RESOLVING CAUSAL CONFUSION IN REINFORCEMENT LEARNING VIA ROBUST EXPLORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A reinforcement learning agent must distinguish between spurious correlations and causal relationships in its environment in order to robustly achieve its goals. Contrary to popular belief, such cases of causal confusion *can* occur in online reinforcement learning (RL) settings. We demonstrate this, and show how causal confusion can lead to catastrophic failure under even mild forms of distribution shift. We formalize the problem of identifying causal structure in a Markov Decision Process, and highlight the central role played by the data collection policy in identifying and avoiding spurious correlations. We find that under insufficient exploration, many RL algorithms, including those with PAC-MDP guarantees, fall prey to causal confusion under insufficient exploration policies. To address this, we present a robust exploration strategy which enables causal hypothesis-testing by interaction with the environment. Our method outperforms existing state-of-the-art approaches at avoiding causal confusion, improving robustness and generalization in a range of tasks.

1 INTRODUCTION

Models that depend on spurious correlations in their training data are vulnerable to catastrophic failure under even mild forms of distribution shift (de Haan et al., 2019). This has motivated the search for *causal* approaches in machine learning which enable systems to identify relationships in the environment that will remain invariant under changes to the data-generating distribution. The reinforcement learning (RL) setting provides the opportunity for an agent attempting to learn cause from effect: unlike supervised learning from observational data, the agent can interact with its environment to test hypotheses as part of the learning process. In this sense, RL agents have the capacity to perform hypothesis testing as part of the learning process, and to some extent it can be argued that algorithms like policy iteration implicitly do this already by acting so as to maximize the predicted reward in the environment.

However, the extent to which standard RL algorithms allow an agent to infer causal structure in its environment remains up for debate. Fedus et al. (2020) provide several examples of catastrophic failure when generalizing to unseen states, suggesting that RL algorithms are prone to memorizing their training data. Failure to generalize is a particular concern in the offline and imitation learning settings. Indeed, de Haan et al. (2019) show that agents trained by imitation learning can learn policies that depend on spurious correlations in their training data, correlations that are not present under the state visitation distribution of the learned behaviour policy. For example, a self-driving car policy can learn to attend only to the brake lights of a car in front of it, rather than a red traffic light, if the training data is not diverse enough. We refer to this phenomenon as *causal confusion*. To the extent that generalization in RL can be thought of as a subset of causal structure identification, this highlights the necessity of explicitly taking causal relationships into account to obtain algorithms that can successfully generalize to new environments and observations.

Existing works at the intersection of RL and causality have focused principally on the imitation learning setting (Zhang et al., 2020b; de Haan et al., 2019), batch learning setting (Bannon et al., 2020), and partial observability setting (Kallus and Zhou, 2018; Forney et al., 2017). Contrary to popular belief, we show that causal confusion can also occur in the online RL setting. This is apparent in the limited data regime, but can also occur with unlimited data in a multi-environment

setting. We believe we are the first to pinpoint and address this issue in model-free algorithms trained in the online, fully observable setting.

This observation leads to the motivating question of this paper: how can we design RL algorithms that allow agents to identify sources of spurious correlations in their environment? To answer this question, we first characterize what it means to learn the relevant causal structure of the environment, and under what conditions an RL agent may fail to identify this structure. Our analysis highlights the central role played by the agent's data collection policy in identifying and avoiding spurious correlations. This motivates our second contribution: a novel perspective on robust exploration for control with theory-backed guarantees. Our method can be straightforwardly incorporated into existing training procedures to improve the robustness of an agent's value function to certain forms of distribution shift. We demonstrate that this method avoids the pitfalls of existing approaches in identifying sources of causal confusion in a range of environments, and improves on the state of the art in generalizing to new levels in the ProcGen suite (Cobbe et al., 2019).

2 Background & Problem Setting

We begin by introducing core concepts from reinforcement learning and causal inference (CI) and formulating the problem setting that this work addresses.

2.1 Background

Reinforcement Learning. The reinforcement learning problem consists of an agent acting in an environment so as to maximize its cumulative reward (Sutton and Barto, 2018). We formalize the environment as a *finite* Markov Decision Process (MDP), defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma, p_0 \rangle$, where $\mathcal{S}$ denotes the states, $\mathcal{A}$ the action set, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ a reward function, $\mathcal{P}$ a transition kernel, γ a discount factor, and $p_0 \in \mathcal{P}(\mathcal{S})$ is an initial state distribution. In a fully observable MDP, the agent has direct access to the states s. In a partially observable MDP (POMDP), the MDP is augmented with a stochastic observation function $o : \mathcal{S} \to \mathbb{P}(\mathcal{O})$ which maps the state space to a distribution over an observation space $\mathcal{O}$, and the agent has access only to these observations. RL agents interact with the environment by taking actions. A policy $\pi : \mathcal{S} \to \mathcal{A}$ determines the action taken by an agent as a function of the state. The value of a state-action pair under a policy π is defined as the expected sum of discounted rewards

$$Q^\pi(s, a) = \mathbb{E}_{(s_t, a_t) \sim P^\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0, a_0 = s, a] \ . \tag{1}$$

Causal discovery concerns itself with identifying causal relationships from data (Pearl, 2009). The central object of study is a Structural Causal Model (SCM), which characterizes the data generating distribution as a set of functions on observed and hidden variables.

Definition 1 ((Pearl, 2009))**.** *A SCM is a tuple* $(\mathbf{U}, \mathbf{V}, \mathcal{F}, P)$*, where* $\mathbf{U}$ *is a set of exogenous variables (e.g. the unobserved source of stochasticity in the environment) drawn from the distribution* p*,* $\mathbf{V}$ *is a set of endogenous variables (e.g. the observed state* s*, the reward* r*, and the action* a *in RL), and* $\mathcal{F}$ *is the set of functions* $f_V : Pa_V \times \mathbf{U} \to V$ *with* $Pa_V \subset \mathbf{V}$*, which determine the value of endogenous variable* V *for each* $V \in \mathbf{V}$*.*

SCMs are best visualized via a causal graph, a directed acyclic graph whose nodes are variables in the SCM and whose arrows indicate causal relationships. We provide an example of a causal graph corresponding to an MDP in Figure 1. Causal models enable reasoning about how changes to the data-generating process, called *interventions*, affect the resulting distribution over variables.

