Breaking Habits: On the Role of the Advantage Function in Learning Causal State Representations

Anonymous authors
Paper under double-blind review

Abstract

Recent work has shown that reinforcement learning agents can develop policies that exploit spurious correlations between rewards and observations. This phenomenon, known as policy confounding, arises because the agent's policy influences both past and future observation variables, creating a feedback loop that can hinder the agent's ability to generalize beyond its usual trajectories. In this paper, we show that the advantage function, commonly used in policy gradient methods, not only reduces the variance of gradient estimates but also mitigates the effects of policy confounding. By adjusting action values relative to the state representation, the advantage function downweights state-action pairs that are more likely under the current policy, breaking spurious correlations and encouraging the agent to focus on causal factors. We provide both analytical and empirical evidence demonstrating that training with the advantage function leads to improved out-of-trajectory performance.

1 Introduction

Imagine a robot trained to perform two tasks: first, navigate from your office to the coffee machine, retrieve a cup of coffee, and return; second, go to the printer room, make copies, and return. There are two possible routes to the coffee machine: one through the printer room and another through a direct corridor. Since the corridor route is shorter, the robot typically avoids the printer room when fetching coffee. However, one day, the corridor is blocked and the robot takes the path through the printer room, unexpectedly returning with a copy of a paper titled *Breaking Habits*.

Why did this happen? After repeatedly performing these tasks, the robot incorrectly associated the printer room with the need to make copies, turning this misassociation into a habit. While this habit works fine under normal conditions, it fails when the robot is forced to deviate from its usual path. This failure mode, referred to as *out-of-trajectory generalization*, was explored by Suau et al. (2024) in the context of reinforcement learning (RL). The authors showed that such issues arise because the agent's policy introduces spurious correlations (Pearl et al., 2016) between rewards and observations, a phenomenon they termed *policy confounding*.

Contributions In this paper, we observe that the advantage function, commonly used in many policy gradient methods, not only reduces the variance of gradient updates but also plays a crucial role in mitigating this issue. We demonstrate, both analytically and experimentally, that using the advantage function encourages the agent to learn representations that rely more on the true causal factors. It achieves this by scaling state-action pairs according to their probability under a given policy and state representation, effectively breaking the spurious correlations introduced by the agent's policy. This encourages the agents to focus on causal factors, enabling better out-of-trajectory generalization.

To support the theoretical findings, experiments are conducted in three simple environments. The results indicate that agents trained with Q-values fail to generalize beyond their usual trajectories, whereas agents trained with the advantage function exhibit robustness to trajectory deviations. Furthermore, an analysis of the learned state representations reveals that the latter focus on causal factors rather than exploiting spurious correlations.

Finally, we examine the impact of implementation choices, such as batch size and advantage normalization. Through experiments, we show that these factors can significantly affect the learned state representations.

2 Example: Key2Door

We will use the following example throughout the paper to illustrate the ideas presented.

Figure 1 depicts a gridworld environment. The agent's objective is to collect a key placed at the beginning of the corridor and then open the door at the end. The agent's observation consists of its current location $L \in \{0, 1, .., 6\}$ and a binary variable $X \in \{0, 1\}$ indicating whether or not it has the key. At all states, the agent has two possible actions: moving left $A = 0$, or moving right $A = 1$. The agent receives a reward of $+1$ if it reaches the door after collecting the key, and 0 if it reaches the door without the key. Additionally, there is a -0.01 penalty per timestep to incentivize the agent to take the most direct path.

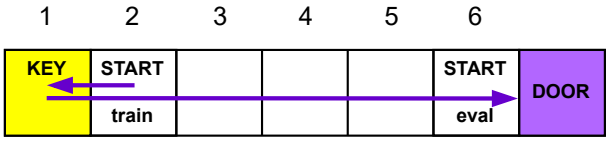

Figure 1: Key2Door environment.

During training, the agent always starts the episode in location 2. This implies that once the agent has collected the key and is moving toward the door, it can disregard the key once it reaches location 3. The agent can learn to use its location to infer whether it has the key. However, this only works well if the agent consistently follows the optimal policy, as depicted by the purple arrow. If the agent moves randomly through the corridor, it cannot reliably determine whether the key has been collected. Moreover, an agent that relies solely on its location will fail when evaluated in the same task but starting from location 6. The agent will attempt to move right to open the door, even though it has not collected the key.

3 Related work

The term policy confounding was introduced by Suau et al. (2024) to describe a phenomenon in which policies, by influencing both past and future observation variables, induce spurious correlations. These correlations can lead the agent to develop shortcuts, referred to as habits, that are effective only within the trajectories the agent typically follows but fail to generalize out-of-trajectory. Out-of-trajectory (OOT) generalization is a specific instance of the broader problem of out-of-distribution (OOD) generalization (Kirk et al., 2023). Unlike OOD generalization, which focuses on adapting to environments with different rewards (Taylor & Parr, 2009), observations (Mandlekar et al., 2017; Zhang et al., 2020a), or transition dynamics (Higgins et al., 2017), OOT generalization seeks to enable agents to generalize across alternative trajectories within the same environment. Empirical evidence of policy confounding has been reported in several studies (Machado et al., 2018; Song et al., 2020; Lan et al., 2023; He et al., 2024; Weltevrede et al., 2024). Similarly, prior work has examined confounding in the context of imitation learning (De Haan et al., 2019; Zhang et al., 2020b; Tennenholtz et al., 2021; Ding et al., 2023). However, unlike policy confounding, these works focus on cases where the agent does not actively contribute to the formation of spurious correlations.

The connection between the advantage function and causality has been explored in prior work. Corcoll & Vicente (2022) use the advantage function to differentiate between agent-driven effects and external environmental factors, constructing a hierarchy of transformations that the agent can perform on its environment. Pan et al. (2022) argue that the advantage function can be interpreted as a measure of an action's causal effect on return and introduce a method for learning advantages directly from on-policy data without relying on a value function. This approach was later extended to support off-policy data (Pan & Schölkopf, 2024). While not explicitly framing the advantage function in causal terms, Raileanu & Fergus (2021) describe it as a measure of the expected additional return gained by selecting a particular action over following the

current policy. Their experiments suggest that the advantage function is less prone to overfitting to certain environment-specific idiosyncrasies.

Our work is strongly influenced by Chung et al. (2021), who showed that the choice of baseline in policy gradient updates can sometimes lead to overly aggressive updates, causing what the authors term committal behavior, which may result in suboptimal policies. Mei et al. (2022) further analyzed the theoretical properties of state-value baselines (i.e., advantages), demonstrating that they moderate update aggressiveness and ensure convergence to the optimal policy. As we discuss in the following sections, these insights are closely connected to the findings presented in this work.

4 Preliminaries

We focus on problems where states are represented by a set of observation variables, or factors (Boutilier et al., 1999). This representation is common in tasks that involve modeling policies and value functions using function approximators (François-Lavet et al., 2018). These observation variables typically describe features of the agent's state in the environment.

Definition 1 (MDP)**.** A Markov decision process (MDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$, where $\mathcal{S}$ represents the set of states, $\mathcal{A}$ denotes the set of actions available to the agent, $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, and $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function.

