Towards Behavior-Level Explanation for Deep Reinforcement Learning

Xuan Chen*, Zifan Wang*, Yucai Fan, Bonan Jin
Piotr Mardziel, Carlee Joe-Wong, Anupam Datta
Carnegie Mellon Univeristy
Pittsburgh, PA 15213
xuanche2@andrew.cmu.edu

Abstract

While Deep Neural Networks (DNNs) are becoming the state-of-the-art for many tasks including reinforcement learning (RL), they are especially resistant to human scrutiny and understanding. Input attributions have been a foundational building block for DNN expalainabilty but face new challenges when applied to deep RL. We address the challenges with two novel techniques. We define a class of behaviour-level attributions *for explaining agent behaviour beyond input importance and interpret existing attribution methods on the behaviour level. We then introduce* λ-alignment*, a metric for evaluating the performance of behaviour-level attributions methods in terms of whether they are indicative of the agent actions they are meant to explain. Our experiments on Atari games suggest that perturbation-based attribution methods are significantly more suitable to deep RL than alternatives from the perspective of this metric. We argue that our methods demonstrate the minimal set of considerations for adopting general DNN explanation technology to the unique aspects of reinforcement learning and hope the outlined direction can serve as a basis for future research on understanding Deep RL using attribution.*

1. Introduction

As machine learning algorithms become ever more ubiquitous, understanding their decisions has become critical to successful learning deployments. Passengers in autonomous vehicles, for example, might find it hard to trust the vehicles' driving decisions if they do not understand how these decisions are made. As a result, an increasing number of works in machine learning have developed *attribution methods* that attempt to explain learning models' decisions. Generally, these works focus on deep neural networks (DNNs) applied to image recognition or other classification problems [28, 38]. Many attribution methods attempt to quantify the relationship between individual input features and the output, or prediction, of a DNN. For example, saliency maps compute the derivative of the prediction with respect to changes in individual inputs. The feature with the largest derivative is then taken to have the most significant influence on the model prediction [11].

More recently, some works have attempted to apply attribution methods not just to DNN-based imaging problems, but to deep RL (reinforcement learning) models that use DNNs to specify the optimal action an agent should take given the state of the environment [3, 9]. The goal of such deep RL algorithms is to maximize a received reward over time, where the reward at each time is a function of the current action taken and the environment state, and each action can influence both the immediate reward and the future state of the environment. Explaining such models is useful for applications like Atari games or autonomous driving, in which deep RL algorithms are used to determine actions taken in the game or on the road. Attribution methods can explain why the chosen action is recommended, e.g., what elements of the state influence the deep RL's choice of action. As argued by Atrey et al. [6], however, the evaluation of most saliency-based deep RL explanations relies on subjective judgements on whether the identified input features "should" be considered in determining the chosen action. We aim to address this challenge by defining a quantitative metric to assess explanations of deep RL algorithms.

In defining this metric, we further argue that directly applying DNN attribution methods to deep RL algorithms is only useful for answering questions about the specific action chosen. These methods do not explain *why this action was chosen instead of other candidate actions*, which is particularly of interest in deep RL applications. Deep RL agents often utilize randomized policies that choose an action according to a probability distribution over all possible actions, so explaining deep RL algorithms requires explaining the full behavior of the agent, not just the action that is chosen. We define the first method to consider

*Equal Contribution

behavior-level explanations of deep RL agents and an associated *alignment metric* that quantifies how well a behavior-level method explains the RL agent behavior. Our metric leverages the fact that deep RL agents must account for both the immediate and expected future reward, and we show that it allows us to see how deep RL agents learn to optimize both type of rewards.

In this paper, we first give a formal description of deep RL algorithms and introduce several existing attribution methods. We then define and evaluate our behavior-based attribution method, making the following contributions:

- A novel class of methods, **behavior-level attributions** for deep RL algorithms to explain why a given action was chosen over others, instead of focusing only on the specific action chosen.
- A **quantitative metric for explanation methods** called λ-alignment, which measures whether an attribution method is indicative of the actions that the agent chooses.
- An **evaluation of our proposed method on Atari games** that shows that it can answer a wider range of relevant explanation questions than state-of-the-art action-based deep RL explanation methods.
- We **empirically demonstrate** that our proposed explanations and metric allow us to compare behavior-level attribution methods and understand how the deep RL agent learns the optimal actions over the training period.

2. Background

In this section, we give an overview of reinforcement learning algorithms and survey existing work on attribution methods, which we will later contrast with our proposed explanation methods. Throughout the paper, we use lowercase x to denote scalar values and its bold font $\mathbf{x}$ to indicate vectors.

Consider an agent that interacts with the environment over a series of discrete timesteps. At each discrete timestep t, the agent arrives at state $\mathbf{s}_t \in \mathbb{R}^m$ and takes an action a_t according to its policy $\pi(\mathbf{s}_t)$, receiving a reward r_t. Generally policies are probabilistic, and we write $\pi(a \mid \mathbf{s})$ to denote the probability of action a at state $\mathbf{s}$ according to policy π. We use $\mathcal{S}$ and $\mathcal{A}$ to denote the state space and action space that contain all possible states and actions, respectively. The cardinality of $\mathcal{A}$, which we assume to be finite, is denoted as $|\mathcal{A}|$.

2.1. Deep Q-Network

Two common approaches that solve the RL problem are policy-based approaches that learn the policy π [27, 31] and value-based approaches, e.g. Deep Q-Network (DQN) [16, 24], that model the action value, which we define as the Q value below:

Definition 1 (Q value) *Given the state $\mathbf{s}_t$ and rewards r, for an action a_t under a policy π, the action value (Q value) $Q^\pi(a_t, \mathbf{s}_t)$ is defined as*

$$Q^\pi(a_t, \mathbf{s}_t) \stackrel{\text{def}}{=} \mathbb{E}\left[\sum_{\tau=t}^{T} \gamma^{\tau-t} r_\tau \mid a_t, \mathbf{s}_t, \pi\right] \tag{1}$$

where T is the number of timesteps before the agent reaches the termination condition and γ is the discount factor that trades off the immediate ($\tau = t$) and future ($\tau > t$) rewards.

Among multiple approaches described above, DQN is widely applied in many RL tasks [2, 21, 23, 26] due to its ability to learn complex state representations. A DQN is a network $f(\mathbf{s}_t)$ that takes a state $\mathbf{s}_t$ and outputs the Q value for each action; we use $Q = f(\mathbf{s})$ to denote the output of the DQN for all actions $a \in \mathcal{A}$. A standard DQN agent takes the action that maximizes the Q value (Eq. 2) at each step (Eq. 3) [24].

$$Q(a_t, \mathbf{s}_t) \stackrel{\text{def}}{=} \mathbb{E}\left[r_t + \gamma \max_{a_{t+1}} Q(a_{t+1}, \mathbf{s}_{t+1})\right] \tag{2}$$

$$\pi^*(a_t \mid \mathbf{s}_t) \stackrel{\text{def}}{=} \mathbb{I}\left[Q(a_t, \mathbf{s}_t) = \max_{a_t} Q(a_t, \mathbf{s}_t)\right] \tag{3}$$

We omit the notation of t in $a_t, \mathbf{s}_t$ for simplicity if not further noted.