Definition 2 (Do-Intervention)**.** *A do-intervention on a variable* V *do(*$V = v$*) in a causal model* $S = (\mathbf{U}, \mathbf{V}, \mathcal{F}, P)$ *induces a new SCM* $S' = (\mathbf{U}, \mathbf{V}, \mathcal{F}', P)$*, where* $\mathcal{F}' = \{f_W \in \mathcal{F} | W \neq V\} \cup \{f_{V=v}\}$ *and* $f_{V=v}(\mathbf{p}, \mathbf{u}) = v \forall \mathbf{p} \in Pa_V, \mathbf{u} \in U$*.*

Predicting the effect of a do-intervention requires identifying the direction of the edges in the causal graph, a process known as *causal discovery*. In many prediction problems, it is desirable to design predictors which depend only on causal ancestors of the target, as these predictors will generally be more robust to distribution shift. Such predictors are called *causal predictors*, and the phenomenon of a predictor depending on a causal descendant of its target is known as *causal confusion*.

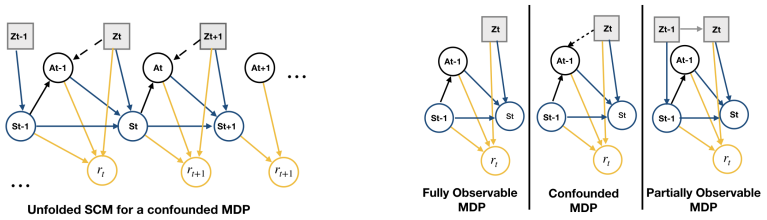

Figure 1: Visualizations of the structural causal model for an MDP under different sets of assumptions.

2.2 Problem Setting

Various works identify the problem of causal confusion in the imitation learning, batch learning, and partial observability settings when spurious correlations are present in the demonstration data that disappear in the online environment (Zhang et al., 2020b; de Haan et al., 2019; Bannon et al., 2020; Kallus and Zhou, 2018; Forney et al., 2017). Nominally, it appears that these issues disappear when we focus on the online setting with full observability, where there are no latent confounders. However, causal confusion can still arise in two scenarios: the limited data regime of the single environment setting, and in both limited and unlimited data regimes of the multi-environment setting. We explore each of these settings in this work, using the following formalism. Given a set of training environments $\mathbf{E}_{\text{train}} = \{\mathcal{E}_i | i \in 1, \dots N_{\text{train}}\}$ and a set of testing environments $\mathbf{E}_{\text{test}} = \{\mathcal{E}_i | i = 1, \dots, N_{\text{test}}\}$, we seek to find a function $\hat{Q} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ which minimizes the worst-case approximation error to the optimal value function Q^* over all environments.

$$\min_{\hat{Q}} \max_{E \in \mathbf{E}_{\text{test}}} \mathbb{E}_{s_0 \sim E}[\max_a |Q^*(s_0, a) - \hat{Q}(s_0, a)|] \ . \tag{2}$$

Limited data in the single-environment setting. In the language of Equation (2), the *single-environment* formulation of this problem sets the test environments to be identical to the singleton training environment, with the exception of their initial-state distribution. Causal confusion can occur when the agent has a limited number of interactions with the environment. As established in Peters et al. (2016), if the agent does not explore sufficiently to intervene on each node of the underlying SCM of the environment, it can learn spurious correlations that do not generalize to *distribution shifts* at test-time, or changes from the data sampling distribution at train-time.

The multi-environment setting. In the multi-environment setting, we are limited by the number of environments available at training time. We are specifically interested in the setting where there are many, potentially infinitely many, tasks that fall under a family of MDPs with consistent causal dynamics. Procedurally generated environments are an example of this scenario, where an agent can have an unlimited amount of data from each task, but still incorrectly generalize to new unseen environments from the same family (Cobbe et al., 2019). However, we assume the test environments are *consistent* with the training environments, i.e. whenever an observation o exists in both the training and test environments, it has the same Q-value for every action a.

3 Causal RL

RL and causal inference both concern themselves with estimating the effect of an agent's interaction with the world, but differ significantly in how they model this interaction. In CI, the agent interacts with the environment by intervening on the values of variables and then observing the effect of this intervention on the distribution. In RL, the agent interacts with the environment by selecting actions. This section provides a framework for expressing causal queries in the reinforcement learning setting, culminating in characterising the objective of Equation (2) as a causal prediction problem.

3.1 Causal Language in RL

Structural causal models and MDPs. We imbue causal semantics to the standard formulation of an MDP as a Bayesian network (Kappen et al., 2012), and consider an SCM on variables $S_t, A_t, Z_t, r_t, S_{t+1}$, where Z_t is a set of hidden or noise variables responsible for the stochasticity in the environment (for example, the random number generator in a simulator) as shown in the example SCM in Figure 1. While the conditional distributions of S_t, Z_t, r_t, S_{t+1} are given by the MDP transition dynamics, the distribution of A_t is defined by the agent's policy π. Given an MDP $\mathcal{M}$ and

a policy π, we obtain an SCM $\mathcal{G}^\pi$, on which we can cast policy evaluation as one of two inference problems.

$$Q^\pi(s,a) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\mathcal{G}^\pi}[R_t | S_0 = s, A_0 = a] \qquad Q^\pi(s, \text{do}(a)) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\mathcal{G}^\pi}[R_t | S_0 = s, \text{do}(A_0 = a)]$$

When π is stationary we obtain $Q^\pi(s,a) = Q^\pi(s, \text{do}(a))$, though in partially observable environments or under confounding this equality will not in general hold. While this might intuitively suggest that naive value-function approximation methods are sufficient to eliminate spurious correlations, the robustness of these methods depends heavily on the coverage of the data-generating policy used to train them. We provide an example to demonstrate this in Appendix A, to which we will also defer proofs of the theoretical results to come.

We define interventions on an MDP to be sequences of actions or policies that allow the agent to manipulate properties of the state. We will be principally interested in the effect of specific policies on the MDP, rather than fixed sequences of actions. For example, we might be interested in evaluating the effect of a policy that takes a robot to a specific room in a house, rather than a policy which takes ten steps forward. To do so, we leverage the options framework introduced by Sutton et al. (1999). While this does not exactly match onto the do-intervention definition given previously, it still suitably captures the concept of agent influence on the environment. We provide a more detailed discussion of the semantics of interventions in an MDP is in Appendix B.