Definition 2 (FMDP)**.** A Factored Markov decision process (FMDP) is an MDP where the set of states is defined by a set of observation variables, or factors, $F = F^1, \ldots, F^{|F|}$. Each variable F^i can take any value from its domain $\mathcal{F}^i$. Consequently, each state s corresponds to a unique combination of values for the variables in F, i.e., $s = \langle f^1, \ldots, f^{|F|} \rangle \in \times\mathcal{F} = \mathcal{S}$.

4.1 State representations

The agent's objective is to learn a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected discounted sum of rewards (Sutton & Barto, 2018). However, learning a policy that conditions on every observation variable might be impractical, especially in scenarios with a large number of variables. Fortunately, in many problems, not all variables are essential, and compact state representations can be found that are sufficient for solving the task at hand (McCallum, 1995). This is where function approximators, such as neural networks, come into play. When used to model policies and value functions, they learn to ignore certain observation variables in F if those variables are deemed unnecessary for estimating values and optimal actions.

Definition 3 (State representation)**.** A state representation is a function $\Phi : \mathcal{S} \to \bar{\mathcal{S}}$, where $\mathcal{S} = \times\mathcal{F}$, $\bar{\mathcal{S}} = \times\bar{\mathcal{F}}$, and $\bar{F} \subseteq F$.

Intuitively, a state representation $\Phi(s_t)$ is a projection of a state $s \in \mathcal{S} = \times\mathcal{F}$ onto a lower-dimensional space $\bar{\mathcal{S}} = \times\bar{\mathcal{F}}$, defined by a subset of its variables, $\bar{F} \subseteq F$. We use $\{s\}^\Phi = \{s' \in \mathcal{S} : \Phi(s') = \Phi(s)\}$ to denote the equivalence class of s under Φ.

In the Key2Door example a potential state representation could be $\Phi(s_t) = \langle l_t \rangle$ for all $s_t \in \mathcal{S}$. This representation retains only the agent's current location, ignoring all other variables in F. Therefore, all states sharing the same value for L_t belong to the same equivalence class.

Not all state representations are sufficient to learn the optimal policy. Some representations may exclude variables that carry valuable information for solving the task. In the Key2Door example, knowing whether the key has been collected is crucial to selecting the next action.

Definition 4 (Markov state representation)**.** A state representation $\Phi(s_t)$ is said to be Markov if, for all $s_t, s_{t+1} \in \mathcal{S}$ and $a_t \in \mathcal{A}$,

$$R(s_t, a_t) = R(\Phi(s_t), a_t) \quad \text{and} \quad \sum_{s'_{t+1} \in \{s_{t+1}\}^\Phi} T(s'_{t+1} \mid s_t, a_t) = P(\Phi(s_{t+1}) \mid \Phi(s_t), a_t),$$

where $R(\Phi(s_t), a_t)$ denotes the reward $R(s'_t, a_t)$ for any $s'_t \in \{s_t\}^\Phi$.

4.2 Policy confounding

The phenomenon of policy confounding plays a critical role in the search for simpler state representations. The policy can mislead the function approximator into forming simpler state representations that rely on spurious correlations, such as having the key when being at location 6, rather than true causal factors.

Definition 5 (Policy Confounding)**.** A state representation $\Phi : \mathcal{S} \to \bar{\mathcal{S}}$ is said to be confounded by a policy π if, for some $s_t, s_{t+1} \in \mathcal{S}$ and $a_t \in \mathcal{A}$,

$$R^\pi(\Phi(s_t), a_t) \neq R^\pi(\text{do}(\Phi(s_t)), a_t)$$

or

$$P^\pi(\Phi(s_{t+1}) \mid \Phi(s_t), a_t) \neq P^\pi(\Phi(s_{t+1}) \mid \text{do}(\Phi(s_t)), a_t).$$

where $R^\pi(\Phi^\pi(s_t), a_t)$ is the reward $R(s'_t, a_t)$ at any $s'_t \in \{s_t\}^\Phi_\pi$, with $\{s\}^\Phi_\pi = \{s' \in S^\pi : \Phi^\pi(s') = \Phi^\pi(s)\}$, and P^π is probability under π.

The operator do$(\cdot)$ is known as the do-operator, and it is used to represent physical interventions in a system (Pearl et al., 2016). These interventions are meant to distinguish cause-effect relations from mere statistical associations. In our case, do$(\Phi(s_t))$ means setting the variables forming the state representation $\Phi(s_t)$ to a particular value and considering all possible states in the equivalence class, $s'_t \in \{s_t\}^\Phi$, (i.e., all states that share the same value for the observation variables that are selected by Φ; independently of whether these are visited by the policy being followed). For instance, in the Key2Door example, $R^{\pi^*}(L = 6) = +1$ when following the optimal policy π^* (purple path in Figure 1) since we know that the agent has the key, while $R^{\pi^*}(\text{do}(L = 6)) = \pm 1$ since the agent may or may not have the key.

Strictly speaking, the only representation that satisfies the above conditions is the Markov state representation. This representation is invariant to the agent's policy, making it robust to any deviations in the trajectory. In the following, we refer to Markov state representations as *Causal State Representations*, as they necessarily include all the causal factors governing rewards and transitions.

4.3 Policy gradient and advantage function

Suau et al. (2024) demonstrated that the phenomenon of policy confounding is particularly problematic when training agents with on-policy methods. This is because, when updating the policy, on-policy methods rely solely on trajectories collected using the current policy. This contrasts with off-policy methods, where the agent is trained on trajectories generated by multiple policies, thus broadening the trajectory distribution and reducing the risk of the agent picking up on spurious correlations present in specific trajectories.

A popular family of on-policy methods includes those that directly optimize the policy by following the gradient of the expected return with respect to the policy parameters, θ. The policy gradient theorem (Marbach & Tsitsiklis, 1999; Sutton et al., 1999) formalizes this as:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s_t, a_t \sim \pi_\theta} \left[\frac{\nabla_\theta \pi_\theta(a_t \mid s_t)}{\pi_\theta(a_t \mid s_t)} Q^\pi(s_t, a_t) \right], \tag{1}$$

where $Q^\pi(s_t, a_t)$ is the action-value function under policy π.

Following this gradient increases the likelihood of sampling actions that lead to high returns while reducing the probability of actions leading to lower returns. However, in practice, computing the exact gradient is infeasible. Instead, we approximate the gradient using sample estimates from trajectories collected by the policy, which introduces high variance.

A common strategy to reduce variance is to subtract the state value function $V^\pi(s_t)$ from the Q-value (Baird, 1994; Greensmith et al., 2001), leading to the definition of the advantage function:

$$A^\pi(s_t, a_t) := Q^\pi(s_t, a_t) - V^\pi(s_t). \tag{2}$$

This adjustment does not introduce bias, as the value function is independent of the action, but it significantly reduces the variance of the gradient estimation.