2.2. A Unified View of Attribution

One approach for explaining the behavior of a Deep RL agent is to employ input attributions, a transparency tool mostly used to explain classification networks in supervised learning. The goal of an explanation is to answer a specific question in which a user is interested, e.g., *What is the most important feature in the input for the model's prediction?*. Leino et al. [22] propose *quantity of interest* (QoI) to incorporate the target of an explanation shown in Def. 2

Definition 2 (Quantity of Interest) *An quantity of interest is a continuous and differentiable mapping $\phi : \mathbb{R}^m \to \mathbb{R}$, where m is the dimension of the input.*

The most common choice of QoI is to index the class of interest in the output distribution, which is equivalent to indexing the action of interest from the Q value distribution in the DQN model, which we refer to as the Action QoI (Def. 3). Similarly, Puri et al. [25] discusses another QoI by adding a softmax function on top of the DQN output before indexing, which we call the Softmax Action QoI (Def. 4).

Definition 3 (Action QoI) *Given a Q-network $Q = f(\mathbf{s})$, the* action QoI *for an action of interest a is defined as* $\phi_a(Q) \stackrel{\text{def}}{=} Q(a, \mathbf{s})$.

Definition 4 (Softmax Action QoI) *Given a Q-network $Q = f(\mathbf{s})$, the* Softmax Action QoI *for an action of interest a is defined as* $\phi_\pi(Q) \stackrel{\text{def}}{=} \exp(Q(a, \mathbf{s}))/\sum_{a' \in \mathcal{A}} \exp(Q(a', \mathbf{s}))$.

We use QoI to refer to either Action QoI or Softmax Action QoI in the rest of the paper if not further noted. For deep RL agents that employ a stochastic policy, existing work also uses the state value [15] in the QoI. We summarize this existing work in a separate "Related Work" section later in the paper.

Given a QoI, an *input attribution* is a function that assigns a score for each input feature that is proportional to its contribution towards this QoI. Higher attribution scores correspond to higher contributions. The range of an attribution method is the same shape as the input state space. Naturally, a vector of state-space size is a vastly less complex object as the policy it attempts to explain. As a result, a variety of attribution methods have been proposed, each focusing on different aspects of the policy. We elaborate on this point in Section 2.3.

Throughout the paper, we will discuss the following input attribution methods. We define them using the corresponding notations in DQN.

Definition 5 (Saliency Map (SM)) *[11, 28] Given a Q-network $Q = f(\mathbf{s})$ and a QoI ϕ, the* Saliency Map *for ϕ under $f(\mathbf{s})$ is defined as*

$$g_{\text{SM}}(\mathbf{s}, \phi) \stackrel{\text{def}}{=} \frac{\partial \phi(f(\mathbf{s}))}{\partial \mathbf{s}}$$

Definition 6 (Integrated Gradient (IG)) *[30] Given a Q-network $Q = f(\mathbf{s})$, a QoI ϕ, and a user-defined baseline state $\mathbf{s}_b$, the* Integrated Gradient *of ϕ under $f(\mathbf{s})$ is defined as*

$$g_{\text{IG}}(\mathbf{s}, \phi) \stackrel{\text{def}}{=} (\mathbf{s} - \mathbf{s}_b) \circ \int_0^1 \frac{\partial \phi(f(m(\alpha)))}{\partial (m(\alpha))} d\alpha$$

where $m(\alpha) = \mathbf{s}_b + \alpha(\mathbf{s} - \mathbf{s}_b)$.

Definition 7 (Smooth Gradient (SG)) *[29] Given a Q-network $Q = f(\mathbf{s})$, a QoI ϕ, and a user-defined standard deviation σ, the* Smooth Gradient *of ϕ under $f(\mathbf{s})$ is defined as*

$$g_{\text{SG}}(\mathbf{s}, \phi) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{s}' \sim \mathcal{N}(\mathbf{s}, \sigma^2)} \frac{\partial \phi(f(\mathbf{s}'))}{\partial \mathbf{s}'}$$

where $\mathcal{N}(\mathbf{s}, \sigma^2)$ is a Gaussian distribution centered at $\mathbf{s}$ with variance σ^2.

Remark 1 *For SM and SG, we usually treat the element-wise multiplication between the input state and SG or SG, $\mathbf{s} \circ g(\mathbf{s}, \phi)$, as the attribution score instead of the raw $g(\mathbf{s}, \phi)$. The motivation to make such modification is because we consider $g(\mathbf{s}, \phi)$ only as the local influence of features, while $\mathbf{s} \circ g(\mathbf{s}, \phi)$ describes features' global contributions towards the QoI [4]. We do not multiply the input with IG again given a similar process (if choosing $\mathbf{s}_b = \mathbf{0}$) is already included in the definition.*

Definition 8 (Occlusion-N (OC-N)) *[38] Given a Q-network $Q = f(\mathbf{s})$, a QoI measurement ϕ, a user-defined baseline input feature value b, and a partition of input features $\mathcal{D}$ with , $|D| = N$ for every $D \in \mathcal{D}$, the* Occlusion-N *for $\mathcal{D}$ is defined as*

$$g^{\mathcal{D}}_{\text{OC}}(\mathbf{s}, \phi) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{D \in \mathcal{D}} \mathbf{1}_D \left[\phi(f(\mathbf{s})) - \phi(f(\mathbf{s}_{-D}))\right]$$

where $\mathbf{s}_{-D}$ is a counterfactual state by perturbing s_i with b in $\mathbf{s}$ for all $i \in D$, and $\mathbf{1}_D$ is a vector if size m equal to 1 for indices corresponding to features in D and 0 for the other indices.

OC-1 is the most common choice where we perturb each feature individually and treat the output difference as the attribution score for each feature. For the baseline value, common choices are zeros or random noise. We will use OC-1 in the experiment section and show other choices of N in the Sec. 5.

Example visualizations of these attribution methods for example policies in a game of Pac-Man are shown in Figure 1 with higher attributions indicated by the strength of the highlights.

We choose these attribution methods because: 1) these methods implementation-invariant to any DQN architecture. Additionally, IG and SG were proven to outperform SM along some desirable criteria [36, 37], though the relevance of those conclusions to DQN has not yet been explored; 2) these methods are widely available in many open-source libraries, e.g. Captum [20]; and 3) various explainable DQN works employ instantiations with minor variations. We give a more detailed discussion in Sec. 5.

2.3. Counterfactuals and Attribution

An attribution is understood by its user in terms of counterfactuals: each exposes the effect of perturbing a given state on the resulting quantity of interest, most directly answering the question *Were the game state changed this particular way, how would the Q value change?* No attribution can explain all counterfactuals except in case of linear policies so each method's motivating perturbations differ.

Most directly, attributions produced by IG add up to exactly the difference in Q values for a given state as compared to the baseline: $\sum_i g_{IG}(\mathbf{s}, \phi_a(Q))_i = Q(a, \mathbf{s}) -$

$Q(a, \mathbf{s}_b)$. Occlusion-1 instead, directly from definition, consider perturbing any single feature to a baseline value b. $g_{OC\text{-}1}(\mathbf{s}, \phi_a(Q))_i = Q(a, \mathbf{s}) - Q(a, \mathbf{s}_{-i})$ for every $i \in [m]$. Alternatively, SM approximates infinitesimal perturbations around $\mathbf{s}$:

$$\sum_i g_{SM}(\mathbf{s}, \phi_a(Q))_i \approx \lim_{\boldsymbol{\epsilon} \to \mathbf{0}} \frac{Q(a, \mathbf{s}) - Q(a, \mathbf{s} - \boldsymbol{\epsilon})}{|\boldsymbol{\epsilon}|} \quad (4)$$

While each attribution is focused on a different set of counterfactuals to explain and can thus be expected to approximate their effect well (or be exact), users interpret attributions in terms of a wider set of counterfactuals. Attribution evaluation metrics measure the fidelity of an attribution relative to a specific set of counterfactuals which typically cannot be perfectly realized by any attribution method [34]. We begin our techniques in the next section by first rephrasing attribution methods' goals in terms of reconstructing the policies being explained.