Definition 3 (Intervention on an MDP)**.** *An intervention on an MDP is a sequence of action-selection rules $\pi_1, \ldots, \pi_k$. When $\pi_1 = \cdots = \pi_k = \pi$, this is a stationary intervention policy and simply write π. Let $\mathbf{S} \subseteq \mathcal{S}$, $\mathcal{S}_0$ denote the support of the initial state distribution, and $\pi_{\mathbf{S}}$ denote the optimal goal-reaching policy for $\mathbf{S}$, i.e. the policy which minimizes the expected number of steps to reach $\mathbf{S}$ from any state s. Then we define a targeting intervention as the option $(\mathcal{S}_0, \pi_{\mathbf{S}}, \delta_{\mathbf{S}})$.*

Having constructed an analogue of do-interventions, we now consider an analogue of causal predictors, which we will call *causally correct value functions*.

Definition 4 (Causal correctness)**.** *A value function Q on an MDP $\mathcal{M}$ is causally correct if for any intervention $\pi_{\mathbf{S}}$, $\mathbb{E}_{S \sim \pi_{\mathbf{S}}}[\max_a |Q(S,a) - Q^*(S,a)|] < \epsilon$. A value function Q on a* family *of MDPs is causally correct if the above holds for every environment $\mathcal{M} \in \mathcal{E}_{test}$.*

Equation 2 closely resembles the characterization of a causal predictor proposed by Peters et al. (2016), which minimizes the worst-case prediction error over the set of possible interventions on the data-generating distribution.

Theorem 1. *Let $\mathbf{E}_{train} = \{\mathcal{M}\}$ for $\mathcal{M} = (\mathcal{S}, A, R, P, \gamma, p_0)$, and $\mathbf{E}_{test} = \{(\mathcal{S}, A, R, P, \gamma, p_S) | S \subseteq \mathcal{S}\}$. Let $\Pi_S = \{\pi_{\{s\}} | s \in \mathcal{S}\}$. Then if $\mathcal{M}$ is fully connected*

$$\min_Q \max_{\mathbf{E}_{test}} \mathbb{E}[\max_a |Q^*(s_0,a) - Q(s_0,a)|] = \min_Q \max_{\pi_{\mathbf{S}} \in \Pi_S} \mathbb{E}_{s_0 \sim \pi_{\mathbf{S}}}[\max_a |Q^*(s_0,a) - Q(s_0,a)|].$$

3.2 Optimality and Exploration

The notion of causal correctness described previously requires that the learned Q-function be approximately equal to Q^* for all state-action pairs. This requires a different approach to exploration from that required by the regret minimization framework, where inaccuracy on states that are not visited by the optimal value function is not penalized. As a result, it is possible to construct examples where a PAC-MDP algorithm, which obtains provable guarantees with respect to a single environment, fails to satisfy our robustness criterion. We provide one such example in Appendix A.

Effective exploration is therefore crucial to identify sources of spurious correlations.

Borrowing from the language of causal effect inference, we observe that an exploration strategy which induces two criteria in the data it generates will guarantee *identifiability* of the optimal value function.

The following theorem is illustrative of how a notion of *targeting* states to obtain sufficient coverage followed by *executing* the policy whose value we seek to approximate can be used to collect data that guarantees convergence to the correct value function in the linear approximation setting.

Theorem 2. *Let $\mathbf{X}_t$, $\mathbf{X}'_t \in \mathbb{R}^{t \times d}$ with k^{th} rows $\mathbf{x}_k$, $\mathbf{x}'_k$ respectively, and $\mathbf{r}_t \in \mathbb{R}^t$ denote a collection of sampled transitions from an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, R, P, \gamma)$. Assume each $\mathbf{x}_k$ is sampled from some*

distribution $\eta(\mathcal{X})$ and $\mathbf{x}'_k$ is obtained by sampling an action from the policy $\pi(\mathbf{x})$ and observing the induced transition in the MDP. Assume the realizable setting: i.e. there exists some $\mathbf{w}_T \in \mathbb{R}^d$ such that $V^\pi(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w}_T \rangle$. Then if the sampling distribution η is such that $\mathbb{E}_{\mathbf{x}\sim\eta}[(\mathbf{x}^\top v)^2] > 0$ for all unit vectors $\mathbf{v}$, we will have

$$\mathbf{w}_t = [\mathbf{X}_t(\mathbf{X}_t - \gamma \mathbf{X}'_t)^\top]^{-1}[\mathbf{X}_t \mathbf{r}_t] \overset{t\to\infty}{\longrightarrow} \mathbf{w}_T \tag{3}$$

Although the conditions required to learn a robust policy are satisfied in the limit of infinite training time by ϵ-greedy exploration in a fully observable MDP, this approach can lead to sub-optimal policies in finite time. The next section presents a robust exploration strategy to efficiently gather the necessary data to identify cause and effect in the environment.

4 Robust Exploration to Combat Causal Confusion

Having established the importance of a sufficiently broad state coverage distribution to learn a causally correct value function, we now turn our attention to the implementation of an exploration strategy which can provide this data. Our goal is to construct a relatively generic strategy that can be slotted into existing algorithms to improve the coverage of the data-generating policy, improving robustness to causal confusion at minimal data cost.

4.1 Target-then-Execute

Whereas in the standard RL framework we evaluate policies by measuring their return, this approach may fail to identify the robust policies sought by our problem setting, as a policy may not visit states at which it attains low value. Recalling the autonomous vehicle example, a policy which stops at red lights and a policy that stops when the car in front of it stops will appear identical if evaluated only when the agent is driving behind another car. To address this failing, we present a straightforward exploration strategy which aims to efficiently guarantee the conditions put forth in Section 3.2 consisting of two distinct steps: a target step and an execution step.

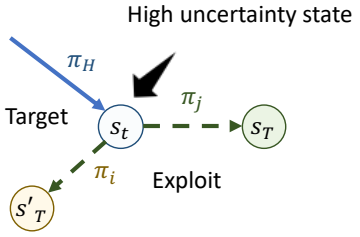

Figure 2: Target-then-execute strategy. s_t is a state of high uncertainty, which our exploration agent π_H targets. From s_t, we can now exploit with any ensemble member such as π_i or π_j.

