Outcome-Based Semifactuals for Reinforcement Learning

Yanzhe Bekkemoen YANZHE.BEKKEMOEN@NTNU.NO
Helge Langseth HELGE.LANGSETH@NTNU.NO
Norwegian University of Science and Technology

Editors: Hung-yi Lee and Tongliang Liu

Abstract

Counterfactual explanations in reinforcement learning (RL) aim to answer what-if questions by demonstrating sparse and minimal changes to states, resulting in the probability mass moving from one action to another. Although these explanations are effective in classification tasks that look for the presence of concepts, RL brings new challenges that counterfactual methods need to solve. These challenges include defining state similarity, avoiding out-of-distribution states, and improving discriminative power of explanations. Given a state of interest called the query state, we solve these problems by asking how long the agent can execute the query state action without incurring a negative outcome regarding the expected return. We coin this outcome-based semifactual (OSF) explanation and find the OSF state by simulating trajectories from the query state. The last state in a subtrajectory where we can take the same action as in the query state without incurring a negative outcome is the OSF state. This state is discriminative, plausible, and similar to the query state. It abstracts away unimportant action switching with little explanatory value and shows the boundary between positive and negative outcomes. Qualitatively, we show that our method explains when an agent must switch actions. As a result, it is easier to understand the agent's behavior. Quantitatively, we demonstrate that our method can increase policy performance and, at the same time, reduce how often the agent switches its action across six environments. The code and trained models are made open source[1].

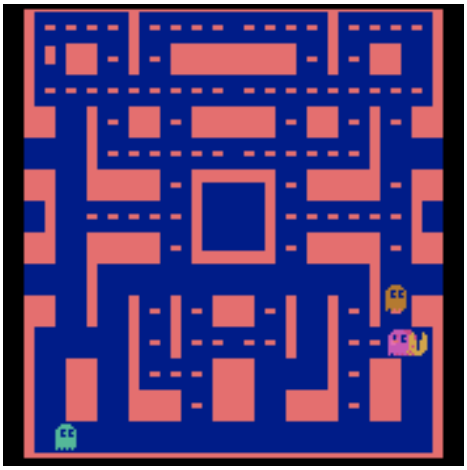
Keywords: Explainable Reinforcement Learning; Interpretability; Policy Explanation

1. Introduction

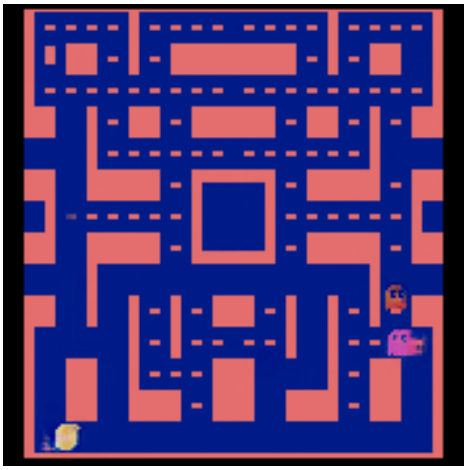
Reinforcement learning (RL) has shown incredible performance in several domains. These include surpassing human performance in various games like Go, chess, poker, Atari (Mnih et al., 2013; Schrittwieser et al., 2020; Silver et al., 2016; Brown and Sandholm, 2017), robotics (Tang et al., 2024), and healthcare (Yu et al., 2023). Much of these accomplishments are due to neural networks being flexible function approximators. However, the flexibility is obtained by sacrificing other desirable properties, such as explainability (Arrieta et al., 2020). Explainability is crucial since it helps stakeholders, from the end users to researchers, understand and trust RL systems. By understanding the systems, they can be improved, corrected, and safely deployed. Furthermore, they can be used in high-stake domains such as healthcare, criminal justice, and finance (Yang et al., 2023).

Explainability in RL has become increasingly popular recently and resulted in the research field known as explainable reinforcement learning (XRL) (Qing et al., 2023; Glanois et al., 2024; Amitai and Amir, 2024; Milani et al., 2024; Hickling et al., 2024; Bekkemoen,

1. https://github.com/observer4599/Outcome-Based-Semifactuals-for-RL



(*a*) **Query state.** The agent moves down.



(*b*) **Counterfactual state.** The agent moves up.

Figure 1: **Ms. Pac-Man.** Example of a counterfactual explanation from the arXiv version of Huber et al. (2023, Figure 3). Licensed under CC BY 4.0 (https://creativecommons.org/licenses/by/4.0/).

2024). The XRL community has mainly focused on leveraging methods from the supervised learning explainability field and often does not consider the specific challenges that make explainability in RL difficult. These challenges include delayed rewards, stochastic environments, and large state spaces (Amitai and Amir, 2024). For example, feature importance is one of the most popular methods in XRL but does not specifically tackle sequential decision-making (Bekkemoen, 2024). We need additional explainability tools tailored towards RL to better understand the behavior in sequential decision-making.

Counterfactual explanations are relatively new in XRL and can potentially help us understand RL agents. However, existing counterfactual methods in RL have problems, as illustrated in the example from Huber et al. (2023). In Fig. 1, we notice two problems with the generative approach to creating counterfactual states. First, the generated game elements like Ms. Pac-Man, ghosts, and some pills look unrealistic, creating out-of-distribution states. Consequently, the generated states can decrease the trust as they do not convince or satisfy the stakeholders' expectations. Second, the counterfactual states should be close to the query states. However, in contrast to supervised learning, the notion of similarity in RL is challenging to define. Visually similar states are not necessarily close. The states might not be reachable from each other, they can occur far from each other in time, and small changes to the states can significantly impact the game semantics. In Fig. 1(*b*), the agent (Ms. Pac-Man) itself is moved, making it difficult to understand what game elements made the agent choose to go down. In Fig. 1(*a*), a retrieval-based approach using a replay buffer can solve the first problem with out-of-distribution states. However, the states we retrieve using a replay buffer can be too visually similar and not have the discriminative power to convey the agent's behavior as seen in Fig. 2 (blue box).

