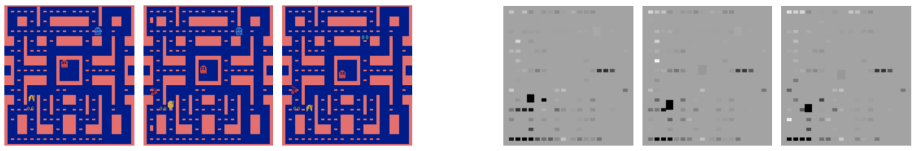


Figure 9: AOPC curves on a pre-trained DQN agent

2.6 Studying Intuitiveness of Object Saliency through Human Experiments

In order to test whether the object saliency maps can help humans understand the learned behavior of a reinforcement learning agent, we performed an initial set of experiments. These were carried out on Atari 2600 MsPacman game. The goals of the experiment were the following :

1. Test whether object saliency maps contain enough information to allow humans to match these maps with corresponding game scenarios.
2. Test whether participants could use object saliency maps to generate reasonable explanations of the behavior of the main agent (Pacman).
3. Test whether object saliency maps allow participants to correctly *predict* the main agent's (Pacman) next action. This requires a deeper causal understanding of what may influence the Pacman in its decisions.



(a) Screen-shots (b) Object Saliency Maps

Figure 10: An example of the stimulus materials participants saw in the test 9 of the prediction task. 75% participants in the screen-shot group thought Pacman would go left to eat the cherry at the left side. 60% participants in the object saliency maps group predicted the Pacman would keep going down for the dark elements (the pellets) below.

Each experiment consists of two tasks:

- **Matching Task :** In each trial, the participants are shown a 5-second video clip of Pacman gameplay generated by a trained DQN network twice. During the video clip, Pacman decides and takes particular actions. The last action taken by Pacman involves a crucial movement in the clip (eg. Pacman moves right instead of going up at an intersection), with the clip ending just after the crucial movement. Three frames from the object saliency map are then shown to participants (see Fig.[10(b)]). The center frame shows the state when Pacman makes the crucial decision and the other two are frames from before and after that instant. In this task, participants are asked to judge whether the saliency maps accurately represent the video they just saw. In the matching cases, the saliency maps shown were all generated from the video clip the participants saw. In the non-matching cases, the three saliency maps were generated from a different video clip. In distractor/non-matching clips, Pacman occupies the same

area of map as in the target video, but makes different movements. This is done to avoid the case where the participants focus solely on the location of the Pacman as a matching criterion, disregarding the movements and other factors in the game state.

Following the match decision, if the participants' answer is "match", they are asked to give an explanation for the Pacman's movements based on the video and saliency maps. In other words, participants are asked to provide a teleological explanation as to 'why' Pacman acted as it did. For example, "Pacman moved up to eat more energy pellets while avoiding the ghost coming from below."

The matching task consisted of 2 training trials and 20 test trials, half (10 trials) presenting matched video and saliency maps, the other half presenting non-matched pairs in a single randomly ordered sequence. Dependent variables were correctness of matches and agreement between explanations and saliency maps.

- **Prediction Task :** In each trial, the participants are shown a video clip not used in the matching task. Each clip ends at the point where Pacman must choose a crucial move. The participants are divided equally into two experimental conditions. In scenario 1 (screen-shot condition), after the video clip, participants see 3 actual screen-shots from the video ending before the crucial move is taken. In scenario 2 (the object saliency map condition), the participants see three object saliency map frames (corresponding to the screen-shot frames) after viewing the video clip (see Fig.[10]). At the decision point in the third frame Pacman's choices (up, down, left, right) may be limited by barriers indicated on the response forms. Participants are asked to predict Pacman's movement among the feasible directions based on the three previous frames (screenshots or saliency maps), and then give an explanation for their prediction which includes their judgment as to which elements of the game influenced the Pacman's decision (indicating these elements by circling them on a hardcopy of the screenshot or saliency map), and explain why Pacman made that decision.

  The prediction task consisted of 2 training trials and 10 test trials. Each participant was assigned to either the screenshot group or the saliency map group. Dependent variables include whether predictions were correct, and whether explanations were consistent with the saliency maps.