As shown in Figure 2, the **target** step identifies states which have not been visited frequently or about which the agent is uncertain, and follows a policy which aims to maximize the probability of visiting these states. Once such a state is reached, the **execute** step follows the hypothesized optimal policy for the remainder of the episode. In practice, to compute uncertainty we use an ensemble of exploiting agents and measure uncertainty as the normalized variance over those agents (Lu and Van Roy, 2017). The data collected may then be used to train an optimal value function in the online setting, or to 'score' the ensemble member in the setting of model selection on agents trained offline.

Algorithm 1: Target-then-execute exploration for the single-environment setting.

Initialize value functions $Q_1, \ldots, Q_K, Q_H$, replay buffers $D_1, \ldots, D_K$;
while *Forever* **do**
  $s_t \sim P(s_0)$, $\pi \leftarrow \pi_H$, $a_t \sim \pi(a|s_0)$, $i \sim \{1, \ldots, K\}$, stop $\leftarrow$ False;
  while *not stop* **do**
    stop, $s \sim T(s_t, a_t)$, $D_i = D_i \cup \{(s_t, a_t, s)\}$;
    `Update`$(Q_1, \ldots Q_k)$; `Update`(Q_H) with reward $\frac{\sigma^2(\{Q_1,\ldots,Q_k\})}{\mu(\{Q_1,\ldots,Q_k\})}$;
    if $Q_H(s) < \epsilon$ **then** $\pi \leftarrow \pi_i$;
    $s_t \leftarrow s$, $a_t \leftarrow \pi(s)$;

This generic exploration strategy can be used in conjunction with a number of different learning algorithms for training the ensemble members, for both learning and model selection problems.

We show in Section 5 that our approach outperforms existing exploration strategies to identify and overcome causal confusion in environments where informative states may be difficult to reach. Our implementation of Algorithm 1 uses an ensemble of Q-functions to model uncertainty, and samples from this ensemble to perform execution. We use the normalized empirical variance of the predicted Q-values as the reward for the targeting policy, which acts greedily with respect to this value function. Finally, we parameterize the method with a threshold ϵ and switch between the targeting and execution policies when the uncertainty bonus exceeds this threshold. We provide additional implementation details in Appendix C.

4.2 Theoretical Motivation

To motivate the target-then-execute approach, we consider an idealized learning algorithm acting in an *episodic* MDP which has access to an expert policy oracle, along with an expert goal-reaching oracle. The agent deploys the goal-reaching oracle to reach a state s, then the expert policy generates the trajectory $\tau = ((s_0 = s, a_0, r_0), \ldots, (s_n, a_n, r_n))$. It then updates its estimate of $Q^*(s, \cdot)$ based on the Monte Carlo return from this trajectory. We demonstrate in Appendix A that this procedure is sufficient to give unbiased estimates of the value of any state-action pair.

Rather than a point estimator of $Q^*(s, a)$, we will be interested in a posterior $P(Q(s, a))$ which is updated based on the Monte Carlo return obtained by following the optimal policy oracle from a given state-action pair. We assume the posterior is computed from a Gaussian prior distribution with diagonal covariance, along with a Gaussian likelihood with fixed noise variance σ^2. At the start of each episode, we set the target state and action s^*, a^* to be the maximizer of $H_P[Q(s^*, a^*)]$, i.e. the entropy of the posterior over Q-values, and follow the intervention policy π_{s^*} until s^* is reached, at which point the action optimal policy oracle π^* is invoked. The resulting trajectory τ is truncated to start from the first visitation of (s^*, a^*), and the Monte Carlo return from this trajectory is used to update the posterior P. Under this setting, it is easy to show that an analogue of the target-then-execute policy provides the maximum *information gain* from any trajectory. We define the information gain (Houlsby et al., 2011) of an expert oracle query from a state-action pair (s, a) as follows:

$$IG(P, s, a) = \mathbb{E}_{\tau \sim (\pi^*, s, a)} H_P(Q|\mathcal{D}) - \mathbb{E}[H_P(Q|\mathcal{D}, \tau)] \,. \tag{4}$$

Theorem 3. *Let s^*, a^* be a state action pair selected by target-then-execute at episode h. Let P_h denote the posterior over Q-functions at episode h obtained by a Bayesian model with Gaussian prior and likelihood. Then*

$$(s^*, a^*) \in \textit{arg max}_{s,a} IG(P_h, s, a) \,. \tag{5}$$

5 Experiments

In this section, we show that the target-then-execute exploration strategy is robustly useful across a range of tasks. First, we demonstrate that target-then-execute resolves causal confusion in a setting where the approach of de Haan et al. (2019) fails. We then show that this approach accelerates learning in online RL agents faced with hard exploration problems, outperforming Thompson sampling and epsilon-greedy exploration in a sparse-reward gridworld. Finally, we demonstrate that our approach scales to rich observations in the multi-environment setting. Additional details for all experiment settings are provided in Appendix C.

5.1 Resolving Causal Confusion

We now show that target-then-execute exploration can effectively collect data which allows the agent to identify sources of causal confusion in its policy. We first consider the confounded imitation learning setting of de Haan et al. (2019), whose approach to eliminating causal confusion presents the state of the art: using the OpenAI Gym (Brockman et al., 2016) MountainCar implementation, we modify the 2-dimensional state consisting of position x and velocity $\dot{x}$ to also include the previous action taken. Because the optimal policy takes several identical actions consecutively to solve the task, this additional input is highly predictive of an expert's next action.

We then train an imitation learner on trajectories generated by an expert, following the procedure of de Haan et al. (2019). The imitation policy π takes as input the state $(x, \dot{x}, a)$, along with a mask ϕ_s with $s \subset \{x, \dot{x}, a\}$ which sets the variables in s to zero, giving s s an interpretation as

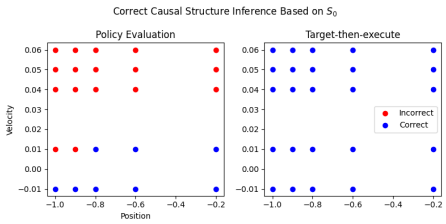

Figure 3: Causal correctness of the state abstraction inferred by baseline (left) and our (right) exploration strategy from a fixed initial state.