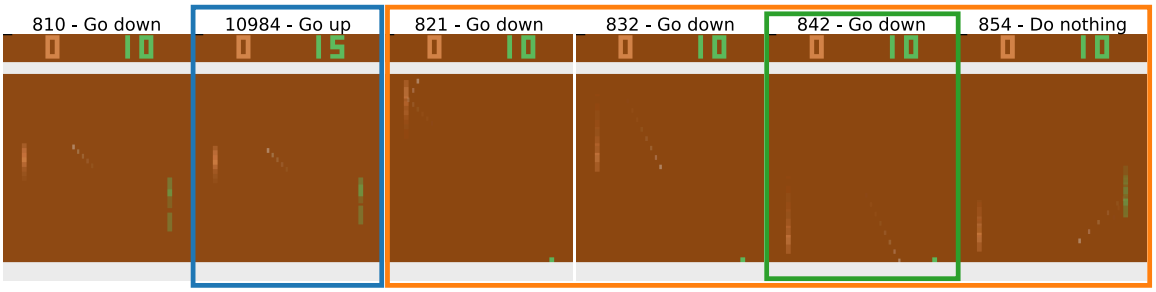


Figure 2: **Pong.** The top row shows the timestep and the action taken in a state. The query state is shown in timestep 810, the counterfactual state (blue) is displayed in timestep 10984, and the outcome-based semifactual (OSF) state (green) in timestep 842. The trajectory from the query state to the OSF state and the state after is shown in the orange box. The counterfactual state is the closest state found via the nearest neighbor search where the distance is measured in the policy's embedding space. The nearest neighbor is selected from a replay buffer of 100 000 states. The timestep counter is not reset when an episode ends, thus, the timestep count for the counterfactual state is large.

Semifactual explanations describe how much we can modify input features without changing the agent's action (Aryal and Keane, 2024). Semifactual states are states furthest away from the query state without crossing the decision boundary as shown in Fig. 3(a). Unfortunately, semifactual states sometimes suffer from a lack of discriminative power like counterfactual explanations. To alleviate the problems mentioned, we propose the outcome-based semifactual (OSF) explanation and introduce a simulation-based approach to find OSF states to explain the agent's behavior. Our OSF explanations differ from semifactual explanations used in supervised learning. We aim to look at it from the value space perspective rather than looking at the probability distribution over actions. Given a query state and simulated trajectories from the query state, we ask how long we can keep executing the query state's action and expect a similar (but not necessarily optimal) outcome. For example, Fig. 3(b) illustrates an OSF state in the mountain car environment where we keep accelerating left beyond the decision boundary while still being able to reach the goal without the agent taking an additional lap. Fig. 2 illustrates an OSF state and a counterfactual state with respect to a query state in Pong. The OSF state shows us the boundary between positive and negative outcomes rather than a single-step decision boundary. Additionally, the OSF state reduces the amount of action switching the agent makes by abstracting away unimportant behavior. From a broader point of view, a set of OSF states can complement other explanation methods and not only be used as a standalone explanation. For instance, we can learn simplified policies through distillation, focusing on mimicking behavior in OSF states. By only focusing on the important states, we reduce the complexity of explanations.

In summary, we propose: 1) Outcome-based semifactual (OSF) explanations that focus on the outcome of an agent. They enable us to understand which action switches are critical to the agent and which can be abstracted away. Furthermore, they are important states on the boundary between positive and negative outcomes that further help us understand the agent. 2) Using the nearest neighbor search, we show the effectiveness of OSF explanations via case studies and compare them to counterfactual explanations. 3) We demonstrate the

use of OSF explanations to minimize action switching by the agent during rollouts. Furthermore, we show that this minimization improves policy performance in six environments while abstracting away unimportant behavior with little explanatory value.

2. Background

Reinforcement Learning. In RL, we model the environment as a Markov decision process (MDP) consisting of a tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$ (Sutton and Barto, 2018; Achiam, 2018; Chapter 2 in Albrecht et al., 2024). $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition probability function, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. Additionally, we have the starting state distribution defined by $\mu : \mathcal{S} \to [0, 1]$. An agent interacts with the environment via a policy $\pi : \mathcal{S} \to \mathcal{A}$ that takes a state and outputs a corresponding action. The policy can either output a distribution over actions or be indirectly defined via a state-action value function. The trajectory is defined by $\tau = (s_1, a_1, s_2, a_2, \ldots, s_N)$ where $s_1 \sim \mu$. The probability distribution over trajectories conditioned on the policy π is defined by $Pr(\tau|\pi) = \mu(s_1) \prod_{t=1}^{N-1} T(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$. We use value functions to measure the expected returns, which are state-based or state-action-based. The expected return starting from the state s_1 and following the policy thereafter is defined by $\mathbb{E}_{\tau \sim Pr(\cdot|\pi)}\big[\sum_{t=1}^{\infty} \gamma^{t-1} R(s_t, a_t, s_{t+1})\big]$. The state value function is defined via the Bellman equation by $V^\pi(s) = \mathbb{E}_{s' \sim T(\cdot|s,a), a \sim \pi(\cdot|s)}[R(s, a, s') + \gamma V^\pi(s')]$. Similarly, the state-action value function known as the Q-function is defined via the Bellman equation by $Q^\pi(s, a) = \mathbb{E}_{s' \sim T(\cdot|s,a)}\big[R(s, a, s') + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q^\pi(s', a')]\big]$. Our work requires access to a policy modeled via the Q-function to measure the expected return. Specifically, we use the deep Q-network (DQN) (Mnih et al., 2013) but any Q-learning method works.

Counterfactual and Semifactual Explanation. We define a counterfactual explanation with respect to a policy π as a tuple $\langle s, s' \rangle$ where s is the query state and s' is the counterfactual state (Guidotti, 2022). Given the policy π and query state s, we get $\pi(s) = a$. The counterfactual state s' is similar to s but where the features have small sparse changes such that $\pi(s') = a' \neq a$. The "optimal" counterfactual state has certain desirable properties such as validity, sparse changes with respect to the query state, and similarity. A semifactual explanation is similar where the features are altered but where the query state s and the semifactual state s' maintain the same action, $\pi(s') = a = \pi(s)$. We want the change between the query and semifactual states to be as large as possible to observe the states' differences while not crossing the decision boundary. Like counterfactual explanations, semifactual explanations should satisfy certain desirable properties (Aryal and Keane, 2023). Figure 3(*a*) shows an example of a query state, a counterfactual state, and a semifactual state in the mountain car environment.

3. Related Work

There are several taxonomies for XRL, each with its strengths and weaknesses (Qing et al., 2023; Glanois et al., 2024; Amitai and Amir, 2024; Milani et al., 2024; Hickling et al., 2024; Bekkemoen, 2024). Our work is motivated and inspired by methods that fall within the post hoc explainability category. Specifically, our method is motivated and inspired by

counterfactual explanation methods for RL and state importance methods that explain by finding and presenting important or critical states to stakeholders.