2.6.1 Results :

The average matching accuracy of the participants was 61.0% ($SD = 14.0\%$). A learning effect was found with participants having higher accuracy (65.5%) in the last half of the trials than the first half (56.5%) ($t(39) = 3.10, p = 0.04$). Comparing hit and false alarm rates, participants reported more "matches" when the video and image stimulus matched ($t(18) = 2.91, p < 0.001$). If the 40 participants are treated as a binary classifier and the percentage of their answers as an output score, a receiver operating characteristic (ROC) curve introduced by Fawcett et al. [5] can be plotted for true positive rates versus false positive rates across a range of threshold parameters (as Fig. 11 shows). The area under the curve is 0.81 which indicates a good classification

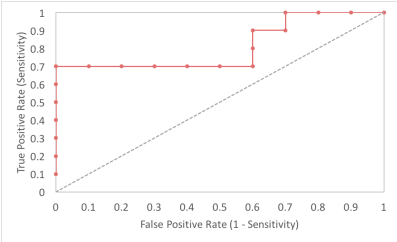


Figure 11: ROC curve of the matching task, AUC = 0.81.

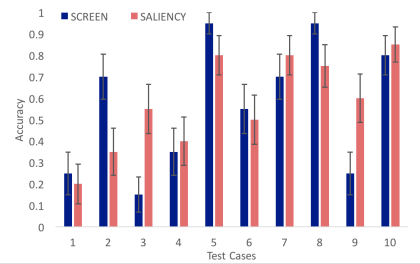


Figure 12: The mean accuracy of participants in each test cases of the prediction task. Error bars are one Standard Error from Means.

between matching and non-matching situations. In summary, human participants were able to link the object saliency maps with the game scenarios.

For the more difficult prediction task, there was no significant difference in accuracy between the object saliency map group ($58.0\% \pm 12.8\%$) and the control group($56.5\% \pm 10.4\%$). However, the main effect of trials ($F(9, 342) = 11.18$, $p < 0.001$) and the interaction between trials and groups ($F(9, 342) = 2.72$, $p = 0.005$) were both highly significant suggesting that characteristics of the trials had a strong influence on performance. Thus, we conducted a simple effect analysis to examine differences among the 10 test scenarios (see Fig. 12). Results show that the screen-shot group has high predictive accuracy in test 2 ($p = 0.027$), while the object saliency map group has higher accuracy in tests 3 and 9 ($p = 0.007, p = 0.025$). These three trials can help provide a deeper insight into the mechanism of how object saliency maps could help humans understand Pacman's learned behavior.

2.6.2 Discussion :

The result of our human experiments show that object saliency maps can be linked to corresponding game scenarios by participants, a prerequisite if they are to provide explanations of behavior. Object saliency maps and screen shots proved equally helpful to humans in predicting Deep Reinforcement Learning Network's behavior.
For the prediction task the group viewing screen shots had access to rich contextual information including obstacles in the environment and the identity of objects making the association between the frames they viewed and rules of the game explicit. The object saliency participants, by contrast, lacked clear identity of objects or environmental features but viewed the valence of objects affecting the decision (via the object shading). That these complementary representations led to equal performance suggest they are both of value and deserve closer attention. In watching a program such as DQN based RL agent play Pacman it is tempting to interpret its actions in a form interpretable to

humans i.e. seeking pellets and avoiding ghosts. This in fact is what we asked participants in the screen-shot group to do. RL agents however have no knowledge about game rules and simply learn to maximize expected reward over time. In many cases due to the reward structure of the game its decisions may happen to match our own and we can attribute teleological causes. However, in other states, the strangeness of its decision making is revealed and we must turn to tools such as the saliency map for help. Such tools offer a better chance to improve the model when the agent executes unexpected or abnormal behaviors (e.g. debugging and testing of the Deep Reinforcement Learning Networks). Alternately, the agent's policies could be examined to identify why they may have been learned and what benefits they might confer, leading to a deeper understanding of the domain and improved decision making.

Performance of the object saliency group on the prediction task may have suffered due to insufficient training and limitations inherent in group testing. We believe that a more comprehensible tutorial and longer training section might lead to better understanding of the object saliency map and improved performance in both tasks. As Fig.[12] shows, the performance pattern of participants in the prediction task depends crucially on situations. If states readily predicted from screen shots can be discriminated from those which are not easy to predict or amenable to naive explanation, saliency maps can be tested and used to support understanding of network policy.

2.7 Building rule based descriptions for object saliency maps

Compared to other saliency approaches, object saliency maps are much more human interpretable. However, they still remain difficult for humans to analyze on a per frame basis. We construct some strict rules for template based textual descriptions of object saliency maps for two purposes. When transparency is the ultimate goal of saliency analysis, an ability to construct textual descriptions from saliency maps will be very important for untrained users. Also constructing such rules helps us verify that there is indeed a consistent description that we can use to understand the reasons for agent behavior from object saliency maps.