Figure 4: Value function learned in a multi-objective gridworld. Target-then-execute is the only exploration strategy which correctly identifies the value function over the entire state space.

a candidate causal graph. To identify the optimal mask ϕ_s, the policy $\pi(\cdot, \phi_s)$ is evaluated on the environment for each subset s, and the Monte Carlo return collected during evaluation is used to update the agent's posterior belief that a particular variable is a causal parent of the optimal policy. We say that a posterior is 'correct' if it assigns lower probability to a being a causal parent of the next action than $\cdot x$. We visualize the correctness of the inferred feature set as a function of initial state in Figure 3, demonstrating that our approach avoids the pitfalls of the existing state-of-the-art in environments with uninformative initialization states.

5.2 Single-Environment Generalization

Efficient exploration from offline data. Recall the autonomous driving setting discussed previously in which an agent frequently drives through a rail crossing on a busy road, but whose training data contains no or very few examples of the car directly in front of the rail crossing when the light is red. We simulate this example using a rich-observation gridworld based on the Gym-Minigrid environment (Chevalier-Boisvert et al., 2018). See **??** for a visualization. We train an ensemble of value functions on training data collected from an optimal policy offline, and then deploy the agent in the evaluation environment to collect additional data to fine-tune this learned policy. We measure the quality of the agent's exploration by the number of times that its exploration strategy takes it to a red light in **??**. We compare target-then-execute against approximate posterior sampling (PSRL), which randomly samples an agent from the ensemble and follows the agent's greedy policy, and ϵ-greedy exploration, two approaches that are widely used in practice and which require little hyper-parameter tuning. We find that our approach outperforms both baselines in finding these informative training examples.

Identifying a robust value function. We then consider a gridworld with two goal states (Figure 4) to evaluate whether our approach is able to identify the optimal policy over the entire state space. In this environment, the optimal policy depends on the distance to the nearest goal from the state the agent is currently in. We find that the two baseline methods, ϵ-greedy exploration with a random initialisation and optimism in the face of uncertainty, identify the correct value function for one of the goal states, but learn a policy which ignores the other goal state. In contrast, target-then-execute identifies both goal states and learns a policy which can effectively reach the nearest goal from any state in the environment, i.e. a policy which is robust to changes in the initial state distribution.

5.3 Scaling to Rich Observations

We now address the multi-environment setting as specified in Section 2.2. OpenAI ProcGen (Cobbe et al., 2019) was proposed as a way to test generalization capabilities of RL agents, where agents train on a set of levels and are tested on a held out set of levels. All levels obey the same underlying SCM. We show that we can use our method to improve generalization to these unseen levels when the agent can choose which level it sees next. This setting was initially introduced in Jiang et al. (2021) and exploited with a form of automatic curriculum generation using L1 value loss as a scoring function. We show that performance can be improved with our target-then-execute strategy in this multi-environment setting. In ProcGen, we do not need a parameterized exploration agent because we are in the bandit setting where any level can be reached in a single step.

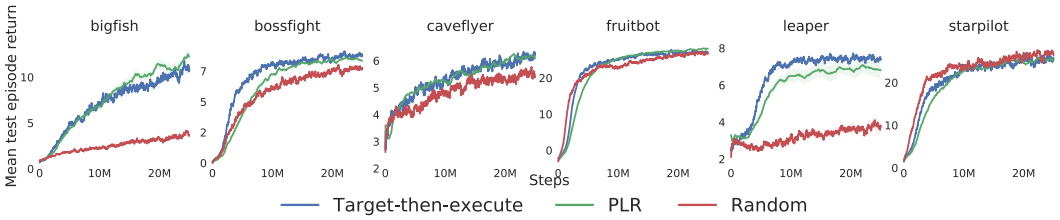

Figure 6: Target-then-execute in multi-environment, rich-observation setting (top). Averaged over 10 seeds, with 1 standard error shaded. Example environments (bottom).

In Figure 6 we see a comparison across our method with the exploiting policy in Prioritized Level Replay (PLR, Jiang et al. (2021)), selecting levels by greatest L1 value loss, and a random policy, which is the default ProcGen setting over 6 environments with significant differences, full results in the appendix. PLR performs much better than random in many environments, showing that exploitation of level control with an automatic curriculum works well. However, we see that Random performs better than PLR in most environments in early stages of training, perhaps because PLR exploits too much early on. Our method, target-then-execute, is able to match Random in early stage training, and PLR in late stage (or even surpass PLR in some cases), achieving a more robust learner across the agent's training timeline.

6 Related Work

The failure of deep RL agents to generalize to new observations or new regions of state space has been observed in several independent works (Zhang et al., 2018b;a; Kenton et al., 2019; Song et al., 2020). Our approach to this problem builds on a rich existing literature applying causal inference to RL. Though we do not explicitly model the SCM, our approach bears some similarity to optimal intervention design, a well-studied problem in both the bandit (Lee and Bareinboim, 2018; Bareinboim et al., 2015; Lattimore et al., 2016) and RL settings (Mozifian et al., 2020; Volodin et al., 2020). Also in the RL setting, Zhang et al. (2019) and Zhang et al. (2020a) both focus on learning state abstractions which remove sources of spurious correlations in the environment. Causal inference methods have also been applied extensively to the offline and imitation learning settings, where confounding is more likely to occur (Wang et al., 2020; de Haan et al., 2019). Additionally, Jaques et al. (2019) use causal inference to provide sources of intrinsic motivation in the multi-agent RL setting.

We also build on existing work on exploration in reinforcement learning (Thrun and Möller, 1992; Dayan and Sejnowski, 1996), drawing lessons from the principle of optimism under uncertainty (Auer, 2002; Chen et al., 2017) and the PAC-MDP literature (Strehl et al., 2006; Jiang et al., 2017). Recent approaches to uncertainty-guided exploration in deep RL have used exploration bonuses (O'Donoghue et al., 2018; Janz et al., 2019; Bellemare et al., 2016) and Thompson sampling approximations (Lu and Van Roy, 2017; Osband et al., 2016; Osband and Van Roy, 2017) which make use of deep ensembles (Lakshminarayanan et al., 2017). Ensembles are also used by Buckman et al. (2018) to estimate model uncertainty in order to improve sample efficiency in model-based RL and by Chua et al. (2018) to disentangle aleatoric and epistemic uncertainty. Some existing exploration strategies aim to maximize information gain, but this is done with respect to the environment dynamics model rather than the value function (Pathak et al., 2019; Shyam et al., 2019). Şimşek and Barto (2004) additionally use a notion of state novelty to construct options.