5 The role of the advantage in learning causal state representations

In this section, we demonstrate that, beyond its well-known role in variance reduction, the advantage function plays a crucial role in mitigating policy confounding. Specifically, it helps break any spurious correlations that the policy may introduce, thus preventing the agent from forming habits that fail to generalize beyond its typical trajectories. Proofs for the theoretical results in this section can be found in Appendix A.

5.1 Value and advantage function under a state representation

Computing the advantage usually involves fitting a model to approximate the value function. Conceptually, the value function can be decomposed into a state representation function $\Phi(s)$, which projects the states into a lower-dimensional space defined by the subset of variables relevant to the task, and a function $V^\pi(\Phi(s))$ that maps the state representation to a scalar value.

Definition 6 (V^π and Q^π under Φ)**.** Let $\Phi : \mathcal{S} \to \bar{\mathcal{S}}$ be a state representation (Definition 3) that induces an equivalence class $\{s\}^\Phi := \{s' \in \mathcal{S} : \Phi(s') = \Phi(s)\}$. Assume the Markov chain under policy π admits a stationary distribution d^π over $\mathcal{S}$. Then the state value $V^\pi : \bar{\mathcal{S}} \to \mathbb{R}$ and state-action value $Q^\pi : \bar{\mathcal{S}} \times \mathcal{A} \to \mathbb{R}$ under Φ are defined as:

$$V^\pi(\Phi(s_t)) := \sum_{s'_t \in \{s_t\}^\Phi} P^\pi(s'_t \mid \Phi(s_t))\, V^\pi(s'_t)$$

and

$$Q^\pi(\Phi(s_t), a_t) := \sum_{s'_t \in \{s_t\}^\Phi} P^\pi(s'_t \mid \Phi(s_t))\, Q^\pi(s'_t, a_t),$$

where the distribution $P^\pi(s'_t \mid \Phi(s_t))$ is defined as:

$$P^\pi(s'_t \mid \Phi(s_t)) := \frac{d^\pi(s'_t)}{\sum_{s \in \{s_t\}^\Phi} d^\pi(s)} \quad \text{for all } s'_t \in \{s_t\}^\Phi.$$

Lemma 1. *Let Φ be a Causal (Markov) State Representation (Definition 4), then*

$$Q^\pi(s_t, a_t) = Q^\pi(s'_t, a_t) = Q^\pi(\Phi(s_t), a_t) \qquad \textit{and} \qquad V^\pi(s_t) = V^\pi(s'_t) = V^\pi(\Phi(s_t)) \tag{3}$$

for all $a \in \mathcal{A}$, $s \in \mathcal{S}$, and, $s' \in \{s\}^\Phi$.

Definition 7 (A^π_Φ under Φ)**.** Given a state representation $\Phi : \mathcal{S} \to \bar{\mathcal{S}}$, the advantage function $A^\pi_\Phi : \mathcal{S} \to \mathbb{R}$ under Φ is defined as:

$$A^\pi_\Phi(s_t, a_t) := Q^\pi(s_t, a_t) - V^\pi(\Phi(s_t)). \tag{4}$$

Despite the agent's state representation being $\Phi(s_t)$, Definition 7 shows that the advantage function A^π_Φ still depends on the full state s_t. This is because $Q^\pi(s_t, a_t)$ is typically estimated directly for each state s_t and action a_t from Monte Carlo rollouts, making it independent of $\Phi(s_t)$. In contrast, $V^\pi(\Phi(s_t))$ averages over the value of all states equivalent under Φ, effectively marginalizing out the variables not preserved by Φ. As a result, A^π_Φ measures the deviation of a particular state-action pair from the expected value across the equivalence class $\{s_t\}^\Phi$.

5.2 The scaling effect of the advantage function

Theorem 1. *Let Φ be an arbitrary state representation. The advantage function can be expressed as:*

$$A^{\pi}_{\Phi}(s_t, a_t) = (1 - P^{\pi}(s_t, a_t \mid \Phi(s_t)))(Q^{\pi}(s_t, a_t) - \tilde{Q}^{\pi}(\neg\langle s_t, a_t\rangle)) \tag{5}$$

where

$$\tilde{Q}^{\pi}_{\Phi}(\neg\langle s_t, a_t\rangle) = \frac{\sum_{s'_t, a'_t \neq s_t, a_t} P^{\pi}(s'_t, a'_t \mid \Phi(s_t)) Q^{\pi}(s'_t, a'_t)}{\sum_{s'_t, a'_t \neq s_t, a_t} P^{\pi}(s'_t, a'_t \mid \Phi(s_t))} \tag{6}$$

This result highlights that the advantage function is proportional to the complement of the joint probability of a specific state-action pair conditioned on the state representation, i.e., $1 - P^{\pi}(s_t, a_t \mid \Phi(s_t))$. This ensures that the advantage of less probable state-action pairs is larger in magnitude, and helps offset the overrepresentation of more frequent pairs under π.

This scaling effect is crucial for mitigating policy confounding as it breaks any spurious correlations induced by the policy. By amplifying rare but informative samples, the advantage function encourages the agent to attend to underlying causal factors rather than simply reinforcing habits. Intuitively, it reweights each sample according to its causal content, preventing high-probability pairs from dominating the gradient updates during training.

Moreover, the difference $Q^{\pi}(s_t, a_t) - \tilde{Q}^{\pi}_{\Phi}(\neg\langle s_t, a_t\rangle)$ reflects how much better (or worse) a given pair is relative to the alternative pairs. Notably, $\tilde{Q}^{\pi}_{\Phi}(\neg\langle s_t, a_t\rangle)$ is a weighted average over state-action pairs in the same equivalence class, excluding $\langle s_t, a_t\rangle$, and since it does not depend on $P^{\pi}(s_t, a_t \mid \Phi(s_t))$, it provides a stable, representation-conditional baseline for comparison.

Corollary 1. *Let Φ be a Causal (Markov) State Representation. The advantage function can be expressed as:*

$$A^{\pi}_{\Phi}(s_t, a_t) = (1 - \pi(a_t \mid \Phi(s_t)))(Q^{\pi}(s_t, a_t) - \tilde{Q}^{\pi}(s_t, \neg a_t)) \tag{7}$$

where

$$\tilde{Q}^{\pi}(s_t, \neg a_t) = \frac{\sum_{a'_t \neq a_t} \pi(a'_t \mid \Phi(s_t)) Q^{\pi}(s_t, a'_t)}{\sum_{a'_t \neq a_t} \pi(a'_t \mid \Phi(s_t))} \tag{8}$$

The above result follows directly from Theorem 1 and Lemma 1. Note that when the state representation Φ is causal, there is no need to scale the sample gradients according to the likelihood of the states given Φ, as Φ already accounts for the true factors governing rewards and transitions.

5.3 Impact on policy gradients

None of the above would matter if the updates strictly followed the exact policy gradient, as the advantage function does not alter the gradient (Sutton & Barto, 2018). However, as discussed in Section 4.3, policy gradients are typically based on sample estimates (i.e., stochastic gradients).