3. Actions vs. Behavior in Explanations

Attributions as discussed so far explain the importance of each feature of state $\mathbf{s}$ on a specified action a. As discussed in the prior section, their explainability power derives from their accuracy in describing the impact of counterfactuals on the score for the given action. Reasoning about a single action in general is appropriate for attributions' traditional use cases in classification but are limited in RL: they are not indicative of agent behaviour which is a function of score over all possible actions. In this section we describe a class of behaviour-level explanations built from the basic building block of attribution (which we term action-level). We then describe how to evaluate the fidelity of such explanations as indicators of actual agent behaviour.

3.1. Action-Level Explanations

Given a policy π, state $\mathbf{s}$, and action a, an action-level attribution $A_* \stackrel{\text{def}}{=} g_*(\mathbf{s}, \phi_a)$ is a per-state-feature attribution representative of the explained Q value $Q^\pi(a, \mathbf{s})$. By representative we mean that the Q value can be expressed in terms of the attribution to some degree of accuracy [37]. By rewriting Equations 4 or the respective equations for the other attribution methods, we define *action-level policy* $\Pi^A : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$, interpreted as a distribution of actions given a state, as the approximation of the actual agent policy π. For SM for example:

$$\begin{aligned} \Pi^A_{SM}(a \mid \mathbf{s}') &\stackrel{\text{def}}{=} Q^\pi(a, \mathbf{s}) + (\mathbf{s}' - \mathbf{s}) \cdot g_{SM}(\mathbf{s}, \phi(Q^\pi)) \quad (5) \\ &\approx Q^\pi(a, \mathbf{s}') \quad \text{for } \mathbf{s}' \text{ near } \mathbf{s} \end{aligned}$$

We say that an *attribution g explains a policy* Π^A when the later is derived from the first as per Equation 5.

Policies Π^A are accurate approximations for π when $\arg\max_a \Pi^A(a \mid \mathbf{s}) = \arg\max_a \pi(a \mid \mathbf{s})$. Naturally, this cannot be expected for all states $\mathbf{s}$ except for the counterfactual states motivating each attribution method. In the above example for SG and SM, accuracy is only expected near the neighbourhood of the state $\mathbf{s}$ for which the attribution was computed.

Existing attribution evaluation metrics can be viewed in terms of properties of the policy they explain such as those based on proportionality of $\Pi^A(\mathbf{s})$ with $Q(a, \mathbf{s})$ [4, 9, 34, 37] and others that also consider its "sanity" [1] or the robustness under (adversarial) noise [14, 36, 37].

While these evaluation metrics are general enough to be applied to any DNN with differentiable outputs, they were not specifically developed for DQNs and therefore most existing work for explaining DQN agents lacks quantitative metrics for the fidelity of an attribution of RL agent policies [6]. Our design for such RL-focused metrics begins with introduction of *behaviour-level attributions* and explained *behaviour-level policies* analogously to the action-level variants of this section.

3.2. Behavior-Level Explanations

Some questions can be answered using action-level attributions, e.g.: *what are the most important features that drive the agent to take the current action?* On the other hand, an action-level attribution cannot answer the question: *What will the agent do given the current state?* In this section, we propose a new form of attribution which we call *behavior-level attribution* aiming to resolve such questions. Importantly our goal is to keep such attributions as simple as action-level attributions so as to be presentable alongside or overlayed on top of state or game board visualizations.

Definition 9 (Behaviour-Level Attribution (Method)) *Given an action space $\mathcal{A}$, state space $\mathcal{S}$ a* behaviour-level attribution *method is a function from $\mathcal{S}$ to $\mathcal{A}^m$. That is, given a state, the attribution produces an action for each state feature (in the same way an action-level attribution produces a real-numbered score for each state feature).*

The definition itself does not impose any restrictions on behaviour-level attributions but the metrics which we will define over them will presume a goal: they should indicate the association between an input feature with a specific action if an agent "sees" that feature individually. For example, given a Pac-Man agent, when considering each feature individually, the food on the agent's north side likely attracts the agent to `go_north` while a ghost at the same position would be expected to have the association with the move in the opposite direction.

Behaviour from Action A behaviour-level attribution can constructed from any action-level attribution in the expected manner. Give an action-level attribution g_*, we de-

fine

$$G_*(\mathbf{s})_i \stackrel{\text{def}}{=} \arg\max_a g_*(\mathbf{s}, \phi(Q^\pi))_i \tag{6}$$

Figure 2 demonstrates the behaviour-level attributions derived from several action-level attributions. The visualization in the figure designates the Pac-Man move attributed to each state feature by directional arrows and colors. The restriction of visualization to the area around Pac-Man designates the locality parameter λ of the local reconstruction technique we introduce shortly.

Explained Behaviour-level Policies To approach our evaluation metric, we first rephrase behaviour-level attributions in terms of the policy $\Pi^B : \mathcal{S} \to \mathcal{A}$ that can be recovered from them. Unlike the action-level case, explained policies assign the action taken for each state. We also parametirize the reconstruction of a behaviour-level policy by the sets of state features. Specifically, we will experiment with actions recovered from state features in a given neighbourhood of the player. Evaluating explained policies derived from various neighbourhood will let us study the spatial "perceptual horizon" of an agent. In the case of Pac-Man, for example, we will be able to quantify how big of the area around Pac-Man is necessary to reconstruct their behaviour.

Definition 10 (λ-Local Behaviour Action Reconstruction) *(or λ-BAR) Given a behaviour-level attribution $\mathbf{c} : \mathcal{A}^m$ for some state and an agent with coordinates, $\mathbf{p}$, and each input feature's coordinate $\mathbf{q}_i$, the* λ-Local Behaviour-level Action Reconstruction *ψ_λ is defined as:*

$$\psi_\lambda(\mathbf{c}) \stackrel{\text{def}}{=} \arg\max_a |\{i : i \in [m], r(\mathbf{p}, \mathbf{q}_i) \leq \lambda, c_i = a\}| \tag{7}$$

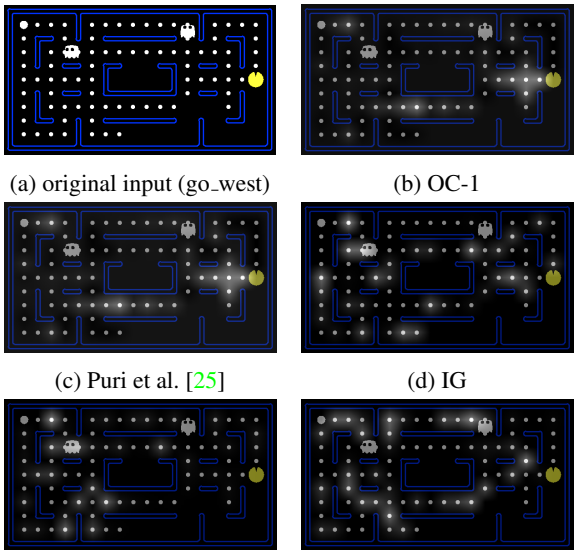
where $r(\cdot, \cdot)$ is an user-defined function measuring the distance between a feature and the agent.