2.7.1 Related Work

An approach discussed in [13] presents techniques for AI rationalization i.e. neural machine translation based "human-like" descriptions of state-action mapping in Atari games. This is based on an annotated training corpus of human utterances collected while playing Atari games. As one of our contributions, we develop rule based verbal descriptions of the agent policy but we note that this is a means to give human like explanations of the actual network decisions as opposed to a natural language description of current state-action pairs.

2.7.2 Describing the rules

The main steps followed to construct the descriptions are :

- We compute reward saliency as (object saliency map at state s_{t+1} relative to object locations in state s_t) - (object saliency map at s_t).
- We compute the centroid of Pacman at state s_t and s_{t+1} as c_t and c_{t+1}.
- We compute the list of valid directions from checking presence of walls in the four directions around Pacman.
- We define $\mathcal{R}_t$ as a region of 70px around Pacman's centroid and construct a set UU_t which contains for every object, a label about its location in one of the eight directions as per Fig.[13].

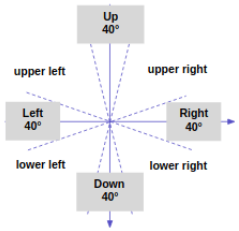


Figure 13: Eight directions used for set $\mathcal{UU}_t$ and set $\mathcal{U}$

- We also construct $\mathcal{C}_t$ to store the centroids of objects found in $\mathcal{R}_t$, a set $\mathcal{S}_t$ to store the saliency values, $\mathcal{N}_t$ to store the object label such as "pellet" or "cherry".
- We compute the total saliency of objects in every quadrant of $\mathcal{R}_t$ and store it in U_t. The quadrant with most attractive influence and least attractive influence is chosen. The object with the most negative saliency value and most positive value is attributed as strongly supporting and strongly opposing the agent's current action respectively.
- We identify if Pacman moves towards or away from the objects of interest by computing the shortest path in the pacman maze using Breadth First Search.
- We repeat a similar procedure for the object saliency map at s_t also. Once the template words are identified, we generate the textual descriptions as defined in the following algorithm.

Algorithm 1: Rules for object map description

Result: Template Based Description of Saliency Maps
Compute Reward Saliency 1. Compute ValidActions, Action ← Current Action
2. Text ← Valid Actions for Pacman are "ValidActions"
1. Initialize $U_{8\times1}= [0]$
2.Compute c_{t+1}, c_t & Identify $\mathcal{R}_t$
3.Compute $\mathcal{C}_t, \mathcal{S}_t, \mathcal{N}_t, UU_t \ \ \forall$ objects $\in R_t$
4. $\forall$ objects $i \in \mathcal{R}_t : U[UU_t[i]] \leftarrow U[UU_t[i]] + \mathcal{S}_t[i]$
5. Determine (DIRECTION, Index) = $\max(U)$; (DIR, In) = $\min(U)$
6. Compute $M, P : \min(\mathcal{S}_t)$ s.t. $M < 0;\ UU[P] =$ Index
7.Compute $m, p : \max(\mathcal{S}_t)$ s.t. $m > 0;\ UU[p] =$ In
8. Ob1 = $\mathcal{N}_t[P]$, Ob2 = $\mathcal{N}_t[p]$
9. $d1_t \leftarrow BFS(\mathrm{c}_t, \mathcal{C}_t[P])\ d1_{t+1} \leftarrow BFS(\mathrm{c}_{t+1}, \mathcal{C}_t[P])$
10. $d2_t \leftarrow BFS(\mathrm{c}_t, \mathcal{C}_t[p])\ d2_{t+1} \leftarrow BFS(\mathrm{c}_{t+1}, \mathcal{C}_t[p])$
Construct templates from prior knowledge of game. For MsPacman :
if $d1_t > d1_{t+1}$ **then**
  Pacman's current action "ACTION"is strongly supported by object "Ob1" in the "DIRECTION" direction. Pacman is moving towards it. Pacman's current action "ACTION"is strongly supported by object "Ob1" in the "DIRECTION" direction. Pacman could be moving away from it.
end
if $d2_t > d2_{t+1}$ **then**
  Pacman's current action "ACTION"is strongly opposed by object "Ob2" in the "DIR" direction. Pacman could be moving away from it. Pacman's current action "ACTION"is strongly opposed by object "Ob2" in the "DIR" direction. Pacman is moving towards it.
end
Repeat for object saliency and fill template : Aggregate - Pacman is attracted/ opposed by object "Ob1/Ob2" in the "DIRECTION/DIR" direction.