7 Conclusions

In this work, we highlight the issue of causal confusion in online RL under two scenarios: the limited data regime for the single-environment setting and the unlimited data regime for the multi-environment setting. We provide theoretical results showing the equivalence of performing causal discovery and learning an optimal value function on the entire state space, highlighting the importance of exploration for learning a robust value function. Finally, we provide a novel exploration algorithm that efficiently gathers data necessary to learn a causally correct policy that effectively generalizes in both single- and multi-environment settings.

REFERENCES

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

James Bannon, Brad Windsor, Wenbo Song, and Tao Li. Causality and batch reinforcement learning: Complementary approaches to planning in unknown domains, 2020.

Elias Bareinboim, Andrew Forney, and Judea Pearl. Bandits with unobserved confounders: A causal approach. *Advances in Neural Information Processing Systems*, 28:1342–1350, 2015.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in neural information processing systems*, pages 1471–1479, 2016.

Steven J Bradtke and Andrew G Barto. Linear least-squares algorithms for temporal difference learning. *Machine learning*, 22(1):33–57, 1996.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, volume 31, pages 8224–8234, 2018.

Richard Y. Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. UCB Exploration via Q-Ensembles. *arXiv:1706.01502*, November 2017.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019.

Özgür Şimşek and Andrew G. Barto. Using relative novelty to identify useful temporal abstractions in reinforcement learning. Association for Computing Machinery, 2004. ISBN 1581138385.

Peter Dayan and Terrence J Sejnowski. Exploration bonuses and dual control. *Machine Learning*, 25(1):5–22, 1996.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *Advances in Neural Information Processing Systems*, pages 11698–11709, 2019.

William Fedus, Dibya Ghosh, John D. Martin, Marc G. Bellemare, Yoshua Bengio, and Hugo Larochelle. On catastrophic interference in atari 2600 games, 2020.

Andrew Forney, Judea Pearl, and Elias Bareinboim. Counterfactual data-fusion for online reinforcement learners. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1156–1164, 2017.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning, 2011.

David Janz, Jiri Hron, Przemysł aw Mazur, Katja Hofmann, José Miguel Hernández-Lobato, and Sebastian Tschiatschek. Successor uncertainties: Exploration and uncertainty in temporal difference learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International Conference on Machine Learning*, pages 3040–3049. PMLR, 2019.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay, 2021.

Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E. Schapire. Contextual decision processes with low Bellman rank are PAC-learnable. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1704–1713. PMLR, 2017.

Nathan Kallus and Angela Zhou. Confounding-robust policy improvement. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.

Z. Kenton, A. Filos, O. Evans, and Y. Gal. Generalizing from a few environments in safety-critical reinforcement learning. *arXiv e-prints*, July 2019.

Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6405–6416, 2017.

Finnian Lattimore, Tor Lattimore, and Mark D Reid. Causal bandits: learning good interventions via causal inference. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1189–1197, 2016.

Sanghack Lee and Elias Bareinboim. Structural causal bandits: where to intervene? *Advances in Neural Information Processing Systems 31*, 31, 2018.

Xiuyuan Lu and Benjamin Van Roy. Ensemble sampling. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 3258–3266. Curran Associates, Inc., 2017.

Melissa Mozifian, Amy Zhang, Joelle Pineau, and David Meger. Intervention design for effective sim2real transfer, 2020.

Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In *Proceedings of the 34th International Conference on Machine Learning*, pages 2701–2710, 2017.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034, 2016.

Brendan O'Donoghue, Ian Osband, Remi Munos, and Volodymyr Mnih. The uncertainty bellman equation and exploration. In *International Conference on Machine Learning*, pages 3836–3845, 2018.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *ICML*, 2019.

Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, New York, NY, USA, 2nd edition, 2009. ISBN 052189560X, 9780521895606.

J. Peters, P. Bühlmann, and N. Meinshausen. Causal inference using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society, Series B (with discussion)*, 78(5):947–1012, 2016.

Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, pages 5779–5788, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

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

Alexander L Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L Littman. Pac model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 881–888. ACM, 2006.

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

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Sebastian B. Thrun and Knut Möller. Active exploration in dynamic environments. In J. Moody, S. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems*,

volume 4, pages 531–538. Morgan-Kaufmann, 1992.

Sergei Volodin, Nevan Wichers, and Jeremy Nixon. Resolving spurious correlations in causal models of environments via interventions. *arXiv preprint arXiv:2002.05217*, 2020.

Lingxiao Wang, Zhuoran Yang, and Zhaoran Wang. Provably efficient causal reinforcement learning with confounded observational data. *arXiv preprint arXiv:2006.12311*, 2020.

Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *CoRR*, abs/1806.07937, 2018a.

Amy Zhang, Zachary C. Lipton, Luis Pineda, Kamyar Azizzadenesheli, Anima Anandkumar, Laurent Itti, Joelle Pineau, and Tommaso Furlanello. Learning causal state representations of partially observable environments. *CoRR*, abs/1906.10437, 2019.

Amy Zhang, Clare Lyle, Shagun Sodhani, Angelos Filos, Marta Kwiatkowska, Joelle Pineau, Yarin Gal, and Doina Precup. Invariant causal prediction for block mdps. In *Proceedings of Machine Learning and Systems 2020*, pages 7623–7633. 2020a.

Chiyuan Zhang, Oriol Vinyals, Rémi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *CoRR*, abs/1804.06893, 2018b.

Junzhe Zhang, Daniel Kumor, and Elias Bareinboim. Causal imitation learning with unobserved confounders. In *Advances in Neural Information Processing Systems*, 2020b.