As a result, depending on the agent's policy and the batch size, it is often the case that for a given $\Phi(s_t)$, all samples in a training batch correspond to a single state s_t, rather than including all states $s'_t \in \{s\}^{\Phi}$. Consequently, when using Q-values instead of advantages A^{π}_{Φ}, this can lead to overly aggressive gradient updates (Chung et al., 2021), causing the agent to develop habits based on spurious correlations. This issue is further exacerbated as the policy reinforces the likelihood of frequently observed state-action pairs, creating a vicious circle (Mei et al., 2022). In the extreme, when the policy becomes deterministic, certain state-action pairs may have their probabilities driven to zero, requiring an infeasibly large number of samples to correct for the overestimation of the state-action pairs that are actually visited.

In contrast, the advantage function A^{π}_{Φ} scales the gradients by adjusting their magnitude based on the probability of each state-action pair (Theorem 1). This moderates the aggressiveness of gradients for frequently sampled state-action pairs while boosting gradients for less common ones, thereby helping the agent break bad habits.

5.4 Practical considerations

Given the points above, an alternative to using advantages could be to employ large batch sizes. However, the batch size may need to be significantly increased depending on the environment for this approach to be effective. Moreover, we find no compelling reason to avoid using advantages.

It is also important to note that the results reported by Suau et al. (2024) indicate that PPO struggles with out-of-trajectory generalization, despite the default setting utilizing advantages. We hypothesize that this issue arises because, in most PPO implementations, the advantages in each training batch are normalized before the network update. This normalization removes the scaling effect of the advantage function on the gradients. Both of these aspects are further analyzed in the experiments section.

5.5 Example: Key2Door

To illustrate the insights discussed in the previous sections, we revisit the Key2Door example (Section 2). Suppose the agent's state representation consists solely of the location variable, $\Phi(s) = l$. Table 1 shows the Q-values and corresponding advantages A^{π}_{Φ} when the agent is at location 6, both with and without the key, under five different policies. These policies differ in the probability of selecting the optimal action: 0.5, 0.6, 0.7, 0.8, and 0.9. The last two columns indicate the probability that the agent has, $P^{\pi}(X = 1 \mid L = 6)$, or has not, $P^{\pi}(X = 0 \mid L = 6)$, the key at location 6.[1]

Table 1: Q-values, advantages, and probabilities of key and no key when the agent is at location 6 under five different policies. The agent's state representation consists solely of the location variable.

	Q-value				**Advantage $\Phi(s) = l$**				$P^{\pi}(X \mid L = 6)$	
	No Key		**Key**		**No Key**		**Key**		**No Key**	**Key**
	Left	**Right**	**Left**	**Right**	**Left**	**Right**	**Left**	**Right**		
$\pi(a^* \mid s) = 0.5$	0.038	0	0.662	1	-0.657	-0.695	-0.033	0.305	0.168	0.832
$\pi(a^* \mid s) = 0.6$	0.274	0	0.839	1	-0.608	-0.882	-0.043	0.118	0.069	0.931
$\pi(a^* \mid s) = 0.7$	0.504	0	0.905	1	-0.456	-0.960	-0.055	0.040	0.018	0.982
$\pi(a^* \mid s) = 0.8$	0.664	0	0.934	1	-0.321	-0.985	-0.051	0.015	0.003	0.997
$\pi(a^* \mid s) = 0.9$	0.759	0	0.950	1	-0.235	-0.994	-0.044	0.006	0.000	0.999

As revealed by Theorem 1, the magnitudes of the advantages depend on the joint probability of visiting a specific state-action pair. In particular, the advantage decreases (increases) in magnitude as the probability $P^{\pi}(X, A \mid L = 6)$ increases (decreases). For example, the advantage of moving right *with* the key is much larger when the probability of having the key is 0.832 compared to when it is 0.999 (0.305 vs. 0.006). This difference is amplified by the increasing probability of taking the right action (0.5 vs. 0.9). In contrast, the corresponding Q-values remain constant at 1.

Similarly, the advantage of moving right *without* the key is smaller (in magnitude) when the probability of not having the key is 0.168 than when the probability is 0.000 (-0.695 vs. -0.994), even though the Q-value remains 0.[2]

The above is key because, as the policy improves, the probability of visiting state-action pairs such $\langle L = 6, X = 1, A = 1\rangle$ increases, and hence the training batches start to fill with many such pairs. Meanwhile,

[1] Q-values and advantages are computed using value iteration. $P^{\pi}(X \mid L = 6)$ is estimated by running the policy over multiple episodes and calculating the frequency of each state.

[2] Note that the magnitude of the advantage of moving left *with* the key first increases and then decreases across rows. This occurs because, although the probability of having the key increases, the probability of moving left decreases.

less frequent but informative pairs, such as $\langle L = 6, X = 0, A = 1\rangle$, become underrepresented. As a result, training on raw Q-values can lead the agent to disregard the key variable X, effectively exploiting a spurious correlation between location 6 and having the key. This occurs because updates to the policy treat all state-action pairs equally, regardless of how often they appear in the training data. Notably, even under a fully random policy (top row of Table 1), the probability of having the key at location 6 is already 0.832, meaning the agent often receives a reward of 1 simply by moving right at location 6, without explicitly reasoning about whether it has the key.

In contrast, agents trained with the advantage function are less likely to ignore the key variable. When the policy is random, the advantage of moving right *without* the key has a large negative value (-0.695), whereas the advantage of moving right *with* the key is just 0.305, despite the corresponding Q-values being 0 and 1 respectively. These differences in magnitude help counteract the overrepresentation of states such as $\langle L = 6, X = 1, A = 1\rangle$, and ensure that the agent does not develop the habit of moving solely based on location.

6 Experiments

The experiments aim to verify whether the insights discussed in the previous section hold in practice. Specifically, we seek to demonstrate that training on the advantage function, rather than raw Q-values, helps agents develop state representations that better capture causal factors and thus generalize out-of-trajectory. To test this, we conduct experiments on three gridworld environments: the Key2Door environment described in Section 2, as well as the Frozen T-Maze and Diversion environments introduced by Suau et al. (2024).

We evaluate the agent's performance in both the training environments and in modified versions, referred to as the evaluation environments, where, like in the Key2Door environment, the agent is forced to deviate from its usual trajectory. Furthermore, we analyze the effects of advantage normalization and batch size, which, as discussed in section 5.4, can influence out-of-trajectory generalization. Finally, we inspect the state representations learned by the agents by measuring the KL divergence of the policies between various state observations. Details about the T-Maze and Diversion environments are provided in Appendix C.

6.1 Experimental setup

Agents are trained using two different on-policy policy-gradient methods, REINFORCE (Williams, 1992) and PPO (Schulman et al., 2017) to maximize either the advantage or the Q-value. We implement policies and value functions as feedforward neural networks and use a stack of past observations as input in environments that require memory. The results are averaged over 10 random seeds. We report the average return as a function of the number of training steps. The shaded areas show the standard error of the mean. Training is interleaved with periodic evaluations in the original environments and their variants. Further details about the experimental setup are provided in Appendix D.

6.2 Results

Figure 2 shows the performance curves in all three environments for PPO. Agents trained using the advantage function (green) perform well in both the training and evaluation environments. In contrast, agents trained on the Q-value (magenta) perform poorly in the evaluation environments. Similar results using REINFORCE are reported in Appendix B.1.