State Importance. State importance methods are among the most popular in XRL (Huang et al., 2019; Lage et al., 2019; Sequeira and Gervasio, 2020). As far as we know, Amir and Amir (2018) and Huang et al. (2018) proposed the earliest work in this category of methods. They show stakeholders a summary of important states to help the stakeholders understand the global behavior of an agent. User studies have shown that these summaries of important states are effective at aligning mental models (Huber et al., 2021; Amitai and Amir, 2024). Their strengths include ease of use and applicability. They discover and find these important states via simulations. The downside is the need for a state transition model, which can partially be fixed by learning world models (Ha and Schmidhuber, 2018). Our method leverages Q-values as an importance measure. It uses a state transition model like the methods above. Similarly, our goal is to abstract away complex agent behavior by finding important states. However, our important states provide additional insights since they are on the boundary between positive and negative outcomes.

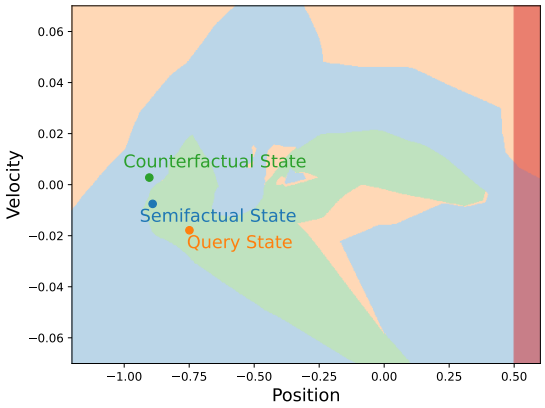
Counterfactual Explanations in RL. Olson et al. (2021) and Huber et al. (2023) generate counterfactual states using generative adversarial networks. The introduction mentioned problems with these approaches, specifically out-of-distribution states, how to handle the similarity measure, and a lack of discriminative power. To solve these problems, we improve upon counterfactual methods by proposing a new type of explanation named OSF explanations.

Semifactual Explanations. Semifactual explanations are not new and have been researched in the context of supervised learning (Kenny and Keane, 2021; Kenny and Huang, 2023). However, there is little work on semifactual explanations in RL (Gajcin et al., 2024). Gajcin et al. (2024) focus on states with discrete features, using the Stochastic Gridworld and Frozen Lake environments in the experiments. In contrast, we focus on environments where the observations are represented as images. Gajcin et al. (2024) use the NSGA-II algorithm (Deb et al., 2002), a genetic algorithm that involves crossover and mutation to generate their explanations. It is unclear how these operations should be applied to high-dimensional states while keeping them valid and meaningful. See the supplementary material for a discussion regarding the desiderata of semifactual explanations.

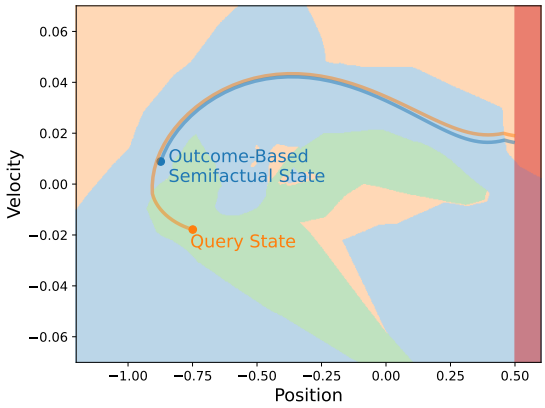
4. Method

We introduce the concept of outcome-based semifactual (OSF) explanations and how we can find OSF states algorithmically.

What is an OSF explanation? An OSF explanation consists of a tuple $\langle s_t, s'_{t+n}, \delta \rangle$ that is constructed with respect to a policy. The state s_t is the query state, s'_{t+n} is the OSF state, and δ is the stopping criterion. We assume the policy is represented as a Q-function. Given $a_t = \arg\max_a Q(s_t, a)$, s'_{t+n} is the state n timesteps after s_t is reached by only executing the action a_t, and the state s_{t+n} is the state n timesteps after s_t is reached by following the policy. The value n is the largest integer for which $\max_a Q(s_{t+j}, a) - Q(s'_{t+j}, a_t) \leq \delta$ for all $j \leq n$. Because s'_{t+n} is produced by a rollout from s_t, we assume that changes in features are sparse and that the OSF state is visually similar to the query state given a small δ value.



(*a*) Example of states: query, counterfactual and semifactual. These states are for illustrative purposes only, manually selected from a trajectory, and not found via an algorithm. The figure is inspired by Aryal and Keane (2024).



(*b*) Example of states: query and OSF. The OSF state is produced by simulating from the query state and is found via our algorithm. The lines show trajectories starting from the two states and following the policy thereafter.

Figure 3: **Mountain Car** (Moore, 1990; Towers et al., 2024). Examples of states with two features, position and velocity. The red area highlights the goal of the environment. The decision boundaries are color-coded where green = accelerate left, orange = do nothing, and blue = accelerate right. We discuss the mountain car environment in Section 5.2.1.

Finding an OSF state. We want to understand how much a query state s_t can be modified by executing the same action without affecting the future outcome negatively. We define negative outcomes as states where the return is significantly lower than what is achieved by following the policy. We seek an OSF state, as seen in Fig. 3(*b*). To find the OSF state, we assume access to a Q-function and a state transition probability function T that enables us to simulate trajectories. The OSF states we find depend on whether the state transition probability function is stochastic or deterministic. Depending on how much stochasticity is in the environment, we might need to simulate several trajectories and show a set of OSF states rather than a single state as in the deterministic case. The experiments presented in Section 5 use deterministic state transition probability functions and are therefore unaffected by stochasticity.

Using the Q-function, we define the importance gap between two states s and s' conditioned on the action a by

$$IG(s, s' \mid a) = \max_{a'} Q(s, a') - Q(s', a). \tag{1}$$

The method starts by first following the policy from the state s_t to find the trajectory $\tau = (s_t, a_t, s_{t+1}, a_{t+1}, \ldots, s_N)$. After obtaining the trajectory τ, we simulate a new trajectory from the state s_t, but instead of following the policy, we execute the action $a_t = \arg\max_a Q(s_t, a)$ in all states that follow and obtain the trajectory $\tau' =$

$(s_t, a_t, s'_{t+1}, a_t, s'_{t+2}, a_t, \ldots, s'_T)$. The OSF state is found by applying the following criterion to the two trajectories τ and τ':

$$n = (\arg\min_i IG(s_{t+i}, s'_{t+i} \mid a_t) > \delta) - 1 \tag{2}$$

Hence, s'_{t+n} is the OSF state. The criterion says to keep executing action a_t as long as the gap is lesser than or equal to δ. The criterion only needs to be one-sided since a negative gap tells us we are reaching states that are better than following the policy. The gap will in general be positive as the policy is "optimal". We allow a small gap since a small loss in expected return will not affect the outcome negatively. For example, a small δ will still allow the policy in mountain car to reach the goal, albeit taking a few more timesteps.