The distance function r is used to build a reachable neighborhood and should be based on the actual environment. For example, ℓ_1 distance is a considered as reasonable for Pac-Man since a valid move of the agent is either horizontal or vertical. In the experiment section we will sweep over a broad range of λ and give several examples of r for different Atari games.

Evaluation. Finally we can construct an explained behaviour-level policy using a local reconstruction.

Definition 11 (λ-local Behaviour-level Policy) *For a given λ and behaviour-level attribution method G we define the lambda-*local Behaviour-level Policyy *Π^B_λ as:*

$$\Pi^B_\lambda(\mathbf{s}) \stackrel{\text{def}}{=} \psi_\lambda(G(\mathbf{s})) \tag{8}$$



(a) original input (go_west) (b) OC-1 (c) Puri et al. [25] (d) IG (e) SG (f) SM

Figure 1. Visualization of Action-Level Attributions. Important features are highlighted by attribution scores toward `go_west` action using different attribution methods. Brightness is proportional to the attribution score.

Whereas the goal of evaluating Π^A is to select an optimal attribution method g so that, for example, $\Pi^A(\mathbf{s})$ is proportional to $Q_\pi(a, \mathbf{s})$, the goal of evaluation for Π^B is to select an optimal attribution method G such at $\Pi^B(\mathbf{s})$ is equal to the the actual action $a^* = \arg\max_a Q_\pi(a, \mathbf{s})$ taken by the agent at that state. We end this section by introducing λ-alignment as the metric we propose for this task.

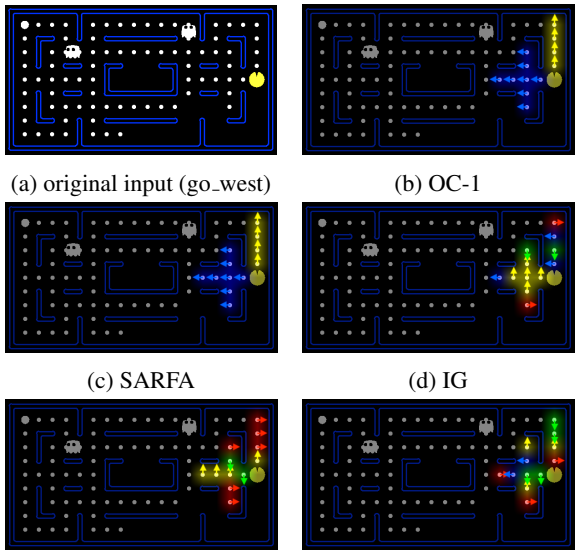
Definition 12 (λ-Alignment) *Given a Q-network $Q(\mathbf{a}, \mathbf{s}) = f(\mathbf{s})$ and a behavior-level policy $\Pi^B_\lambda(\mathbf{s})$ explained by behaviour-level attribution G, we define* λ-Alignment *$L_\lambda(G)$ as*

$$L_\lambda(G) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{s}\sim\mathcal{S}}\mathbb{I}\left[\arg\max_a Q(a, \mathbf{s}) = \Pi^B_\lambda(\mathbf{s})\right] \tag{9}$$

Alignment as defined here is inspired by Annasamy and Sycara [5], who use the same conceptual metric over actions recovered from attention-based DQN models. The set of states in the definition of alignment would ideally be all states or representative states according to a criteria of interest. In our experiments, we approximate the over a significant number of matches.

4. Experiments

We apply the proposed behavior explanations on Pac-Man and evaluate how different attribution methods perform in terms of the λ-alignment. We also include other Atari Games in the supplementary materials. We use the Pac-Man environment produced by van der Ouderaa [32].



(a) original input (go_west) (b) OC-1

(c) SARFA (d) IG

(e) SG (f) SM

Figure 2. Visualization of $\rho(\mathbf{s})$ using different attribution methods. We use colors and arrows to indicate the action each feature mostly attributes towards. **blue**: `go_west`, **green**: `go_south`, **red**: `go_east`, and **yellow**: `go_north`. We show features within $\lambda \leq 4$ by setting r as ℓ_1 distance. This figure is better visualized on screen.

We give a brief introduction to the rule of Pac-Man game first.

Pac-Man World. The goal of the agent is to consume the `food` (small white dots in Fig. 1) in the environment. The game is terminated either when the agent consumes all food (marked as "win") or encounters a `ghost` (marked as "lose"). There is a special feature called `capsule` in the corners of the environment (big white dots in Fig. 1). If the agent consumes a capsule, a `ghost` feature will become a `scared_ghost` such that the agent will receive high rewards if encountering it instead of "lose".

The Q-network used to train the agent contains 4 hidden layers, and its input is a 6-channel binary matrix indicating the location of the wall, the agent, `ghost`,`scared_ghost`, `food` and `capsule`. The DQN is trained with 12000 matches and the agent has a winning rate of 90% when evaluated over 100 matches. We include information on other hyper-parameters in the Supplementary Material A.

Visualization. We first demonstrate how a behavior-level explanation differs from an action-level explanation in the visualization. We use the input attribution methods described in Def. 5-8 for the action and behavior-level explanations. We also include a recent proposed method SARFA [25], which has been only tested on deep RL. Since the aim of SARFA still focuses on discovering the feature importance towards a specific action, we still consider it as an action-level explanation. We discuss the detail of hyper-parameters in each attribution methods in Supplementary Material B.

An example visualization of using the Softmax Action QoI with the methods mentioned above for the Pac-Man agent is shown in Fig. 1. In the visualization, we find that SM, IG and SG highlight features very far away from the agent and assign those features relatively high scores (brightness is proportional to the attribution scores) compared to nearby ones. These results are counter-intuitive since foods very far away from the agent will have a high discount (γ^t decreases exponentially), so the agent can be expected to weight local rewards more than food far away. On the other hand, SARFA produces almost the same results as OC-1 with Softmax Action QoI. We provide further discussion on this result in Supplementary Material C. We then visualize $\rho(\mathbf{s})$ in Fig. 2, as part of the behavior-level explanation, for the same state used in Fig. 1. Each color in the visualization represents an action to which this feature attributes most significantly. In this example we focus on the neighborhood around the agent by setting $\lambda \leq 4$, using the ℓ_1 distance. We justify the choice of ℓ_1 in the following quantitative experiments. OC-1 and SARFA again show the same results by identifying that all the food on the west attract the agent to `go_west` (similarly for "north"), while SM, IG, and SG associate each food with an action under which the agent cannot consume the food.

Best Attribution for Π^B. In this experiment, we evaluate λ-alignment with Pac-Man, and we include evaluations on Breakout and Space Invader in the Supplementary Material C. For the distance function r in Pac-Man, we use $r(\mathbf{p}, \mathbf{q}) = ||\mathbf{p} - \mathbf{q}||_1$ since the Pac-Man agent can only move horizontally or vertically. We randomly sample 7419 states from 30 matches to approximate the expectation in Def. 12. We plot the average alignment score over all sampled states against $\lambda \in [0, 20]$ in Fig. 3, for all input attributions. We separate four major features in the Pac-Man game, `food`, `ghost`, `scared_ghost` and `capsule` where the agent needs to avoid `ghost` but it can consume the others. There are many more `food` than other features at the beginning of each game. We make the following conclusions from Fig. 3:

- λ-alignment may increase at first but will eventually decrease as λ grows. This matches the motivation of "Local Feature First:" using all features instead of local features may not allow us to reconstruct the optimal action.
- Not all methods are better than a random guess (which has 25% accuracy for the Pac-Man game) under some choices of λ, e.g. SG on the food features.