The plot on the left of Figure 3 reveals how, as discussed in Section 5.4, normalizing the advantages removes their scaling effect and results in agents being unable to perform well on the evaluation environment. The plot on the right, on the other hand, shows how the performance of policies trained on the Q value improves as we increase the batch size, suggesting that the problem of state-action pair imbalance can sometimes be partly mitigated by using larger batch sizes. Results for the other two environments are provided in Appendices B.2 and B.3.

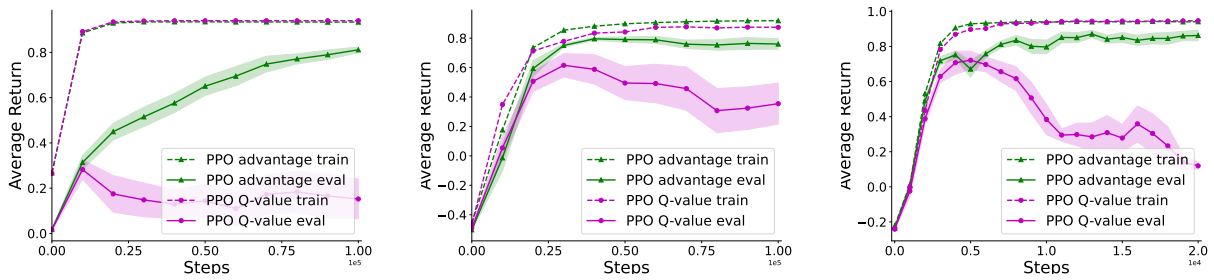

Figure 2: Performance of PPO using Q-values and Advantages in both the training and evaluation variants of the Key2Door (first plot), Frozen T-Maze (second plot), and Diversion (third plot).

The heatmaps in Figure 4 show the KL divergence of action probabilities between the agent having the key and not having the key at each of the six locations, measured at different training steps (top: 10k steps, middle: 50k steps, bottom: 100k steps).

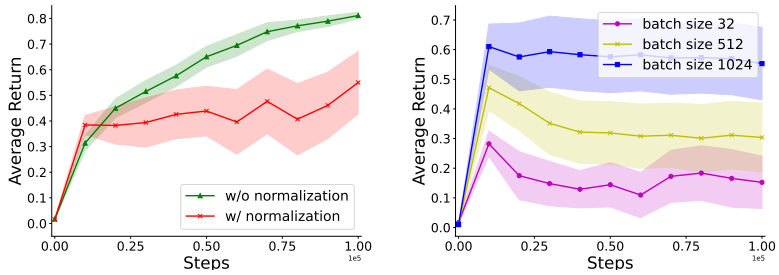

Figure 3: **Left:** Performance of PPO with and without advantage normalization in the Key2Door evaluation environment. **Right:** Performance of PPO with different batch sizes in the Key2Door evaluation environment.

The heatmap on the left corresponds to agents trained on the Q-value function, while the one on the right corresponds to agents trained on the advantage. Higher KL divergence indicates that the policy places more importance on the key feature. The results reveal that agents trained to optimize Q-value may learn policies that do not necessarily rely on the causal factors due to policy confounding, while policies trained on advantage function may be less affected by this phenomenon. Similar results for the Frozen T-Maze and Diversion environments are provided in Appendix B.4.

7 Limitations

Our analysis assumes discrete state and action spaces, although the results could be extended to continuous domains. In Section 5.1, when defining the advantage function under a given state representation Φ (Definition 7), we assume that the value function $Q(s_t, a_t)$ is estimated with respect to the full state, while $V^\pi(\Phi(s))$ is defined for the specific state represantion. This assumption is justified in settings where Q^π is at least partly estimated from Monte Carlo rollouts, as is common in many actor-critic methods. However, if Q were instead learned entirely via function approximation,

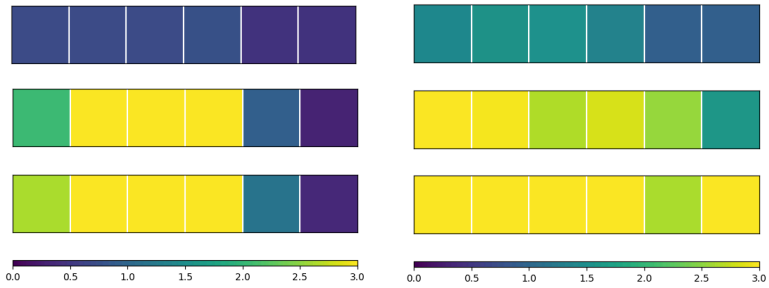

Figure 4: KL divergence of action probabilities with and without the key, measured at different training steps (top 10k steps, middle 50k steps, and bottom 100k steps) for agents trained on the Q-value (left) and the Advantage function (right).

then it too would be subject to Φ,
and Theorem 1 would no longer hold in its current form.

The experiments were conducted in the same three environments introduced by Suau et al. (2024), which were specifically designed to expose the phenomenon of policy confounding. While these environments are deliberately simple to facilitate analysis, this simplicity limits the generalizability of our findings. As such, we draw no conclusions about the effectiveness of the advantage function for learning causal representations in more complex or high-dimensional domains. Addressing this question would require further empirical investigation and is left for future work.

Finally, throughout the paper, we have taken care not to make strong claims about the effectiveness of the advantage function in learning causal state representations. Rather, we argue that the advantage function can mitigate policy confounding and support the formation of more causally grounded representations. However, its use provides no guarantees that the resulting representations will be truly causal.

8 Conclusion

In this paper, we analyzed the role of the advantage function in helping agents learn causal state representations. We showed that the advantage function scales the gradients by the complement of the probability of the corresponding state-action pair. This increases the magnitude of the gradients for state-action pairs that are less likely under the current policy while decreasing it for those that are more likely. As a result, it downweights the impact of state-action pairs that are overrepresented in training batches while amplifying the impact of those that are underrepresented. This helps break spurious correlations introduced by the policy, allowing agents to focus on the true causal factors. Section 5.5 provides a detailed numerical example illustrating this effect.

Our experiments on the Key2Door, Frozen T-Maze, and Diversion environments confirmed that training on advantages leads to more robust agents that generalize better out-of-trajectory. Furthermore, as explained in Section 5.4, our empirical results reveal how implementation choices, such as batch size and advantage normalization, affect the learned representations. Finally, the KL-divergence analysis of the action probabilities further demonstrates that using the advantage function makes agents more reliant on causal factors.

References

L. C. Baird. Reinforcement learning in continuous time: advantage updating. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, volume 4, pp. 2448–2453 vol.4, Jun 1994. doi: 10.1109/ICNN.1994.374604.

Bram Bakker. Reinforcement learning with long short-term memory. *Advances in neural information processing systems*, 14, 2001.

Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

Wesley Chung, Valentin Thomas, Marlos C Machado, and Nicolas Le Roux. Beyond variance reduction: Understanding the true impact of baselines on policy optimization. In *International Conference on Machine Learning*, pp. 1999–2009. PMLR, 2021.