The δ value depends on the environment because the expected return varies based on the reward, which differs between environments. On the one hand, a too small δ value results in the query state and OSF state being very similar and lowers the discriminative power. On the other hand, a too large δ value makes the OSF state dissimilar to the query state and ends up being a state that the policy rarely encounters. The δ value has to be set by a human stakeholder that interactively explores different δ values. We recommend setting the δ value as large as possible as long as there is no significant drop in performance.

5. Experiments

We detail the experimental setup to reproduce our results and give qualitative and quantitative evidence of the method's effectiveness in mountain car and several Atari environments.

5.1. Experimental Setup

The policy used for the mountain car environment is a pre-trained DQN by Raffin (2019). The DQN policies used in Atari environments are trained using CleanRL (Bellemare et al., 2013; Huang et al., 2022). These DQN policies are neural networks with three convolutional layers followed by two linear layers and ReLU activation functions between layers. We use the default hyperparameters used by CleanRL for all Atari environments. All the environments are implemented in Gymnasium v0.29.1 (Towers et al., 2024). Our method only has one hyperparameter, which is the δ value that we document separately for each experiment. For the figures, we overlay observations from 9 timesteps before the indicated timestep to show movements leading up to the state shown. The quantitative results shown in Table 1 are computed by averaging 30 episodes, where the environments are initialized with a new seed for each episode. All experiments ran on a MacBook Pro 2023 with the Apple M2 MAX chip and 64 GB RAM. We used Python v3.11.8 and PyTorch v2.2.2.

5.2. Case Studies

We look at OSF explanations for two query states across three environments: Mountain Car, Pong, and Breakout. For Pong and Breakout, in addition to OSF explanations, we inspect the counterfactual explanations for the same query states. The counterfactual states are found by looking for the closest states based on the policy's embedding space, measured using the Euclidean distance. To obtain the policy embeddings, we save the outputs of

the penultimate layer of the policy network. We generate 100 000 states from simulations to find the closest states. From those 100 000, we select the 100 closest states and sample 4 states from those 100 at random to avoid selecting visually similar states. These 4 are visualized as the counterfactual states in Figs. 6 and 8.

5.2.1. Mountain Car

In the first case study, we look at specific parts of the decision boundary that are unsmooth in the mountain car environment. We ask why they exist and whether they are necessary. Since the mountain car environment only has two features, position and velocity, we can easily visualize its state space and the corresponding decision boundaries.

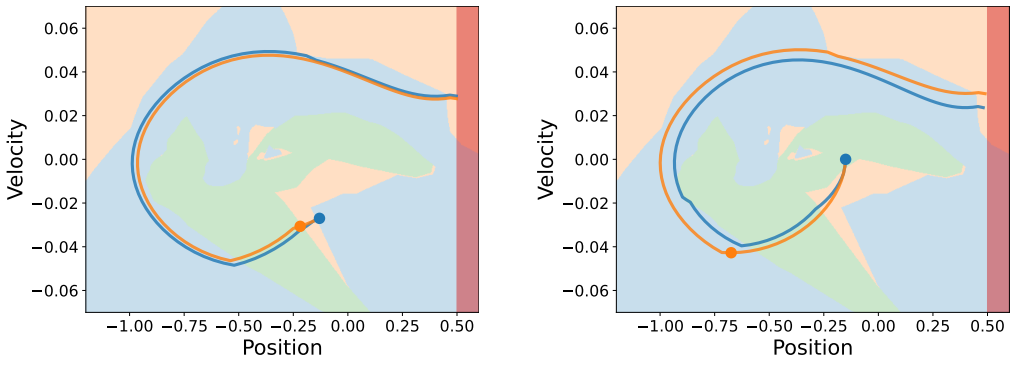
In Fig. 4(a), the query state is in a region where the policy accelerates right but changes quickly to do nothing before changing to accelerate left. In this specific case, we see that the OSF state is in the accelerate left region of the state space. This means we can skip the do nothing action and keep accelerating right until gravity pulls the car into the accelerate left region. By only using the query state action, the agent can reach the goal quicker than by following the policy. It is difficult to conclude that the do nothing region is useless. However, in this case, the complexity of the decision boundary can be reduced.

Fig. 4(a) shows that specific parts of the orange region are not necessary, while Figs. 4(b) and 4(d) depict that other parts are needed. Figs. 4(b) and 4(d) show that the orange region helps reduce the car's acceleration when going towards left. Although we want the car to get high up on the left to get enough speed to go right, too high is unnecessary. Going too high up to the left wastes time because the car can get to the goal with less speed. Fig. 4(c) shows the small blue island next to the query state is unneeded as the OSF trajectory after the island is similar to the original trajectory. Figs. 4(e) and 4(f) show two situations where a small patch of orange area next to the query state decreases the time it takes to reach the goal if we use it and increases the time if we do not use it. We have observed that the complexity of the decision boundaries is sometimes needed. In other cases, they add unnecessary complexity without increasing the policy's performance. We see that OSF states help understand the decision boundaries learned by the agent.