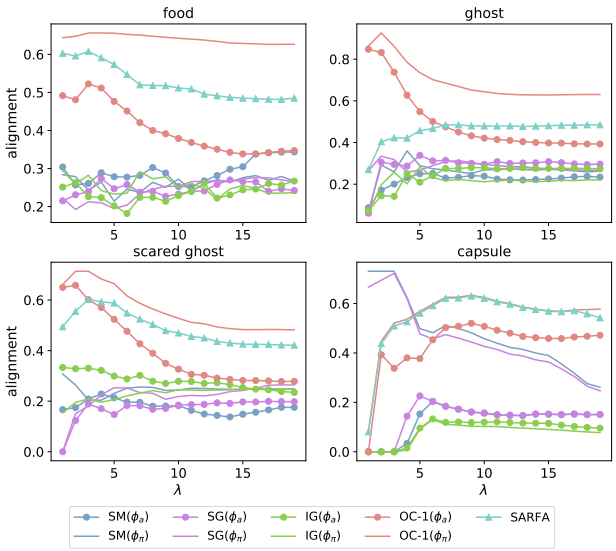


Figure 3. λ-alignment for four kinds of features in Pac-Man game using different attribution methods combined with Action QoI ϕ_a and Softmax Action QoI ϕ_π. The distance metric r is ℓ_1. Results are averaged over 7419 states, respectively.

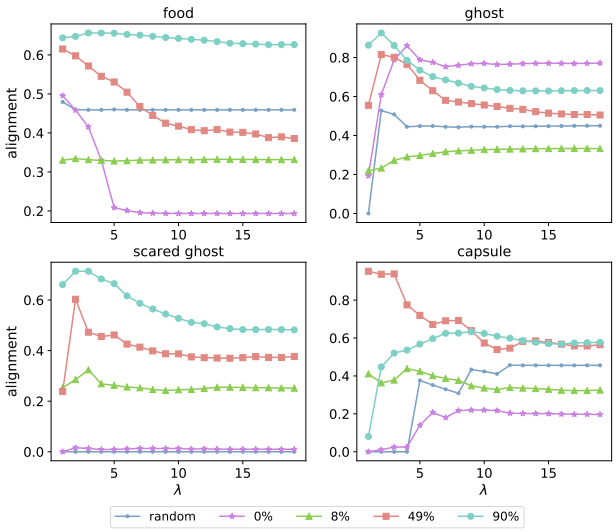


Figure 4. λ-alignment of four DQNs at different training stages for four kinds of features in Pac-Man game using OC-1 with Softmax Action QoI.

- OC-1 with any QoI outperforms other methods for most of the choices of λ on `food`, `ghost`, `scared_ghost`. Given `food` is the most important feature in Pac-Man, we find OC-1 with Softmax Action QoI ϕ_π consistently has the highest alignment with the optimal action under all λ.

- The λ that maximizes the alignment (peaks on curves) corresponds to a neighborhood from which the agent is most likely to make actions based on those features. In other words, the corresponding λs of peaks disclose how far away the agent can "see" into the future. For example, we know that the agent will dodge `ghost`. In the `ghost` plot in Fig. 3, Oc-1(ϕ_π) shows that the agent learns to take action to dodge `ghost` when it is about 2 steps away from it (the peak happens approximately at $\lambda = 2$).

From the above evaluation we conclude that **OC-1 with softmax action QoI best aligns with the optimal action over other methods**, while SM, IG and SG perform worse than the perturbation-based methods, OC-1 and SARFA, over all λs. SM tends to highlight noise in the attribution map, which has also been mentioned in related work on action-level explanations [19]. Since the action-level explanation is the building block for the behavior-level explanation, this may explain why SM does not have high alignment. On the other hand, IG (and SG) may suffer from the fact that the interpolations between the baseline and the input state (or the Gaussian noise around the input) are not semantically meaningful: these states may not exist in $\mathcal{S}$. On the other hand, we currently use zeros as the baseline input for IG, which is also not semantically meaningful (in contrast, zeros in image classification indicate a black image). Looking for a meaningful baseline input can be a significant challenge in explaining reinforcement learning.

Taking the advantage of OC-1 with Softmax Action QoI, we show the following use case of exploring a DQN model leveraging behavior-level explanation.

How DQN Plays Pac-Man. We demonstrate how to leverage behavior-level explanations to monitor the learning process of the Pac-Man agent. We trained 4 DQN models to play Pac-Man games with different training epochs, which are then tested to have success rates of 0%, 8%, 49% and 90% over 100 matches. We also add a baseline model where the weights are randomly initialized without preforming any training. We use OC-1 with Softmax Action QoI to produce the behavior-level explanation, since performs best in the previous experiments. The results are shown in Fig. 4, from which we make the observations:

- Compared with other agents, at the beginning of the training when the success rate is 0%, the agent learns to align its action with the position of `ghost` instead of paying a lot of attention to `food`, given that the alignment score on the `ghost` feature is close to other well-trained models and the curve is above `random`. In other words, the Pac-Man agent learns to survive first instead of win first.

- An agent with 49% success rate tends to win the game by eating the `capsule` so that it can consume `scared_ghost`, given the alignment curves in the

`capsule` plot are much higher than the rest of the agents. However, the reason why it only has 49% success rate is probably that it only focuses on the closest `food`, so the alignment curve decrease much faster than the agent with 90% success rate in the `food` plot.

- The alignment curves on `scared_ghost` shift up when the success rate increases from 0% to 90%, showing that the agent gradually leans to consume `scared_ghost` to earn higher rewards.

5. Related Work

Attribution in Deep RL. We discuss several related work that have not been well-discussed in previous sections or the supplementary materials and how they are related to different QoI and input attribution methods. One line of work is Jacobian method [35], which is equivalent to combining SM and Action QoI to identify important features. We then discuss two methods that both use Action QoI and OC-N but with different treatment to identify the baseline b or the subset $\mathcal{D}$. Instead of perturbing the input with zeros, another approach is to perturb the input features with Gaussian blurs [15], which is equivalent to choosing the blurred pixels as a baseline b and setting N equal to the number of blurred pixels in OC-N. Template matching [10] has also been used to identify a semantically meaningful subset of input features [19] , which is equivalent to setting N equal to the number of pixels in the template in OC-N. Therefore, we consider Gaussian blurs and the template matching as two prepossessing techniques in applying OC-N to DQN models with image input instead of two different attribution methods.

Action Reconstruction. The concept of behavior-level explanation is largely motivated by the *agreement* metric introduced by Annasamy and Sycara [5] that reconstructs an action from the key-value in the proposed attention layer [7]. Besides, It has shown with counterfactual analysis that reconstructing actions from an attribution map by human's judgement may not actually correspond to the agent's behavior [6], which also motivates the work of proposing concrete reconstruction rule, λ-BAR.

Attention. On the other hand, using attentions belongs to another line of work which aims to build networks inherently explainable, which has been discussed both in general tasks [12, 13, 39] and deep RL [8, 17, 33].