2.7.3 Descriptions generated from Saliency Maps

The following examples show rule based verbal descriptions generated for two states in an Atari game.

Fig.[14] shows one game state from Atari MsPacman and its corresponding reward and object saliency map. The following verbal description is generated using the rules and templates :
Valid actions for Pacman are up, right, down
Pacman's current action "Downright" is strongly supported by object "Pellet" in "RIGHT" direction. Pacman is moving towards it.
Aggregate - Pacman is "attracted" by object "Pellet" in "Down" Direction.

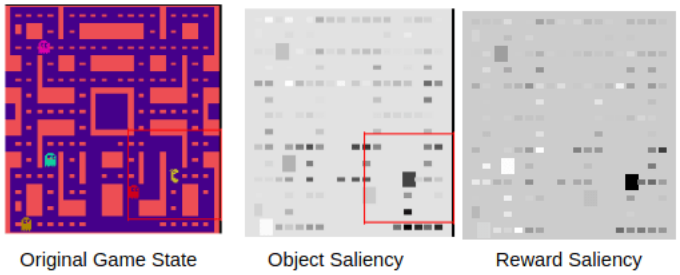


Figure 14: Original game state with object saliency map and reward saliency map respectively for one input state in MsPacman

---

Fig.[15] shows one game state from Atari MsPacman and its corresponding reward and object saliency map. The following verbal description is generated using the templates and rules :
Valid actions for Pacman are left, right, down
Pacman's current action "Upright" is strongly supported by object "Pellet" in "UP" direction. Pacman is moving towards it.
Pacman's current action "Upright " is strongly opposed by object "Pellet" in "LEFT" direction. Pacman is moving away from it.
Aggregate - Pacman is "attracted" by object "Pellet" in "Upright" Direction.

2.8 Merits of Object Saliency - Detecting Adversarial Noise

As introduced in Sec. 2.3, adversarial noise can be injected into game frames in a manner such that network decisions are completely altered for the adversarial state vs the original state, both of which are however indistinguishable to a human. We study the differences between object saliency and pixel saliency on game states with adversarial noise added and demonstrate the results in Fig.[16] and Fig.[17]. We observe that ab-

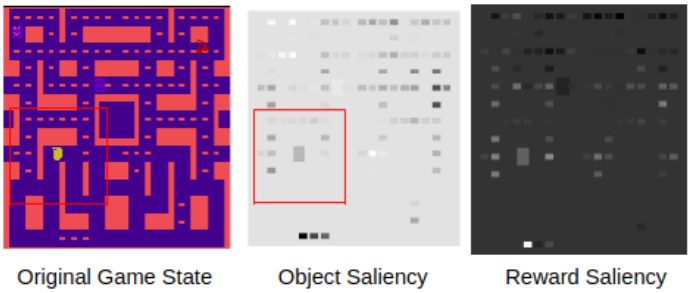


Figure 15: Original game state with object saliency map and reward saliency map respectively for one input state in MsPacman

normalities in an object saliency map are distinct whereas in other saliency approaches, such as pixel saliency, even though differences can be seen, it is not immediately apparent if noise has been added to the image. Here, images in Fig.[16] are shown for Atari Game Freeway and images in Fig.[17] are shown for Atari game SpaceInvaders.

- The first and second frames in the images represent the original game state and state with adversarial noise added in a manner that is difficult for a human to detect unassisted.
- The third and fourth frames show object saliency on the original game state with no noise and game state with adversarial noise added. We observe that object saliency generates very distinct results for the two cases and it is possible for an unassisted human or a simple classifier to learn to detect the noise based on the object saliency maps.
- The fourth and fifth frames in these images show the result of pixel saliency applied to the original game state and the adversarial game state respectively. We observe that while there are differences between these two maps, the lack of interpretability of this method makes it very difficult for a human to detect the addition of noise simply by inspecting the output maps.