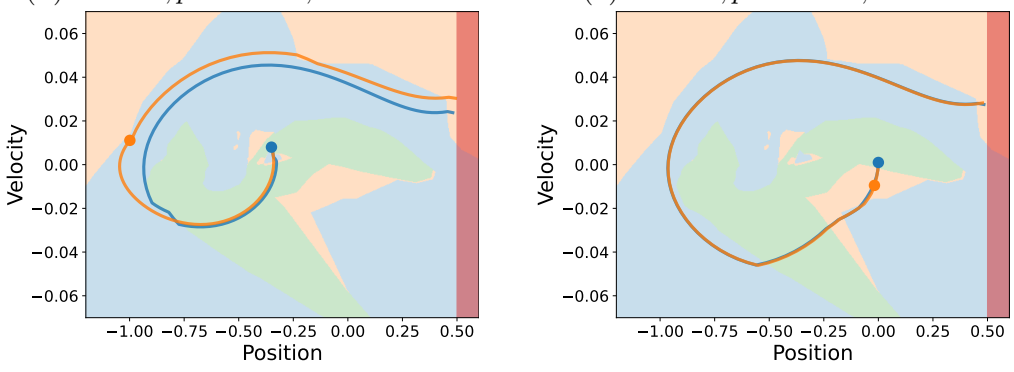
5.2.2. Pong

For Pong, we look at a query state from an episode at timestep 200. Fig. 5 shows the query state and the corresponding OSF explanation. The first row in the figure displays how the policy behaves from timestep 200 to 220. We observe the policy is making several moves, which can be unintuitive for humans as the ball is far away, and thus, those moves are unnecessary and convey little explanatory value. On the second row, we observe that the OSF explanation tells us that none of the moves the policy is making are necessary. The policy only needs to move into a receiving position after timestep 213, which is the OSF state. The OSF explanation simplifies the policy's movements so that we can focus on important moves the policy needs to make. Moreover, the OSF state is the last chance to move into a receiving position, indicating the boundary between outcomes.

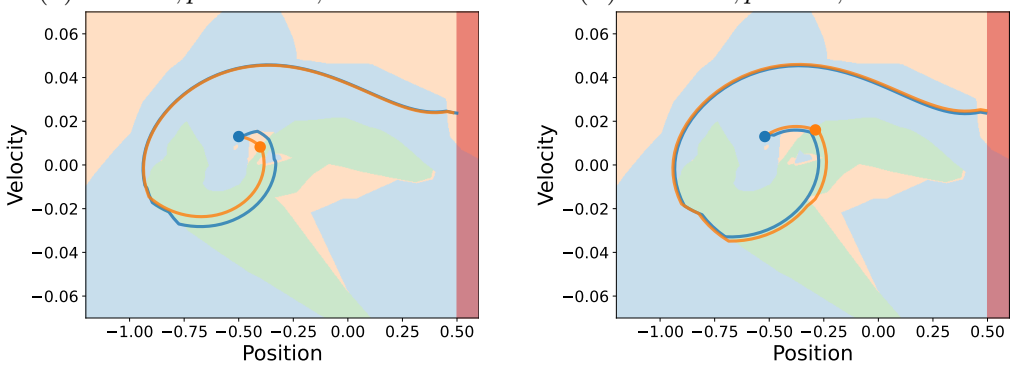
Fig. 6 shows counterfactual explanations for the same query state. The counterfactual explanations indicate that the policy focuses on the ball and the opponent's paddle when representing a state internally as they are the most similar parts of the counterfactual



$(a)\ \delta = 0.5, p = -0.13, v = -0.027$ $(b)\ \delta = 0.1, p = -0.15, v = 0.0$



$(c)\ \delta = 0.5, p = -0.35, v = 0.008$ $(d)\ \delta = 0.43, p = 0.0, v = 0.001$



$(e)\ \delta = 0.5, p = -0.5, v = 0.013$ $(f)\ \delta = 0.5, p = -0.52, v = 0.013$

Figure 4: **Mountain Car.** Decision boundaries of a policy in the **mountain car** environment, where green = accelerate left, orange = do nothing, and blue = accelerate right. The red area highlights the goal of the environment. Query states have blue markers and corresponding OSF states have orange markers. The value δ is the stopping criterion, p indicates position of the car, and v is the velocity at the query state.

states. However, it is difficult to use counterfactuals because the states are similar and it is not clear what decides the action selection. Hence, the counterfactual explanation lacks discriminative power for this query state.

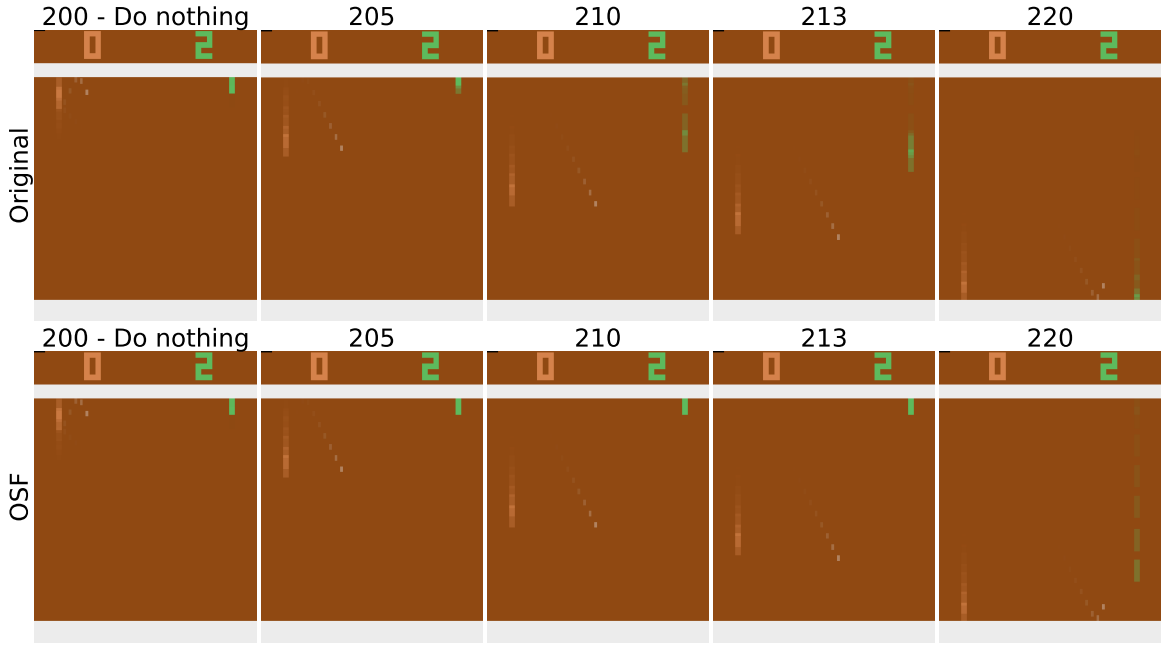


Figure 5: **Pong.** The numbers on top indicate the timesteps. Timestep 200 is the query state and timestep 213 is the OSF state. The sequence of images shows the agent's movement from the query state and onward. The OSF explanation (bottom row) indicates the agent (green paddle) does nothing until the state after the OSF state at the timestep 213. The threshold value is $\delta = 0.05$.