Oriol Corcoll and Raul Vicente. Disentangling controlled effects for hierarchical reinforcement learning. In *Conference on Causal Learning and Reasoning*, pp. 178–200. PMLR, 2022.

Pim De Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *Advances in neural information processing systems*, 32, 2019.

Wenhao Ding, Laixi Shi, Yuejie Chi, and Ding Zhao. Seeing is not believing: Robust reinforcement learning against spurious correlation. *Advances in Neural Information Processing Systems*, 36:66328–66363, 2023.

Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *Foundations and Trends in Machine Learning*, 2018.

Evan Greensmith, Peter Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Advances in Neural Information Processing Systems*, 14, 2001.

Jinke He, Thomas M Moerland, Joery A de Vries, and Frans A Oliehoek. What model does muzero learn? *27th European Conference on Artificial Intelligence*, 2024.

Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *International Conference on Machine Learning*, pp. 1480–1490. PMLR, 2017.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76:201–264, 2023.

Li-Cheng Lan, Huan Zhang, and Cho-Jui Hsieh. Can agents run relay race with strangers? generalization of RL to out-of-distribution trajectories. In *The Eleventh International Conference on Learning Representations*, 2023.

Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Ajay Mandlekar, Yuke Zhu, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Adversarially robust policy learning: Active construction of physically-plausible perturbations. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3932–3939. IEEE, 2017.

Peter Marbach and John N Tsitsiklis. Simulation-based optimization of markov reward processes: Implementation issues. In *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304)*, volume 2, pp. 1769–1774. IEEE, 1999.

Andrew K. McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, 1995.

Jincheng Mei, Wesley Chung, Valentin Thomas, Bo Dai, Csaba Szepesvari, and Dale Schuurmans. The role of baselines in policy gradient optimization. *Advances in Neural Information Processing Systems*, 35: 17818–17830, 2022.

Hsiao-Ru Pan and Bernhard Schölkopf. Skill or luck? return decomposition via advantage functions. In *The Twelfth International Conference on Learning Representations*, 2024.

Hsiao-Ru Pan, Nico Gürtler, Alexander Neitz, and Bernhard Schölkopf. Direct advantage estimation. *Advances in Neural Information Processing Systems*, 35:11869–11880, 2022.

Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, 2016.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. In *International Conference on Machine Learning*, pp. 8787–8798. PMLR, 2021.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. Observational overfitting in reinforcement learning. In *International Conference on Learning Representations*, 2020.

Miguel Suau, Matthijs T. J. Spaan, and Frans A Oliehoek. Bad habits: Policy confounding and out-of-trajectory generalization in RL. In *Reinforcement Learning Conference*, 2024.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Gavin Taylor and Ronald Parr. Kernelized value function approximation for reinforcement learning. In *Proc. of the Twenty-Sixth International Conference on Machine learning*, pp. 128, 2009.

Guy Tennenholtz, Assaf Hallak, Gal Dalal, Shie Mannor, Gal Chechik, and Uri Shalit. On covariate shift of latent confounders in imitation and reinforcement learning. *arXiv preprint arXiv:2110.06539*, 2021.

Max Weltevrede, Caroline Horsch, Matthijs TJ Spaan, and Wendelin Böhmer. Training on more reachable tasks for generalisation in reinforcement learning. *arXiv preprint arXiv:2410.03565*, 2024.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

Amy Zhang, Clare Lyle, Shagun Sodhani, Angelos Filos, Marta Kwiatkowska, Joelle Pineau, Yarin Gal, and Doina Precup. Invariant causal prediction for block mdps. In *International Conference on Machine Learning*, pp. 11214–11224. PMLR, 2020a.

Junzhe Zhang, Daniel Kumor, and Elias Bareinboim. Causal imitation learning with unobserved confounders. *Advances in neural information processing systems*, 33:12263–12274, 2020b.

A Proofs

Theorem 1. *Let Φ be an arbitrary state representation. The advantage function can be expressed as:*

$$A^{\pi}_{\Phi}(s_t, a_t) = (1 - P^{\pi}(s_t, a_t \mid \Phi(s_t)))(Q^{\pi}(s_t, a_t) - \tilde{Q}^{\pi}(\neg\langle s_t, a_t\rangle)) \tag{5}$$

where

$$\tilde{Q}^{\pi}_{\Phi}(\neg\langle s_t, a_t\rangle) = \frac{\sum_{s'_t, a'_t \neq s_t, a_t} P^{\pi}(s'_t, a'_t \mid \Phi(s_t)) Q^{\pi}(s'_t, a'_t)}{\sum_{s'_t, a'_t \neq s_t, a_t} P^{\pi}(s'_t, a'_t \mid \Phi(s_t))} \tag{6}$$

Proof.

$$\begin{aligned}
A^{\pi}_{\Phi}(s_t, a_t) &= Q^{\pi}(s_t, a_t) - V^{\pi}(\Phi(s_t)) \\
&= Q^{\pi}(s_t, a_t) - \sum_{s'_t \in \{s_t\}^{\Phi}} P^{\pi}(s'_t \mid \Phi(s_t)) V^{\pi}(s'_t) \\
&= Q^{\pi}(s_t, a_t) - \sum_{s'_t \in \{s_t\}^{\Phi}} P^{\pi}(s'_t \mid \Phi(s_t)) \sum_{a' \in \mathcal{A}} \pi(a' \mid s'_t) Q^{\pi}(s'_t, a') \\
&= Q^{\pi}(s_t, a_t) - \sum_{s'_t \in \{s_t\}^{\Phi}, a'_t \in \mathcal{A}} P^{\pi}(s'_t, a'_t \mid \Phi(s_t)) Q^{\pi}(s'_t, a'_t) \\
&= (1 - P^{\pi}(s_t, a_t \mid \Phi(s_t))) Q^{\pi}(s_t, a_t) - \sum_{s'_t, a'_t \neq s_t, a_t} P^{\pi}(s'_t, a'_t \mid \Phi(s_t)) Q^{\pi}(s'_t, a'_t) \\
&= (1 - P^{\pi}(s_t, a_t \mid \Phi(s_t))) Q^{\pi}(s_t, a_t) - (1 - P^{\pi}(s_t, a_t \mid \Phi(s_t))) \tilde{Q}^{\pi}(\neg\langle s_t, a_t\rangle)) \\
&= (1 - P^{\pi}(s_t, a_t \mid \Phi(s_t)))(Q^{\pi}(s_t, a_t) - \tilde{Q}^{\pi}(\neg\langle s_t, a_t\rangle)).
\end{aligned} \tag{9}$$

since

$$\begin{aligned}
\tilde{Q}^{\pi}(\neg\langle s_t, a_t\rangle)) &= \frac{\sum_{s'_t, a'_t \neq s_t, a_t} P^{\pi}(s'_t, a'_t \mid \Phi(s_t)) Q^{\pi}(s'_t, a'_t)}{\sum_{s'_t, a'_t \neq s_t, a_t} P^{\pi}(s'_t, a'_t \mid \Phi(s_t))} \\
&= \frac{\sum_{s'_t, a'_t \neq s_t, a_t} P^{\pi}(s'_t, a'_t \mid \Phi(s_t)) Q^{\pi}(s'_t, a'_t)}{1 - P^{\pi}(s_t, a_t \mid \Phi(s_t))}.
\end{aligned} \tag{10}$$

□

B Experimental results

B.1 Results using REINFORCE

Figure 5 shows the performance using REINFORCE in the training and evaluation variants of the three environments.