6. Conclusions

In this work, we first categorize existing attribution methods for explaining deep RL agents into (i) action-level explanations, which aim to identify each feature's importance score towards a quantity of interest; and (ii) behavior-level explanations, which aim to reconstruct the agent's prediction through well-defined and explainable functions. We demonstrate how to construct behavior-level explanations from action-level methods for the Pac-Man game in the main paper and for other games in our supplementary materials. Our empirical results show that OC-1 with Softmax QoI outperforms other methods in the behavior-level explanation. In practice, we show that by leveraging behavior-level explanations, the human user is able to monitor the learning process of a deep RL agent and produce more insightful explanations than are possible with action-level explanation methods.

Ethical Impact

Our work enables (human) users or observers of deep reinforcement learning (RL) algorithms to better understand an RL agent's decisions, by identifying the state features that contribute to the agent behavior and quantifying the degree to which these explanations match the true RL agent behavior. These explanations can help increase users' trust in RL decisions, and our metrics for the explanation quality can further inform users of how seriously they should treat an explanation method's findings.

Conversely, the results of our methods, when applied to specific realizations of deep RL agents, can be used to scrutinize the ethics of these agents' behavior, by revealing the underlying reasons behind agent decisions and the degree to which these reasons are accurate. For example, if we attribute a decision to an element of the state that should not be considered, such as a robot utilizing a user's race when using RL to decide how to interact with this user, we may conclude that this RL agent is behaving unethically. Our quantitative metrics for the quality of our explanations can further inform such ethical debates, by informing the degree of trust that we should place in an explanation method's findings.

References

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems 31*, pages 9505–9515. 2018). 4

[2] Smruti Amarjyoti. Deep reinforcement learning for robotic manipulation-the state of the art, 2017. 2

[3] Marco Ancona, Enea Ceolini, Cengiz ztireli, and Markus Gross. A unified view of gradient-based attribution methods for deep neural networks, 2017. 1

[4] Marco Ancona, Enea Ceolini, Cengiz ztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018). 3, 4

[5] Raghuram Mandyam Annasamy and K. Sycara. Towards better interpretability in deep q-networks. In *AAAI*, 2019). 5, 8

[6] A. Atrey, K. Clary, and David Jensen. Exploratory not ex-

planatory: Counterfactual analysis of saliency maps for deep reinforcement learning. In *ICLR*, 2020). 1, 4, 8

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015). 8

[8] Ritwik Bera, Vinicius G. Goecks, G. Gremillion, John Valasek, and Nicholas R. Waytowich. Podnet: A neural network for discovery of plannable options. *ArXiv*, abs/1911.00171, 2020). 8

[9] Umang Bhatt, Adrian Weller, and José M. F. Moura. Evaluating and aggregating feature-based model explanations. *ArXiv*, abs/2005.00631, 2020). 1, 4

[10] Roberto Brunelli. *Template Matching Techniques in Computer Vision: Theory and Practice*. Wiley Publishing, 2009). 8

[11] N. Courty and E. Marchand. Visual perception based on salient features. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, 2003). 1, 3

[12] Yinpeng Dong, Hang Su, Jun Zhu, and Bo Zhang. Improving interpretability of deep neural networks with semantic information, 2017. 8

[13] X. Gao, Tingting Mu, J. Y. Goulermas, Jeyarajan Thiyagalingam, and Meng Wang. An interpretable deep architecture for similarity learning built upon hierarchical concepts. *IEEE Transactions on Image Processing*, 29:3911–3926, 2020). 8

[14] A. Ghorbani, Abubakar Abid, and James Y. Zou. Interpretation of neural networks is fragile. In *AAAI*, 2019). 4

[15] S. Greydanus, Anurag Koul, J. Dodge, and A. Fern. Visualizing and understanding atari agents. *ArXiv*, abs/1711.00138, 2018). 3, 8, 11

[16] H. V. Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *AAAI*, 2016). 2

[17] Aadil Hayat, Utsav Singh, and Vinay P. Namboodiri. Inforl: Interpretable reinforcement learning using information maximization, 2019. 8

[18] Itseez. Open source computer vision library. https://github.com/itseez/opencv, 2015. 11

[19] Rahul Iyer, Y. Li, H. Li, Michael Lewis, R. Sundar, and K. Sycara. Transparency and explanation in deep reinforcement learning neural networks. *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 2018). 7, 8, 11

[20] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Jonathan Reynolds, Alexander Melnikov, Natalia Lunova, and Orion Reblitz-Richardson. Pytorch captum. https://github.com/pytorch/captum, 2019. 3

[21] Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. *ArXiv*, 2017). 2

[22] Klas Leino, Linyi Li, Shayak Sen, Anupam Datta, and Matt Fredrikson. Influence-directed explanations for deep convolutional networks. *2018 IEEE International Test Conference (ITC)*, 2018). 2

[23] Gabriel Maicas, G. Carneiro, A. Bradley, J. Nascimento, and I. Reid. Deep reinforcement learning for active breast lesion detection from dce-mri. In *MICCAI*, 2017). 2

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013. 2

[25] Nikaash Puri, Sukriti Verma, Piyush Gupta, Dhruv Kayastha, Shripad Deshmukh, Balaji Krishnamurthy, and S. Singh. Explain your move: Understanding agent actions using specific and relevant feature attribution. *arXiv: Computer Vision and Pattern Recognition*, 2020). 2, 5, 6

[26] D. Silver, Aja Huang, Chris J. Maddison, A. Guez, L. Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, S. Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016). 2

[27] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning - Volume 32*, 2014). 2

[28] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2013. 1, 3

[29] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Vigas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise, 2017. 3

[30] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *Proceedings of the 34th International Conference on Machine Learning*, 2017). 3

[31] Richard S Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*. MIT Press, 2000). 2

[32] Tycho van der Ouderaa. Deep reinforcement learning in pacman. 2016). 5, 10

[33] Xinzhi Wang, H. Li, Rui Liu, H. Zhang, Michael Lewis, and K. Sycara. Explanation of reinforcement learning model in dynamic multi-agent system. *ArXiv*, abs/2008.01508, 2020). 8

[34] Zifan Wang, Piotr Mardziel, Anupam Datta, and Matt Fredrikson. Interpreting interpretations: Organizing attribution methods by criteria. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020). 4

[35] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning, 2015. 8

[36] Zifan Wang, Haofan Wang, Shakul Ramkumar, Matt Fredrikson, Piotr Mardziel, and Anupam Datta. Smoothed geometry for robust attribution, 2020. 3, 4

[37] Chih-Kuan Yeh, Cheng-Yu Hsieh, Arun Sai Suggala, David I. Inouye, and Pradeep D. Ravikumar. On the (in)fidelity and sensitivity of explanations. In *NeurIPS*, 2019). 3, 4

[38] Matthew D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014). 1, 3

[39] Q. Zhang, Y. Wu, and S. Zhu. Interpretable convolutional neural networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, 2018). 8

Supplementary Materials

Supplementary Material A

For experiments of Pac-Man, we use environment produced by van der Ouderaa [32] [1], which is adopted from the project created by UC Berkeley [2]. This PacMan implementation has native support for different maps or levels. Our paper utilize the medium-map (Figure 5) which is a game grid of 20 $\times$ 11 tiles. The original version of PacMan has a size of 27 $\times$ 28 game grid.