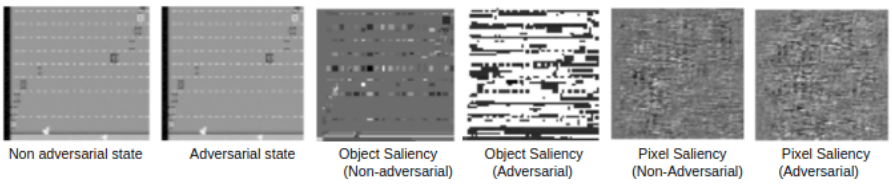


Figure 16: Atari Game Freeway - i) Original Game State ii) Input with Adversarial Noise Added iii) Object Saliency on Original Image iv) Object Saliency on Image with Adversarial Noise v) Saliency on Original Image vi) Saliency on image with Adversarial Noise Added

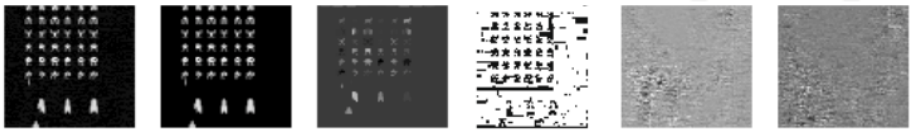


Figure 17: Atari Game SpaceInvaders - i) Original Game State ii) Input with Adversarial Noise Added iii) Object Saliency on Original Image iv) Object Saliency on Image with Adversarial Noise v) Saliency on Original Image vi) Saliency on image with Adversarial Noise Added

2.9 Merits of Object Saliency - Identifying Differences in Value Functions

We study here visually recognizable and human understandable differences in object saliency maps obtained for two trained DRL agents with significant differences in architecture as well as training processes and average score during test episodes. We compare two RL agents - Rainbow and Value Prediction network (VPN). We choose the Rainbow agent because it combines a wave of improvements introduced in the last few years for deep reinforcement learning in one single model to achieve current state of the art scores. We choose the value prediction network agent because it uses a modular architecture to learn components of the RL MDP model including reward, discount factor, abstractions of the next state as well as the value of the state and explicitly plans a set of actions for a specified number of future steps.

2.9.1 Rainbow Architecture Overview

The rich representation given by a deep neural network allows the Q function to be approximated by a parameterized network $Q(s, a; \theta)$ as shown by Mnih et al. [23]. The parameter θ is learned iteratively by minimizing a sequence of loss functions, $L_i(\theta_i) = \mathbb{E}(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_i) - Q(s_t, a_t; \theta_i))^2$. DQN can reach human-level performance on many Atari 2600 games but suffers substantial overestimation in some other games. Thus, a Double DQN (DDQN) was proposed by Hasselt et al. [38] to reduce overestimation by decoupling the target max operation into action selection and action evaluation. Two independent Q value functions are learned and the recursive bellman update equation is modified to the form $Q(s_t, a_t) = r_t + \gamma \tilde{Q}(s_{t+1}, \arg\max_{a_t} Q(s_{t+1}, a_t))$. Wang et al. proposed a dueling network architecture (DuelingDQN) [40] that decouples the state-action values into state values and action values to yield better approximation of the state value. That is, the Q value estimation is split into $Q(s, a) = V(s) + A(a)$ where $A(a)$ is the advantage function. Schaul et al. [29] replaced uniform replay with a prioritized experience replay memory to give the agent a higher probability of picking samples with a greater TD-error, allowing important transitions to be replayed more often during learning. The Distributional RL proposed by Bellemare et al. [3] models the distribution of value function instead of

its expected value to obtain state of the art results on several atari games. They approximate distributions d_t with a discrete support z and probability mass $p_\theta(s_t, a_t)$, construct a distributional variant of Q learning and finally minimize KL divergence between an L2 projection of target distribution $(\Phi_z d'_t)$ and orginal distribution d_t i.e. minimize $D_{KL}(\Phi_z d'_t || d_t))$ where $d'_t = (r_{t+1} + \gamma_{t+1} z, p_\theta(s_{t+1}, at + 1))$. Fortunato et al. [8] introduced a means for better exploration by adding parametric noise to RL network weights with a low computational overload. Replacing a standard linear layer with a layer that combines a deterministic and noisy stream allowed the network to learn to ignore the noise at different rates in different states, thus introducing a state conditional exploration. With a large number of individual improvements to the original DQN architecture, a recent work by Hessel et al. [15] combines several of these individual improvements along with multi step returns to form one integrated agent referred to as Rainbow network which gives state of the art performance in terms of training efficiency as well as agent score across multiple Atari 2600 games.