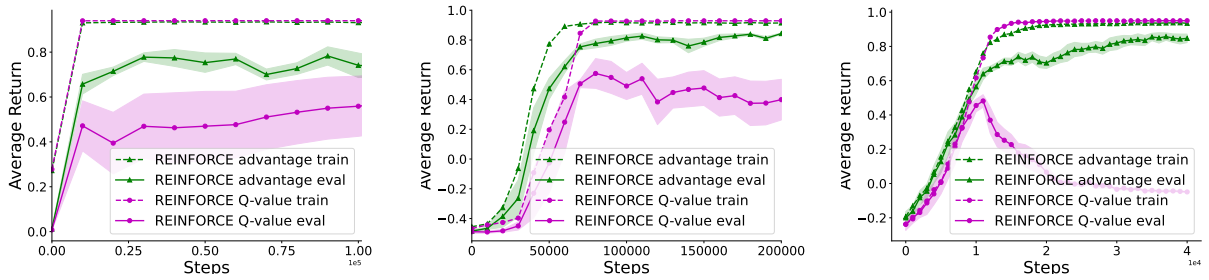

Figure 5: Performance of REINFORCE using Q-values and Advantages in both the training and evaluation variants of the Key2Door (first plot), Frozen T-Maze (second plot), and Diversion (third plot) environments.

B.2 Advantage normalization

Figure 6 compares the performance of PPO with and without advantage normalization in the evaluation variants of the three environments.

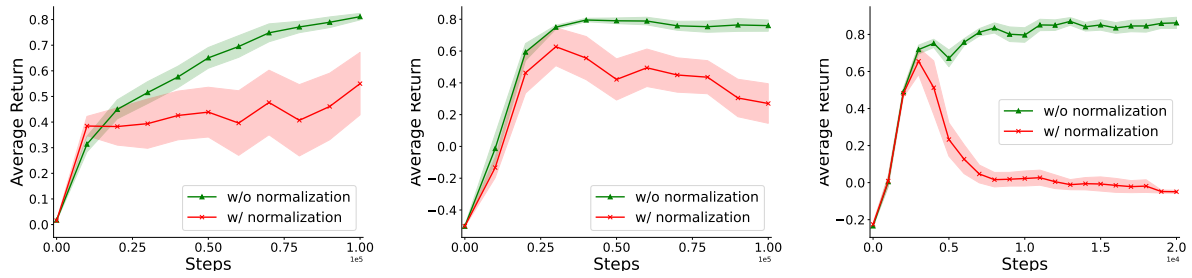

Figure 6: Performance of PPO with and without advantage normalization in the evaluation variants of the Key2Door (first plot), Frozen T-Maze (second plot), and Diversion (third plot) environments.

B.3 Batch size

Figure 7 compares the performance of PPO with different batch sizes in the evaluation variants of the three environments. While increasing the batch size seems to help in the Key2Door and Diversion environments, it has little effect in the Frozen T-Maze environment.

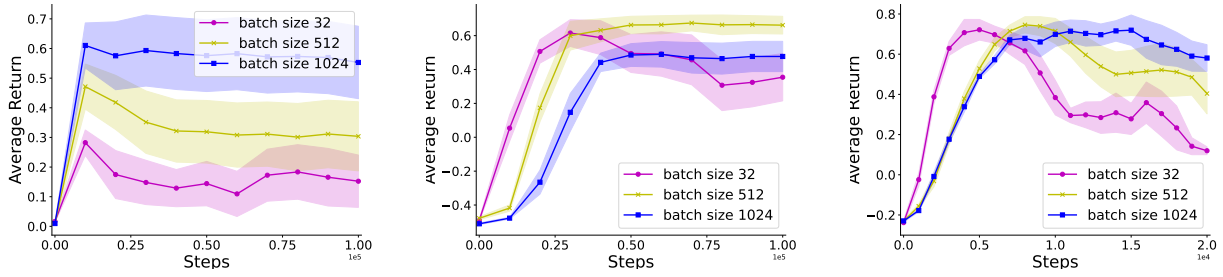

Figure 7: Performance of PPO with different batch sizes in the evaluation variants of the Key2Door (first plot), Frozen T-Maze (second plot), and Diversion (third plot) environments.

B.4 Policy KL divergence

Frozen T-Maze Figure 8 shows the KL divergence of action probabilities in the Frozen T-Maze environment, produced by the policy when the signal is either purple or green, measured at different training steps (top: 10k steps, middle: 50k steps, bottom: 100k steps). To compute these divergences, we take the observation stack received by the agent and query the policy network twice: once with the original stack, and once with the signal bit flipped to the opposite value. The KL divergence at each cell is the average over 100 evaluation episodes.

The left heatmaps reveal that after 100k training steps, the agent trained on the Q-value largely ignores the signal, except at the starting location. In contrast, the agent trained on the advantage function conditions its action choices on the signal value throughout the maze.

Interestingly, the left heatmaps also reveal that as training progresses, the trajectories followed by the Q-value-trained agent become increasingly deterministic. By the end of training, the agent consistently chooses the top path when the signal is green and the bottom path when the signal is purple. In contrast, the agent trained on the advantage function continues to follow a diverse set of trajectories and does not exhibit a strong preference for any particular path. Note that multiple optimal paths exist for each signal value.

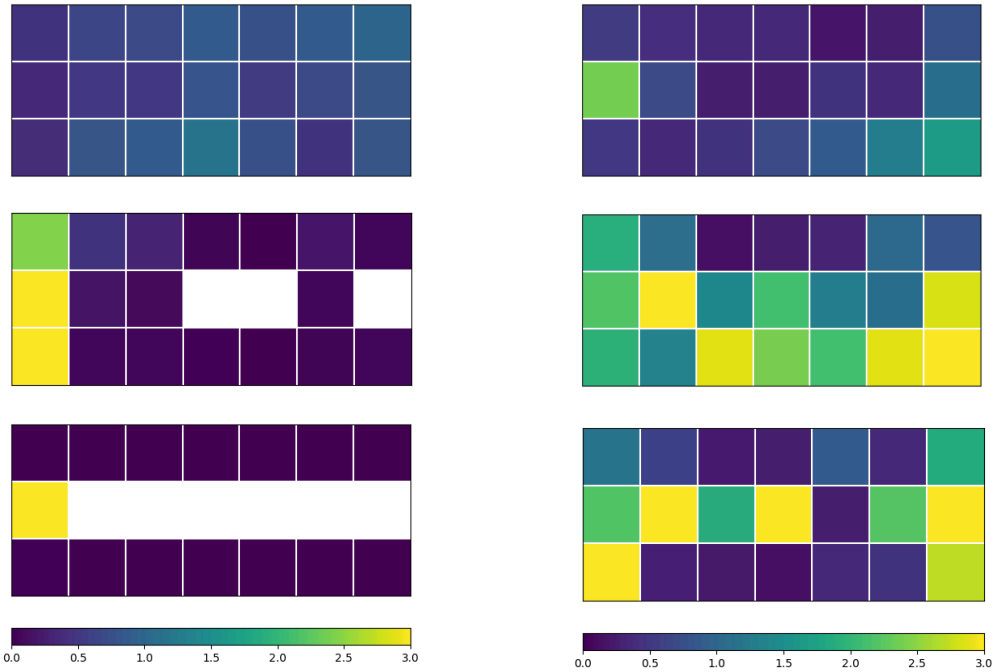

Figure 8: KL divergence of action probabilities in the Frozen T-Maze environment at different training steps (top 10k steps, middle 50k steps, and bottom 100k steps) for agents trained on the Q-value (left) and the Advantage function (right).