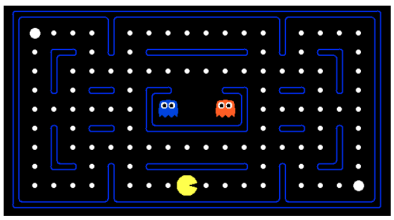


Figure 5. The layout of medium grid

State Representation Each game frame consists of a grid or matrix containing all 6 features. In each matrix a 0 or 1 respectively express the existence or absence of the element on its corresponding matrix. As a consequence, each frame contains the locations of all game-elements represented in a $W \times H \times 6$ tensor, where W and H are the respective width and height of the game grid. Conclusively a state is represented by a tuple of two of these tensors together representing the last two frames, resulting in an input dimension of $W \times H \times 6 \times 2$.

Network architecture The Q-network and the target network consist of two convolutional layers followed by two fully connected layers. The parameters of each layer is shown in Table 1.

Training Parameters For the professional agent and the one we use to compare with other attribution methods, we set $\gamma = 0.95$. The replay memory we use has a maximum memory of 10000 experience tuples to limit the memory usage. To ensure this replay memory is filled before training, the first Q-function update occurs after the first 5000 iterations. Every training iteration a mini-batch, consisting of 32 experience tuples, is sampled.

Attribution Methods For Integrated Gradient, we use the all zero matrix as the baseline and set the step to be 50. For Smooth Gradient, the noise we add to the original input follows a Gaussian distribution centered at $\mathbf{s}$ with variance $\sigma^2 = 0.15|\max(\mathbf{s}) - \min(\mathbf{s})|$.

[1] https://github.com/tychovdo/PacmanDQN

[2] http://ai.berkeley.edu/project_overview.html

Supplementary Material B

We compare and contrast OC-1 with Softmax QoI with SARFA. We first write SARFA with the following definition.

Definition 13 *Given a Q-network $Q = f(\mathbf{s})$, a user-defined baseline input feature value b and an action of interest, SARFA for each feature $g_{SA}(\mathbf{s}))i$ is defined as*

$$g_{SA}(\mathbf{s})_i = \frac{2K\Delta p}{K + \Delta p}$$
$$\Delta p = \phi^a_\pi(f(\mathbf{s})) - \phi^a_\pi(f(\mathbf{s}_{-i}))$$
$$K = 1/[1 + D_{KL}(\sigma((f(\mathbf{s})), \sigma((f(\mathbf{s}_{-i}))]$$

where σ is the a softmax function but excludes the action of interest a of in $f(\mathbf{s})$ output, D_{KL} denotes the KL divergence and ϕ^a_π is the softmax QoI with the action a.

Recall the definition of OC-N in Def. 8 when N=1, $\Delta p = g_{OC}(\mathbf{s})_i$ in Def. 13. SARFA introduces a new variable K to measure the difference between the distribution of post-softmax Q values of other actions excluding the action of interests. As KL divergence goes from 0 to the infinity, K goes from 0 to 1; Δp is the difference between the softmax scores, so Δp goes from -1 to 1. Therefore, if $\Delta p < 0$, we have $g_{OC}(\mathbf{s})_i < g_{SA}(\mathbf{s})_i$, which means SARFA tends to assign higher attribution scores than OC-1 if OC-1 finds the feature with negative attribution towards the action of interest. For $\Delta p > 0$, $g_{OC}(\mathbf{s})_i = g_{SA}(\mathbf{s})_i$ can be viewed as the Harmonic Mean $(2/[1/a + 1/b])$ between K and Δp. Compared to the Algorithm Mean $([a + b]/2)$ where the result is closer to $\max(a, b)$, the result of Harmonic Mean is closer to $\min(a, b)$. Therefore, $g_{SA}(\mathbf{s})_i$ will be more closer to $\min(K, g_{OC}(\mathbf{s})_i)$, which is the motivation of SARFA to penalize the attribution score $g_{OC}(\mathbf{s})_i$ if the perturbation $\mathbf{s}_{-i}$ also causes big change to all other actions. It seems reasonable for the action-level attribution that only highlights features that exclusively important to the action of interest and excluding features that are both important to the selected action and other actions. However, when it comes the the behavior-level attribution, the penalty of K seems to be redundant since $G_*(\mathbf{s})$ will go over all actions in $\mathcal{A}$ to calculate and compare the attribution scores for each action. When the effect of K becomes dominant in $g_{SA}(\mathbf{s})_i (K << \Delta p)$, $g_{SA}(\mathbf{s})_i$ starts to find whether a feature will exclusively attribute to only one action. It could be one of the reason why empirically SARFA produces lower alignment curves than OC-1 with Softmax Action QoI.

Type	Kernels	Kernel size	Stride	Outputs
Convolutional	16	3×3	1	$W - 2 \times H - 2 \times 16$
Convolutional	16	3×3	1	$W - 2 \times H - 2 \times 16$
Fully-connected	256	/	/	256
Fully-connected	4	/	/	4

Table 1. Network Architecture Parameters

Supplementary Material C

We repeat the same evaluation on Breakout and Space Invaders provided by Greydanus et al. [15]. Unlike the Pac-Man game which takes binary matrices as inputs, DQNs for Breakout and Space Invaders take image input. Therefore, our treatment for these games are different from Pac-Man. Instead of performing OC-1 on each pixel, we use the object identification [19] to locate a subset of features which are semantically meaningful to the agent.

Breakout. Instead of select each feature and perform OC-1 or SARFA, we identify the whole `brick` in Breakout and perform OC-N (N equals the number of pixels in a `brick`). As features in Breakout are rectangular, we therefore use a sliding window with fixed size to select each `brick` and perform OC-N or SARFA with two baselines, blurring the whole `brick` with Gaussian kernel [15] and replacing all pixels with zeros. We further define the distance function r between a brick and the agent as one plus the amount of bricks between the brick of interest and the agent, given the ball will get reflected once it hits a brick.

Space Invader. Given the feature shape is much more complicated in Space Invader, we perform object identification by template matching with OpenCV [18] first and use $r(\mathbf{p}, \mathbf{q}) = ||\mathbf{p} - \mathbf{q}||_2$ given that the possibility of being attacked increases with the decrease of l_2 distance between the agent and invaders. We use the Gaussian blur and zeros as two baselines for OC-N and SARFA as well.

Experiments. We evaluate and select which action-level attribution method is the best choice to build the behavior-level explanation for Breakout and Space Invader. Given the poor performance of the gradient-based methods, we only demonstrate how perturbation-based methods, OC-N and SARFA perform. For OC-N, use two baselines, Gaussian blurs and zeros, as mentioned above and two QoI, Action QoI and Softmax Action QoI. For SARFA, also explore how two baselines will influence its performance. We play each game for 20 times and each time we sample 400 frames, 8000 states in total, respectively, to approximate the expectation in the λ-alignment. The results are shown in Fig. 6. We find even though OC-N with Softmax QoI is still the best attribution across all possible λ, the alignment scores are less than 50% even though Breakout and

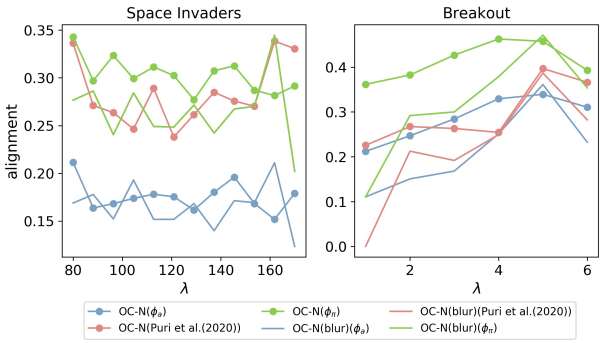
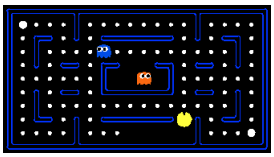


Figure 6. λ-alignment curves comparing approaches on Atari game dataset.For Space Invaders, OC-1 assigns high alignment to smaller λ while SARFA also give bigger λ a high alignment, which is contradictory to our intuition since closer the invaders, more dangerous it is. For Breakout, it seems that all of the attribution methods assign higher alignment to relatively bigger λ, indicating that a well-trained agent focuses more on the thicker part of the bricks and aims to break it out.