2.9.2 Saliency Maps on Rainbow

Fig.[18] and Fig.[19] show the result of object saliency and pixel saleincy methods as applied to three different game frame states for the Rainbow agent.

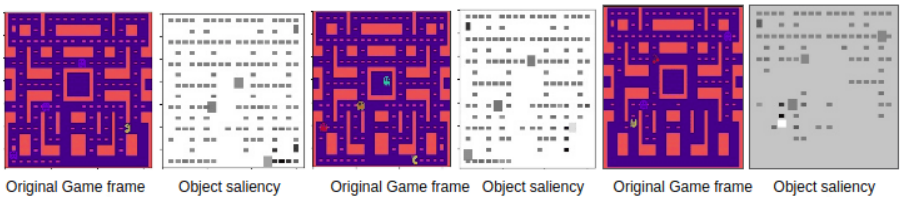


Figure 18: Object saliency maps on three game frames of MsPacman on a trained model with Rainbow architecture

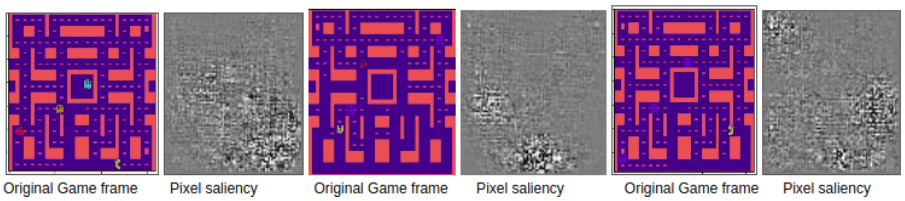


Figure 19: Pixel saliency maps on three game frames of MsPacman on a trained model with Rainbow architecture

2.9.3 Value Prediction Network Overview

While planning on model based approaches will yield maximum interpretability of network actions, learning correct representative models of the real world or stochastic environments is very hard. An Encoding-Transformation-Decoding architecture for Action Conditional Video Prediction in 2D Atari game environments was introduced in Oh et al. [24] and an extension by Leibfried et al. [22] for simultaneous next state and reward prediction. While this network makes qualitatively reasonable predictions, small objects such as "laser beam" in Atari Space Invaders, infrequently observed objects such as "cherry" in Mspacman or stochastic "ghosts" are not learned well. Value Prediction Network attempts to overcome this by learning to predict abstractions of next state instead of the entire game frame and is introduced by Oh et al. [25]. It uses a modular architecture with a separate Encoding module - to determine an abstract state for a given observation, Value module - to determine value of an abstract state, Outcome module - to determine reward and discount factor for an abstract state under a chosen sequence of primitive actions and Transition module - to determine the next abstract state in an option conditional manner.

2.9.4 Saliency Maps on VPN agent

Fig.[20] and Fig.[21] show the result of object saliency and pixel saleincy methods as applied to three different game frame states for the VPN agent.

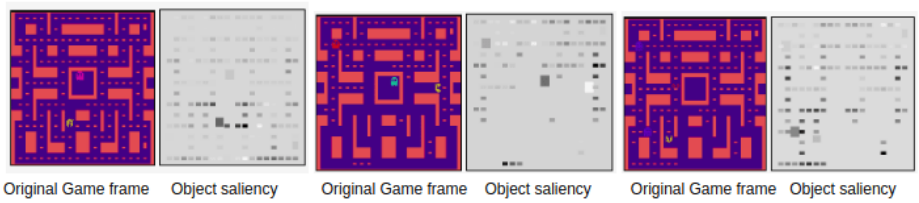


Figure 20: Object saliency maps on three game frames of MsPacman on a trained model with VPN architecture

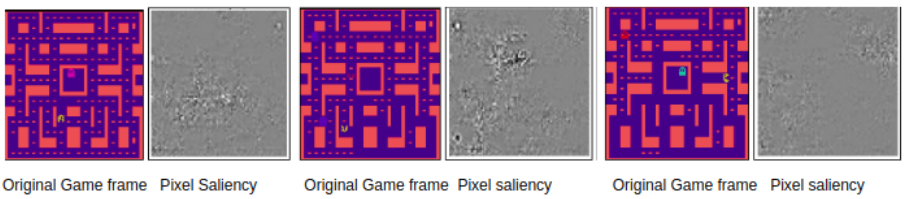


Figure 21: Pixel saliency maps on three game frames of MsPacman on a trained model with VPN architecture

2.9.5 Discussion

- Rainbow agent receives almost double the average score that a VPN agent receives on the Atari game MsPacman.
- Object saliency map reveals that Rainbow agent learns to associate a high cumulative return from pellets close to the agent and along the path the main agent will take. All other objects far away from the main agent have equal influence on the value of action taken by the main agent in a given state.
- For the VPN agent, pellets in the direction of action of the main agent have a positive influence and pellets in the direction away from the the main agent have negative influence as expected. However, objects far away from the main agent also impact the decision of the main agent.
- From other saliency approaches, it is hard to interpret or draw any useful insights on reasons for differences between the performance of the two agents.