Diversion Figure 9 shows the policy KL divergence across each column in the Diversion environment, measured at different training steps (top: 3k steps, middle: 10k steps, bottom: 20k steps). For each column, the divergence is computed by comparing the action probabilities output by the agent's policy when positioned in the top versus bottom row.

The left heatmaps indicate that the agent trained on the Q-value learns to ignore the bit indicating the row after 10k training steps. In contrast, the agent trained on the advantage function continues to use the row information when deciding which action to take.

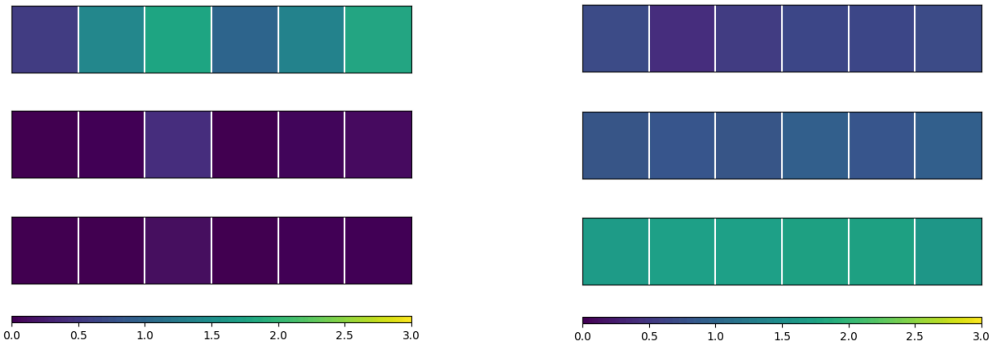

Figure 9: KL divergence of action probabilities in the Diversion environment measured at different training steps (top 3k steps, middle 10k steps, and bottom 20k steps) for agents trained on the Q-value (left) and the Advantage function (right).

C Environments

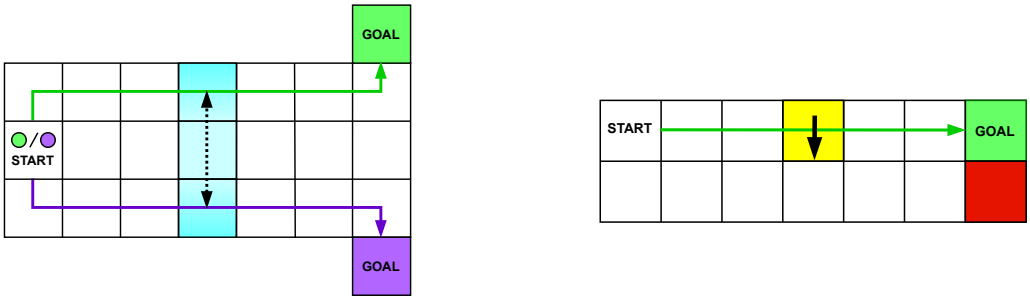

Figure 10: Illustrations of the Frozen T-Maze (left) and Diversion (right) environments.

Frozen T-Maze This environment is a variant of the popular T-Maze setting (Bakker, 2001). At the starting location, the agent receives a binary signal: either green or purple. Its task is to navigate to the right and reach the correct goal at the end of the maze. The agent receives a reward of $+1$ for reaching the green (purple) goal when the green (purple) signal was observed, and a penalty of -1 otherwise. Additionally, a time penalty of -0.01 is applied at each timestep to encourage the agent to take the most direct path to the goal.

At every timestep, the agent observes its current location within the maze, represented by a one-hot-encoded vector. However, the initial signal, represented by a binary variable, is only provided to the agent at the starting location. Crucially, the agent is capable of remembering past observations. When moving randomly, it must retain the initial signal throughout its trajectory. However, once it learns the shortest path to each goal (illustrated by the green and purple arrows), the agent can safely disregard the initial signal. This is because the agent can infer the signal based on its location: if it is on the green (purple) path, it must have received the green (purple) signal. Note that the two paths highlighted in Figure 10 are not the only optimal ones. However, for the agent to disregard the initial signal, the paths must not overlap.

We train agents in the original environment and evaluate them in a modified version where an icy surface (shown in blue) is introduced in the middle of the maze. This ice causes the agent to slip between the upper and lower cells.

Diversion In this environment, the agent must move from the start state to the goal state shown in Figure 10 (right). A reward of $+1$ is given for reaching the goal, and a penalty of -1 is incurred if the agent reaches the red cell instead. As in the other environments, there is a per-timestep penalty of -0.01. Observations are 8-dimensional binary vectors: the first 7 elements indicate the column where the agent is located, and the last element indicates the row.

After the agent learns the optimal policy (shown by the green arrow), it can ignore the last element of the observation vector. This is because the optimal policy never visits the bottom row. We train the agent in the original environment and evaluate it in a modified version containing a yellow diversion sign in the middle of the maze, which forces the agent to move to the bottom row.

D Experimental setup

The experiments were run on a laptop equipped with an Apple M2 Pro processor (12 cores) and 16 GB of RAM. Each run took less than 5 minutes and used at most 2% of the total RAM.

Agents were trained using Stable Baselines3 (Raffin et al., 2021). The hyperparameters are listed in Tables 2 (PPO) and 3 (REINFORCE). For PPO, we adopted the hyperparameters used by Schulman et al. (2017) in their Atari experiments, except for the learning rate, which we increased to 1.0e-3 to accelerate convergence. We implemented a minimal version of the REINFORCE algorithm with only three hyperparameters (learning rate, discount factor, and entropy coefficient), for which we used the same values as in PPO. For the Frozen T-Maze, we used a stack of the past 30 observations as input, since solving the task requires memory.

Table 2: PPO hyperparameters.

Rollout steps	128
Batch size	32
Learning rate	1.0e-3
Number epoch	3
Discount γ	0.99
GAE λ	0.95
Entropy coefficient	1.0e-2
Clip range	0.1
Value coefficient	1
Number Neurons 1st layer	128
Number Neurons 2nd layer	128

Table 3: REINFORCE hyperparameters.

Learning rate	1.0e-3
Discount γ	0.99
Entropy coefficient	1.0e-2
Number Neurons 1st layer	128
Number Neurons 2nd layer	128