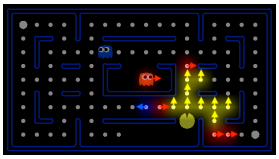
Space Invader have smaller action space and type of different features compared to Pac-Man. For Breakout, the highest alignment for OC-N with Softmax QoI occurs at around $\lambda = 4$, which indicates that the agent is likely to consider 3 more `bricks` it may hit with the current action. For Space Invader, the highest aliment occurs at the very beginning when $\lambda = 80$, which is the closest distance between an agent and a `monster`; therefore, we consider the agent treats the closest enemy most significantly than all other enemies.

Supplementary Material D

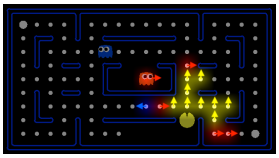
More visualizations of the behavior-level explanations on Pac-Man game (Fig. 7 to 12). We find SARFA and OC-1 with Softmax Action QoI tend to produce similar results when there are many `food` in the environment (Fig. 7 to 9). But when the game is close the termination (Fig. 10 to 12), where `food` becomes less, SARFA starts to assign different directions to nearby features other than the actual action. One possible reason is that when food becomes less, K in SARFA becomes dominate. SARFA tends to indicate the importance of this feature towards all other actions when it is perturbed instead of the importance towards the action of interest. But why and how the number of food influences the behavior of K remain unknown to us. We consider these results as discovering one of the limitations of SARFA.



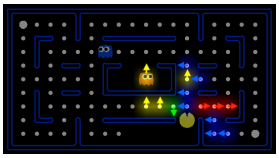
(a) original input (go_north)



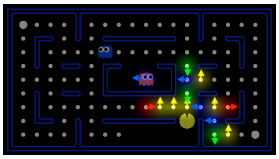
(b) OC-1



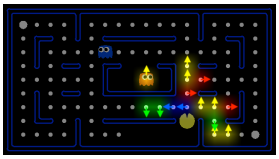
(c) SARFA



(d) IG

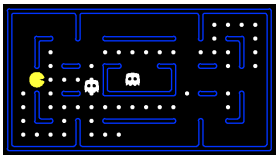


(e) SG

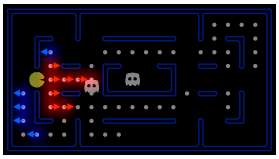


(f) SM

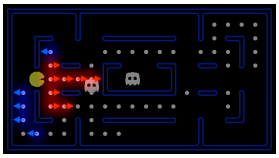
Figure 7.



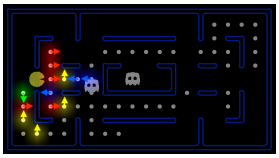
(a) original input (go_east)



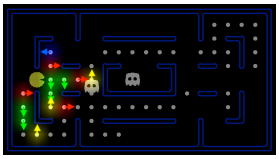
(b) OC-1



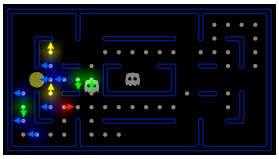
(c) SARFA



(d) IG

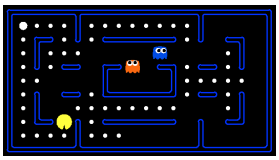


(e) SG

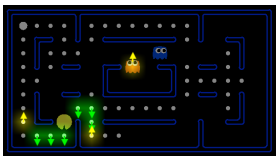


(f) SM

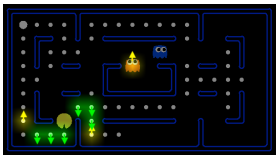
Figure 9.



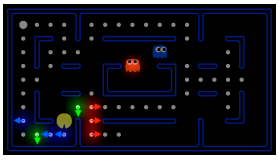
(a) original input (go_south)



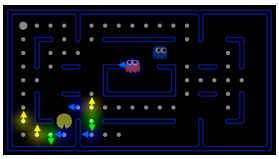
(b) OC-1



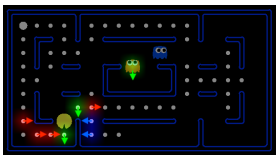
(c) SARFA



(d) IG

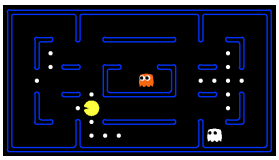


(e) SG

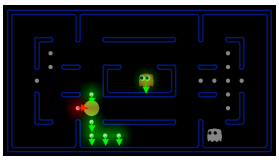


(f) SM

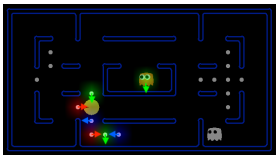
Figure 8.



(a) original input (go_south)

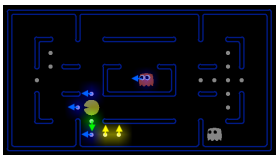


(b) OC-1

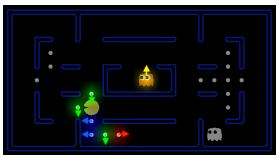


(c) SARFA

(d) IG

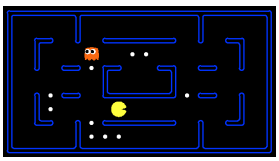


(e) SG

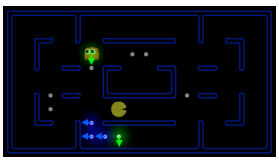


(f) SM

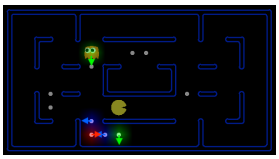
Figure 10.



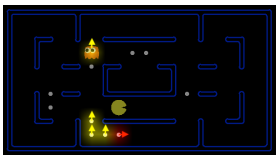
(a) original input (go_west)



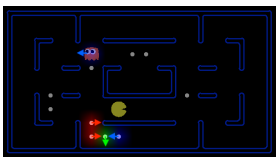
(b) OC-1



(c) SARFA



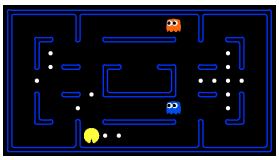
(d) IG



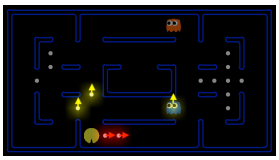
(e) SG

(f) SM

Figure 11.



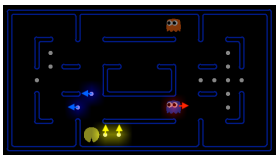
(a) original input (go_east)



(b) OC-1

(c) SARFA

(d) IG



(e) SG

(f) SM

Figure 12.