A PROOFS

Observation 1. *Let $\mathcal{M}$ be an MDP with state space $\mathcal{X} \subset \mathbb{R}^d$ and π a policy inducing stationary distribution η^π over $\mathcal{X}$. Let $\mathbf{w}_t$ be such that $V^\pi(\mathbf{x}) = \mathbf{w}_T^\top \mathbf{x}$ for all $\mathbf{x} \in \mathcal{X}$. Then if $\exists \mathbf{v} \in \mathbb{R}^d$ $\mathbb{E}_{\mathbf{x}\sim\eta^\pi(\mathcal{X})}[(\mathbf{v}^\top \mathbf{x})^2] = 0$, we have for any real number $M \in \mathbb{R}$, there exists vector $\mathbf{w} \in \mathbb{R}^d$ such that*

$$\mathbb{E}_{\mathbf{x}\sim\eta^\pi(\mathcal{X})}[|V^\pi(\mathbf{x}) - \mathbf{w}^\top \mathbf{x}|] = 0 \text{ and } \max_{\mathbf{x}\in\mathcal{X}}[|V^\pi(\mathbf{x})] - \mathbf{w}^\top \mathbf{x}|] > M \ . \tag{6}$$

Proof. The condition of this observation requires that the dimension of the support of η^π be strictly less than d. We then note that for any $\mathbf{w}$ attaining zero linear approximation error on the support of η^π, we can construct a new vector also attaining zero error of the form $\mathbf{w}_\alpha = \mathbf{w} + \alpha \mathbf{v}$ for any $\alpha \in \mathbb{R}$. Let $\beta = \mathbf{w}_T^\top \mathbf{v}$. Take any $x \in \mathcal{X} : \mathbf{x}^\top v = \gamma \neq 0$. Then given $M \in \mathbb{R}$, it suffices to set $\alpha = \frac{M}{\gamma} \pm \beta$ to obtain in $\mathbf{w}_\alpha$ a linear predictor with zero error on η^π and error at least M on some state in $\mathcal{X}$. □

Theorem 1. *Let $\mathbf{E}_{train} = \{\mathcal{M}\}$ for $\mathcal{M} = (\mathcal{S}, A, R, P, \gamma, p_0)$, and $\mathbf{E}_{test} = \{(\mathcal{S}, A, R, P, \gamma, p_S) | S \subseteq \mathcal{S}\}$. Let $\Pi_S = \{\pi_{\{s\}} | s \in \mathcal{S}\}$. Then if $\mathcal{M}$ is fully connected*

$$\min_Q \max_{\mathbf{E}_{test}} \mathbb{E}[\max_a |Q^*(s_0, a) - Q(s_0, a)|] = \min_Q \max_{\pi_\mathbf{S} \in \Pi_S} \mathbb{E}_{s_0\sim\pi_\mathbf{S}}[\max_a |Q^*(s_0, a) - Q(s_0, a)|].$$

Proof. The proof follows straightforwardly from the observation that for any initial state distribution p_0, it is possible to construct an intervention which induces this state distribution at its termination. When $p_0 = \delta_s$, this is trivial: we simply pick select the targeting intervention π_s, which deterministically returns the state s. The $p_0 = \sum \alpha_s \delta_s$, then we construct a policy over options μ of the form $\mu(\pi_s|s_0) = \alpha_s$ for s_0 in the training MDP's initial state distribution. There is therefore a one-to-one correspondence between initial state distributions and distributions over states output by the intervention policies, and so the set over which the maximization problem occurs is the same in both the left hand side and the right hand side of the above equation, which means the optimization problems have the same set of solutions. □

Theorem 2. *Let $\mathbf{X}_t$, $\mathbf{X}'_t \in \mathbb{R}^{t\times d}$ with k^{th} rows $\mathbf{x}_k$, $\mathbf{x}'_k$ respectively, and $\mathbf{r}_t \in \mathbb{R}^t$ denote a collection of sampled transitions from an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, R, P, \gamma)$. Assume each $\mathbf{x}_k$ is sampled from some distribution $\eta(\mathcal{X})$ and $\mathbf{x}'_k$ is obtained by sampling an action from the policy $\pi(\mathbf{x})$ and observing the induced transition in the MDP. Assume the realizable setting: i.e. there exists some $\mathbf{w}_T \in \mathbb{R}^d$ such that $V^\pi(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w}_T \rangle$. Then if the sampling distribution η is such that $\mathbb{E}_{\mathbf{x}\sim\eta}[(\mathbf{x}^\top v)^2] > 0$ for all unit vectors $\mathbf{v}$, we will have*

$$\mathbf{w}_t = [\mathbf{X}_t(\mathbf{X}_t - \gamma \mathbf{X}'_t)^\top]^{-1}[\mathbf{X}_t \mathbf{r}_t] \overset{t\to\infty}{\longrightarrow} \mathbf{w}_T \tag{3}$$

Theorem 2 highlights that in the function approximation regime it is not necessary to visit all states in the MDP in order to find a robust value function. Rather, it is only necessary to follow a policy which visits a sufficiently informative set of states so as to enable interpolation at evaluation time, at least in the realizable setting.

The proof of this result is simplified by the following Lemma from Bradtke and Barto (1996) showing convergence of the LS TD algorithm. While the LS TD algorithm approximates the state-value function, it is straightforward to generalize to the state-action values following the approach of Lagoudakis and Parr (2003) by considering instead of the state encoding $\mathbf{x}$, a state-action encoding $\phi(\mathbf{x}, a)$. For simplicity of exposition, we focus on the value approximation case. We will use the notation $\bar{r}$ to denote the vector of expected rewards, where $\bar{r}_\mathbf{x}$ denotes the expected reward at state $\mathbf{x}$. We let P be the matrix such that $P\mathbf{X} = \mathbb{E}[\mathbf{X}_{t+1}|\mathbf{X}_t = \mathbf{X}]$, where $\mathbf{X}$ is the matrix of states in $\mathcal{X}$.