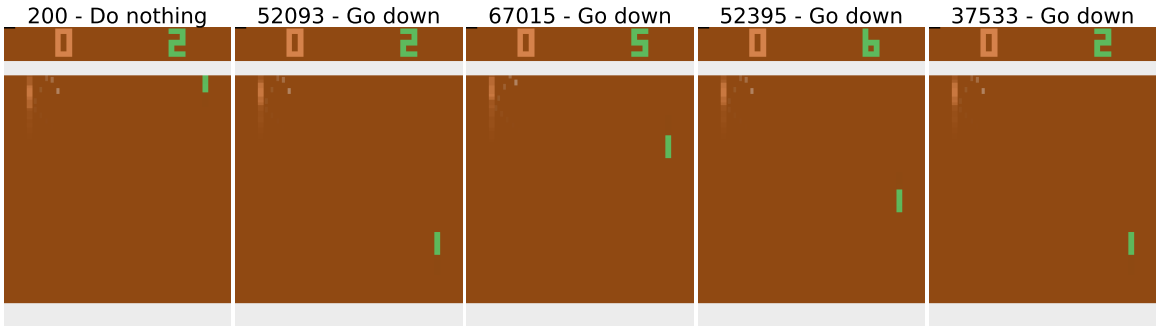


Figure 6: **Pong.** The numbers indicate the timesteps with the corresponding action for the state below. The timestep is not reset when an episode ends. Timestep 200 is the query state. The rest are counterfactual states retrieved based on closeness measured in the embedding space of the policy.

5.2.3. Breakout

We run the same setup for Breakout as we did for Pong. We changed the color of the game background to ease visualization post hoc but did not alter the input given to the agent. Figure 7 demonstrates that when the ball leaves the paddle, it is unnecessary to make any movement. However, without our method, we would not know that the additional move-

Table 1: The performance of policies averaged over 30 episodes. The furthest left column indicates the environment for the experiments. Original refers to the DQN policy without any modification. Simplified is the version where we set the starting state as the query state, find the OSF state, set the next state as the query state, and continue with the same procedure until termination. The same underlying policy is used in both. The value δ is the stopping criterion in Eq. (2). #Timestep/Action Switch refers to how many timesteps on average the policy executes the same action consecutively before switching action.

	Method	δ	Episodic Return ↑	#Timestep/Action Switch ↑
Pong	Original		**21.00**	7.70 ± 0.14
	Simplified	0.02	**21.00**	64.66 ± 7.16
		0.03	**21.00**	75.12 ± 1.84
		0.04	**21.00**	73.27 ± 3.74
		0.05	**21.00**	$\mathbf{75.96 \pm 6.21}$
Ms. Pac-Man	Original		1423.33 ± 331.65	12.79 ± 0.95
	Simplified	0.02	1966.00 ± 185.22	19.21 ± 1.37
		0.03	2119.00 ± 386.20	21.59 ± 2.25
		0.04	2037.00 ± 208.21	24.38 ± 1.82
		0.05	$\mathbf{2363.33 \pm 639.23}$	$\mathbf{29.48 \pm 3.72}$
Space Invaders	Original		1231.00 ± 182.63	10.17 ± 0.18
	Simplified	0.02	$\mathbf{1579.33 \pm 1030.24}$	18.20 ± 1.49
		0.03	1293.50 ± 370.62	23.90 ± 2.03
		0.04	1467.83 ± 876.03	24.67 ± 2.24
		0.05	724.50 ± 197.60	$\mathbf{27.29 \pm 1.65}$
Assault	Original		1681.23 ± 350.44	8.18 ± 0.27
	Simplified	0.02	1833.23 ± 481.17	11.57 ± 0.41
		0.03	2652.50 ± 717.39	14.10 ± 1.42
		0.04	2997.13 ± 2109.81	15.69 ± 1.59
		0.05	$\mathbf{3377.30 \pm 1416.44}$	$\mathbf{16.91 \pm 1.19}$
Seaquest	Original		2928.00 ± 459.85	8.13 ± 0.69
	Simplified	0.02	2238.67 ± 316.79	13.41 ± 0.54
		0.03	2776.00 ± 759.59	14.33 ± 0.65
		0.04	$\mathbf{4088.00 \pm 1697.26}$	15.17 ± 1.16
		0.05	1103.33 ± 833.75	$\mathbf{19.03 \pm 2.69}$
Breakout	Original		382.20 ± 3.50	13.77 ± 3.24
	Simplified	0.02	$\mathbf{387.50 \pm 24.05}$	19.61 ± 4.48
		0.03	308.47 ± 139.57	31.53 ± 18.02
		0.04	375.23 ± 29.42	$\mathbf{35.73 \pm 12.66}$
		0.05	314.37 ± 44.90	23.61 ± 4.27

ments made by the agent are unneeded. Again, erratic movements increase the complexity for a stakeholder trying to understand the agent.