3 Conclusion and Future Work

3.1 Conclusion

In this work we discussed object saliency at length and demonstrated that it is a simple and effective means to explain network decisions for value based deep reinforcement learning agents. We first show that pixels corresponding to objects are most impactful for creating a change in network decisions to demonstrate that object level perturbations could be the most meaningful for the purpose of describing the decision of a DRL agent. We then show that object saliency is more robust and identifies pixels of higher relevance when compared to other methods for saliency analysis using a quantitative metric. Further, we demonstrate through pilot human experiments and construction of template based descriptions that object saliency maps are human intelligible and better understandable compared to other existing approaches for saliency. Finally, we discuss some merits of object saliency over existing approaches - including the ability to detect adversarial noise injected into game states in a visually distinct manner as well as the ability to highlight differences in value function learned by DRL agents with different architectures and performances.

3.2 Future Work

Some directions for future work to improve transparency in reinforcement learning networks are :

1. We can train a second model from template based rules defined using object saliency maps to generate descriptions for causes of agent actions similar to the process of learning to generate "image captions" in the supervised learning domain. While this second network may not provide reasons that are completely correlated to actual causes for actions in the original network, it can still be promising to use neural attention mechanisms to trace which part of the input contributed most to generating specific components of the description.

2. We have developed a post-hoc explanation scheme for black box deep reinforcement learning networks. Another direction of methods include building intrinsically explainable models similar to Verma et al. [39] who represent policies using a domain specific high level programming language. They use a Neurally Directed Program Search (NDPS), for solving the challenging nonsmooth optimization problem of finding a programmatic policy with maximal reward. NDPS works by first learning a neural policy network using DRL, and then performing a local search over programmatic policies that seeks to minimize a distance from this neural "oracle". In related work by Hein et al. [14], genetic programming for reinforcement learning (GPRL) approach using a model-based batch reinforcement learning and genetic programming is used, which autonomously learns policy equations from pre-existing default state-action trajectory samples. Building on ideas from these approaches for domains more complex than those discussed is another promising direction for building interpretable reinforcement learning networks.

References

[1] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS one* 10.7 (2015), e0130140.

[2] Vahid Behzadan and Arslan Munir. "Vulnerability of deep reinforcement learning to policy induction attacks". In: *International Conference on Machine Learning and Data Mining in Pattern Recognition*. Springer. 2017, pp. 262–275.

[3] Marc G Bellemare, Will Dabney, and Rémi Munos. "A distributional perspective on reinforcement learning". In: *arXiv preprint arXiv:1707.06887* (2017).

[4] Mariusz Bojarski, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Larry Jackel, Urs Muller, and Karol Zieba. "VisualBackProp: visualizing CNNs for autonomous driving". In: *arXiv preprint arXiv:1611.05418* (2016).

[5] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern recognition letters* 27.8 (2006), pp. 861–874.

[6] Pedro F Felzenszwalb and Daniel P Huttenlocher. "Efficient graph-based image segmentation". In: *International journal of computer vision* 59.2 (2004), pp. 167–181.

[7] Ruth C Fong and Andrea Vedaldi. "Interpretable explanations of black boxes by meaningful perturbation". In: *arXiv preprint arXiv:1704.03296* (2017).

[8] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. "Noisy networks for exploration". In: *arXiv preprint arXiv:1706.10295* (2017).

[9] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[11] Sam Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. "Visualizing and Understanding Atari Agents". In: *arXiv preprint arXiv:1711.00138* (2017).

[12] Shixiang Gu, Ethan Holly, Timothy P. Lillicrap, and Sergey Levine. "Deep Reinforcement Learning for Robotic Manipulation". In: *CoRR* abs/1610.00633 (2016). arXiv: 1610.00633. URL: http://arxiv.org/abs/1610.00633.