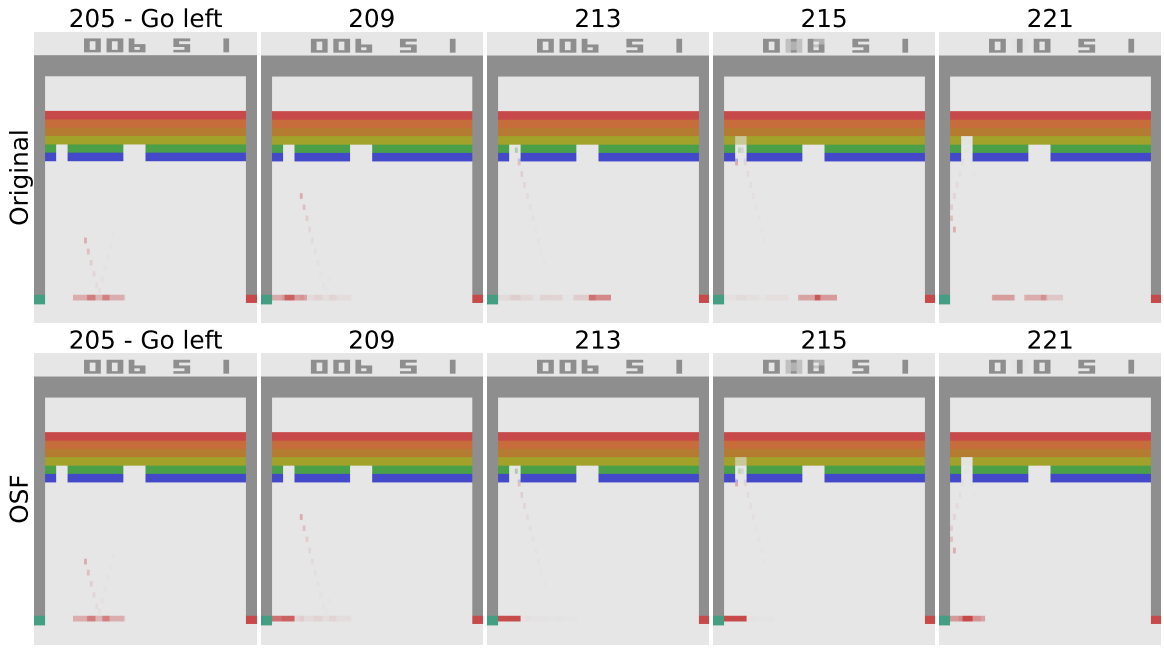


Figure 7: **Breakout.** The numbers on top indicate the timesteps. Timestep 205 is the query state and timestep 215 is the OSF state. The sequence of images shows the movement of the agent from the query state and onward. For the OSF explanation (bottom row), the agent (red paddle) keeps going left until the state after the OSF state at the timestep 215. The threshold value is $\delta = 0.05$.

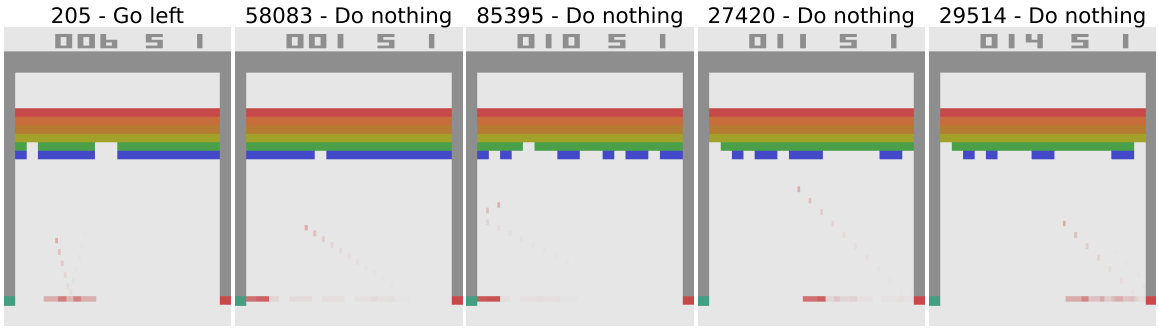


Figure 8: **Breakout.** The numbers on top indicate the timesteps with the corresponding action for the state. The timestep is not reset when an episode ends. Timestep 205 is the query state. The rest of the states are counterfactual states retrieved based on closeness in the embedding space of the policy.

Fig. 8 shows the same query state as in Fig. 7 but with the counterfactual states. These counterfactual states are hard to use as they do not pinpoint specific game elements triggering an action. Moreover, the counterfactual explanations assume that by observing them, the stakeholder should understand the behavior, which we do not believe works well in RL. Finally, the states are similar, yet a different action is triggered. Much of the issue is caused by the decision regions being small. The decision regions in RL are typically not

like in supervised learning where the decision boundary can be far away from the query state.

5.3. Policy Performance with OSF

Until now, we have seen how OSF explanations work qualitatively. To show quantitative results, we roll out policies using our method to show that it results in less action switching. Action switching refers to the act of executing a different action a_t in the current state compared to the action a_{t-1} in the previous state, that is $a_t \neq a_{t-1}$. We set the starting state as the query state and execute the query state's action until a threshold is reached, following Eq. (2). When the threshold is reached, we set the state after the OSF state as the query state and repeat the process. This process repeats until an episode ends. To check whether less action switching happens, we divide an episode's length by the number of times the agent switches actions. We use the same set of δ values across all Atari environments because CleanRL bins the reward to the set $\{-1, 0, 1\}$ depending on the sign of the reward that the agent receives (Raffin et al., 2021; Huang et al., 2022).

Table 1 shows that we can reduce the action switching by at least a factor of two across six environments. Additionally, we observe that the performance of the policy in terms of return is not negatively affected and even improves the behavior in some environments. Surprisingly, the performance increases even though we naively set the query states. These results support the case studies that show there are situations where the agent moves unnecessarily. These extra movements add to the complexity of interpreting the agent without performance gain. Also, the results align with the thought that in RL, not all states have the same importance.

6. Discussion and Conclusion

We presented a new method to understand the long-term behavior of RL agents. The method finds outcome-based semifactual (OSF) explanations that focus on explaining the outcome rather than actions. Given a state of interest, called the query state, the OSF explanation asks how long an agent can execute the same action before it *has to* switch due to a negative outcome. We achieve this by simulating two trajectories and keeping track of the expected return to see what happens if the agent keeps following the policy versus executing the same action. When the difference between these two estimates reaches a threshold, we stop and label the state as the OSF state. Like state importance methods, we show the entire rollout from the query state to the OSF state to better understand the agent's behavior. To demonstrate the usefulness of our method, we qualitatively show examples of explanations produced by our method in three environments. The results show that the agent stops unnecessarily changing actions with our method, decreasing the complexity of understanding its behavior. To emphasize that the usefulness extends beyond these few examples, we quantitatively show how our method can improve policy performance while reducing the number of action switches across six environments. From a broader perspective, our method can complement other XRL explanations so they can focus on explaining important decisions. For instance, policy distillation methods can focus on learning behavior in states that affect outcomes rather than trying to cover all states, which results in complex explanations.

Our method depends on selecting effective query states. However, the same applies to most local explanations, such as counterfactuals or saliency maps. Another limitation of our method is the need for a state transition probability function. This can be improved by training world models. Because our method does not need to forecast far into the future, we believe small compounding errors in the state transition probability model should not significantly impact our method. Finally, our method can be computationally intensive. We need two environments initialized at the same query state for rollouts and comparisons. In the future, we should perform evaluations through user studies and consider how they should be set up so that the results are comparable to other works. Furthermore, we should investigate how using world models affects the method.

References

J. Achiam. Spinning Up in Deep Reinforcement Learning, 2018. URL https://github.com/openai/spinningup.

S. V. Albrecht, F. Christianos, and L. Schäfer. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. The MIT Press, 2024. ISBN 9780262380508.

D. Amir and O. Amir. HIGHLIGHTS: Summarizing Agent Behavior to People. In *Proc. of AAMAS*, 2018. URL http://dl.acm.org/citation.cfm?id=3237869.

Y. Amitai and O. Amir. *A Survey of Global Explanations in Reinforcement Learning*, page 21–42. CRC Press, 2024. doi: 10.1201/9781003355281-2.

A. B. Arrieta, N. D. Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion*, 2020. doi: 10.1016/J.INFFUS.2019.12.012.

S. Aryal and M. T. Keane. Even If Explanations: Prior Work, Desiderata & Benchmarks for Semi-Factual XAI. In *Proc. of IJCAI*, 2023. doi: 10.24963/IJCAI.2023/732.

S. Aryal and M. T. Keane. Even-Ifs from If-Onlys: Are the Best Semi-factual Explanations Found Using Counterfactuals as Guides? In *Proc. of ICCBR*, 2024. doi: 10.1007/978-3-031-63646-2_3.

Y. Bekkemoen. Explainable reinforcement learning (XRL): a systematic literature review and taxonomy. *Mach. Learn.*, 2024. doi: 10.1007/S10994-023-06479-7.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *J. Artif. Intell. Res.*, 2013. doi: 10.1613/JAIR.3912.

N. Brown and T. Sandholm. Libratus: The Superhuman AI for No-Limit Poker. In *Proc. of IJCAI*, 2017. doi: 10.24963/IJCAI.2017/772.

K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.*, 2002. doi: 10.1109/4235.996017.

J. Gajcin, J. Jeromela, and I. Dusparic. Semifactual Explanations for Reinforcement Learning. In *Proc. of HAI*, 2024. doi: 10.48550/arXiv.2409.05435.

C. Glanois, P. Weng, M. Zimmer, D. Li, T. Yang, J. Hao, and W. Liu. A survey on interpretable reinforcement learning. *Machine Learning*, 2024. doi: 10.1007/s10994-024-06543-w.

R. Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Min. Knowl. Discov.*, 2022. doi: 10.1007/s10618-022-00831-6.

D. Ha and J. Schmidhuber. Recurrent World Models Facilitate Policy Evolution. In *Proc. of NeurIPS*, 2018. doi: 10.48550/arXiv.1809.01999.

T. Hickling, A. Zenati, N. Aouf, and P. Spencer. Explainability in Deep Reinforcement Learning: A Review into Current Methods and Applications. *ACM Comput. Surv.*, 2024. doi: 10.1145/3623377.

S. Huang, R. F. J. Dossa, C. Ye, J. Braga, D. Chakraborty, K. Mehta, and J. G. M. Araújo. CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms. *J. Mach. Learn. Res.*, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

S. H. Huang, K. Bhatia, P. Abbeel, and A. D. Dragan. Establishing Appropriate Trust via Critical States. In *Proc. of IROS*, 2018. doi: 10.1109/IROS.2018.8593649.

S. H. Huang, D. Held, P. Abbeel, and A. D. Dragan. Enabling robots to communicate their objectives. *Auton. Robots*, 2019. doi: 10.1007/S10514-018-9771-0.

T. Huber, K. Weitz, E. André, and O. Amir. Local and global explanations of agent behavior: Integrating strategy summaries with saliency maps. *Artif. Intell.*, 2021. doi: 10.1016/J.ARTINT.2021.103571.

T. Huber, M. Demmler, S. Mertes, M. L. Olson, and E. André. GANterfactual-RL: Understanding Reinforcement Learning Agents' Strategies through Visual Counterfactual Explanations. In *Proc. of AAMAS*, 2023. doi: 10.48550/ARXIV.2302.12689.

E. M. Kenny and W. Huang. The Utility of "Even if" Semifactual Explanation to Optimise Positive Outcomes. In *Proc. of NeurIPS*, 2023. doi: 10.48550/ARXIV.2310.18937.

E. M. Kenny and M. T. Keane. On Generating Plausible Counterfactual and Semi-Factual Explanations for Deep Learning. In *Proc. of AAAI*, 2021. doi: 10.1609/AAAI.V35I13.17377.

I. Lage, D. Lifschitz, F. Doshi-Velez, and O. Amir. Exploring Computational User Models for Agent Policy Summarization. In *Proc. of IJCAI*, 2019. doi: 10.24963/IJCAI.2019/194.

S. Milani, N. Topin, M. Veloso, and F. Fang. Explainable Reinforcement Learning: ASurvey and Comparative Review. *ACM Comput. Surv.*, 2024. doi: 10.1145/3616864.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing Atari with Deep Reinforcement Learning. *NIPS Deep Learning Workshop*, 2013. doi: 10.48550/arXiv.1312.5602.

A. W. Moore. Efficient Memory-based Learning for Robot Control. Technical report, University of Cambridge, 1990. URL https://doi.org/10.48456/tr-209.

M. L. Olson, R. Khanna, L. Neal, F. Li, and W. Wong. Counterfactual state explanations for reinforcement learning agents via generative deep learning. *Artif. Intell.*, 2021. doi: 10.1016/J.ARTINT.2021.103455.

Y. Qing, S. Liu, J. Song, and M. Song. A Survey on Explainable Reinforcement Learning: Concepts, Algorithms, Challenges. *CoRR*, 2023. doi: 10.48550/ARXIV.2211.06665.

A. Raffin. RL Baselines3 Zoo: A Training Framework for Stable Baselines3 Reinforcement Learning Agents, 2019. URL https://github.com/DLR-RM/rl-baselines3-zoo.

A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *J. Mach. Learn. Res.*, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. P. Lillicrap, and D. Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nat.*, 2020. doi: 10.1038/S41586-020-03051-4.

P. Sequeira and M. T. Gervasio. Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. *Artif. Intell.*, 2020. doi: 10.1016/J.ARTINT.2020.103367.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nat.*, 2016. doi: 10.1038/NATURE16961.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction, Second Edition*. The MIT Press, 2018. ISBN 9780262352703.

C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone. Deep Reinforcement Learning for Robotics: A Survey of Real-World Successes. *CoRR*, 2024. doi: 10.48550/arXiv.2408.03539.

M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, et al. Gymnasium: A Standard Interface for Reinforcement Learning Environments. *CoRR*, 2024. doi: 10.48550/arXiv.2407.17032.

W. Yang, Y. Wei, H. Wei, Y. Chen, G. Huang, X. Li, R. Li, N. Yao, X. Wang, X. Gu, M. B. Amin, and B. Kang. Survey on Explainable AI: From Approaches, Limitations and Applications Aspects. *Hum. Centric Intell. Syst.*, 2023. doi: 10.1007/S44230-023-00038-Y.

C. Yu, J. Liu, S. Nemati, and G. Yin. Reinforcement Learning in Healthcare: A Survey. *ACM Comput. Surv.*, 2023. doi: 10.1145/3477600.