[13] Brent Harrison, Upol Ehsan, and Mark O Riedl. "Rationalization: A Neural Machine Translation Approach to Generating Natural Language Explanations". In: *arXiv preprint arXiv:1702.07826* (2017).

[14] Daniel Hein, Steffen Udluft, and Thomas A Runkler. "Interpretable policies for reinforcement learning by genetic programming". In: *arXiv preprint arXiv:1712.04170* (2017).

[15] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. "Rainbow: Combining Improvements in Deep Reinforcement Learning". In: *arXiv preprint arXiv:1710.02298* (2017).

[16] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. "Adversarial attacks on neural network policies". In: *arXiv preprint arXiv:1702.02284* (2017).

[17] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter. "Control of a Quadrotor With Reinforcement Learning". In: *IEEE Robotics and Automation Letters* 2.4 (Oct. 2017), pp. 2096–2103. ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2720851.

[18] Michal Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaskowski. "ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning". In: *CoRR* abs/1605.02097 (2016). arXiv: 1605.02097. URL: http://arxiv.org/abs/1605.02097.

[19] Lukasz Kidzinski, Sharada Prasanna Mohanty, Carmichael F. Ong, Zhewei Huang, Shuchang Zhou, Anton Pechenko, Adam Stelmaszczyk, Piotr Jarosik, Mikhail Pavlov, Sergey Kolesnikov, Sergey M. Plis, Zhibo Chen, Zhizheng Zhang, Jiale Chen, Jun Shi, Zhuobin Zheng, Chun Yuan, Zhihui Lin, Henryk Michalewski, Piotr Milos, Blazej Osinski, Andrew Melnik, Malte Schilling, Helge Ritter, Sean F. Carroll, Jennifer L. Hicks, Sergey Levine, Marcel Salathé, and Scott L. Delp. "Learning to Run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments". In: *CoRR* abs/1804.00361 (2018). arXiv: 1804.00361. URL: http://arxiv.org/abs/1804.00361.

[20] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. "Adversarial examples in the physical world". In: *arXiv preprint arXiv:1607.02533* (2016).

[21] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. "Building machines that learn and think like people". In: *Behavioral and Brain Sciences* 40 (2017).

[22] Felix Leibfried, Nate Kushman, and Katja Hofmann. "A deep learning approach for joint video frame and reward prediction in atari games". In: *arXiv preprint arXiv:1611.07078* (2016).

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[24] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. "Action-conditional video prediction using deep networks in atari games". In: *Advances in Neural Information Processing Systems*. 2015, pp. 2863–2871.

[25] Junhyuk Oh, Satinder Singh, and Honglak Lee. "Value prediction network". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6120–6130.

[26] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. "The Limitations of Deep Learning in Adversarial Settings". In: *CoRR* abs/1511.07528 (2015). arXiv: 1511.07528. URL: http://arxiv.org/abs/1511.07528.

[27] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. "Practical black-box attacks against machine learning". In: *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM. 2017, pp. 506–519.

[28] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. "Evaluating the visualization of what a deep neural network has learned". In: *IEEE transactions on neural networks and learning systems* (2017).

[29] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. "Prioritized experience replay". In: *arXiv preprint arXiv:1511.05952* (2015).

[30] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *See https://arxiv. org/abs/1610.02391 v3* 7.8 (2016).

[31] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. "Mastering the game of Go without human knowledge". In: *Nature* 550.7676 (2017), p. 354.

[32] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps". In: *arXiv preprint arXiv:1312.6034* (2013).

[33] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. "SmoothGrad: removing noise by adding noise". In: *CoRR* abs/1706.03825 (2017). arXiv: 1706.03825. URL: http://arxiv.org/abs/1706.03825.

[34] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. "Striving for simplicity: The all convolutional net". In: *arXiv preprint arXiv:1412.6806* (2014).

[35] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge, 1998.

[36] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199* (2013).

[37] Alexandru Telea. "An image inpainting technique based on the fast marching method". In: *Journal of graphics tools* 9.1 (2004), pp. 23–34.

[38] Hado Van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning." In: *AAAI*. Vol. 16. 2016, pp. 2094–2100.

[39] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. "Programmatically Interpretable Reinforcement Learning". In: *arXiv preprint arXiv:1804.02477* (2018).

[40] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. "Dueling network architectures for deep reinforcement learning". In: *arXiv preprint arXiv:1511.06581* (2015).

[41] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. “Adaptive deconvolutional networks for mid and high level feature learning”. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2018–2025.