Given a sample of t transitions ($\mathbf{X}_t$, $\mathbf{r}_t$, $\mathbf{X}'_t$) drawn by following a policy π in an MDP $\mathcal{M}$ (i.e. by following transitions in a Markov chain), the LS TD algorithm outputs the following linear function approximator

$$w_t = [\frac{1}{t}\sum_{k=1}^{t} \mathbf{x}_t(\mathbf{x}_t - \gamma \mathbf{x}'_t)^\top]^{-1}[\frac{1}{t}\sum_{k=1}^{t} \mathbf{x}_t r_t] \ . \tag{7}$$

Lemma 1 ((Bradtke and Barto, 1996))**.** *For any Markov chain, when (1) θ_t is found using the algorithm LS TD; (2) each state $\mathbf{x} \in \mathcal{X}$ is visited infinitely often; (3) each state $\mathbf{x} \in \mathcal{X}$ is visited in*

the long run with probability 1 in proportion to π_x; and (4) [$\mathbf{X}^\top \Pi(I-\gamma P)\mathbf{X}$] is invertible, and Π is the diagonal matrix diag(π), then:

$$w_{LSTD} = [\mathbf{X}^\top \Pi(I-\gamma P)\mathbf{X}]^{-1}[\mathbf{X}^\top \Pi \bar{r}] \tag{8}$$

Proof. Because we assume the realizable setting, i.e. that there exists a regressor $\mathbf{w}_T$ which attains zero value approximation error, and that the dimension of the state space visited during training is equal to d, we have the existence and uniqueness of an optimal value function approximator. Further, under any stationary state visitation distribution π with full support on the state space, with $\Pi = \text{diag}(\pi)$, we have

$$[\mathbf{X}^\top \Pi(I-\gamma P)\mathbf{X}]^{-1}[\mathbf{X}^\top \Pi \bar{r}] = [\mathbf{X}^\top (I-\gamma P)\mathbf{X}]^{-1}[\mathbf{X}^\top \bar{r}]\,.$$

The principal difference between the conditions of our result and the conditions of Lemma 1 is that we do not assume that the policy used by the agent to reach the states $\mathbf{X}$ is the same as the policy used by the agent in generating the transition $\mathbf{x}, r, \mathbf{x}'$. Instead, our assumption on η requires that it visit each dimension of the state space with some nonzero probability.

In particular, we have

$$\left[\mathbf{X}^\top \Pi_\eta (I-\gamma P^\pi)\mathbf{X}\right]^{-1}[\mathbf{X}^\top \Pi_\eta \bar{r}] = \mathbf{w}_T\ .$$

We can then apply the mechanics of the proof of Lemma 1 as follows.

$$\begin{aligned}
\lim_{t\to\infty} w_t &= \lim_{t\to\infty}\left[\frac{1}{t}\sum \mathbf{x}_k(\mathbf{x}_k - \gamma \mathbf{x}'_k)^\top\right]^{-1}\left[\frac{1}{t}\sum \mathbf{x}_k r)k\right] \\
&= \left[\lim_{t\to\infty}\frac{1}{t}\sum \mathbf{x}_k(\mathbf{x}_k - \gamma \mathbf{x}'_k)^\top\right]^{-1}\left[\lim_{t\to\infty}\frac{1}{t}\sum \mathbf{x}_k r)k\right] \\
&= \left[\sum \eta_{\mathbf{x}} \sum P(\mathbf{x},\mathbf{x}')\mathbf{x}(\mathbf{x}-\gamma\mathbf{x}')\right]^{-1} \\
&\left[\sum \eta_x \mathbf{x} \quad \sum_{\mathbf{x}'} P(\mathbf{x},\mathbf{x}')r(\mathbf{x},\mathbf{x}')\right] \\
&= \left[\sum \eta_{\mathbf{x}} \sum P(\mathbf{x},\mathbf{x}')\mathbf{x}(\mathbf{x}-\gamma\mathbf{x}')\right]^{-1}\left[\sum \eta_x \mathbf{x} \sum_{\mathbf{x}'} \bar{r}(\mathbf{x})\right] \\
&= \left[\mathbf{X}^\top \Pi_\eta (I-\gamma P^\pi)\mathbf{X}\right]^{-1}[\mathbf{X}^\top \Pi_\eta \bar{r}] = \mathbf{w}_T
\end{aligned}$$

□

Observation 2. *Let $s \in \mathcal{S}$, and let $\tau = (s_i, a_i, r_i)_{i=1}^N$ be a trajectory collected from the target-then-execute policy $\pi^* \circ \pi_s$. Let t denote the first timestep such that $s_t = s$. Then*

$$\mathbb{E}_{\tau\sim\pi^*\circ\pi_s}[\sum_{k=t}^{T}\gamma^{k-t}R_k] = \mathbb{E}[Q^*(s_0,a^*)|p(s_0)=\delta_s]. \tag{9}$$

Proof. The result follows immediately from the assumption of full observability. In particular, the expected return from a state s is independent of the trajectory taken to reach s. Thus, letting T be the random variable denoting the termination time of the episode, we obtain

$$\mathbb{E}[\sum_{k=0}^{T}\gamma^k R_k | s_0 = s] = \mathbb{E}[\sum_{k=t}^{T}\gamma^k R_k | s_t = s]\ . \tag{10}$$

□

Theorem 3. *Let s^*, a^* be a state action pair selected by target-then-execute at episode h. Let P_h denote the posterior over Q-functions at episode h obtained by a Bayesian model with Gaussian prior and likelihood. Then*

$$(s^*, a^*) \in \textit{arg max}_{s,a} IG(P_h, s, a)\ . \tag{5}$$

A note on the modelling assumptions. The above result assumes that the agent is only interested in the mean value of the return from the state-action pair, and is not also modelling its variance. If the

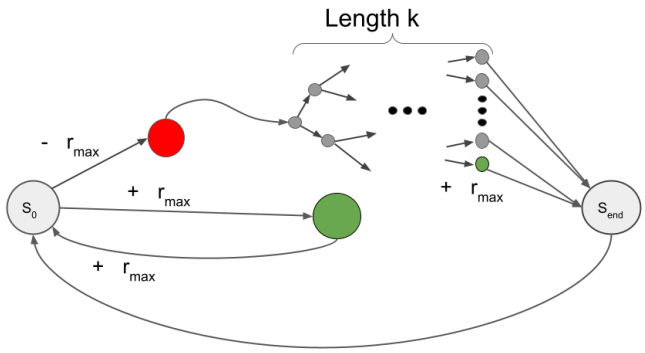

Figure 7: Hard MDP in Proof of Observation 2. Green states yield positive reward, red states yield negative reward, and grey states give zero reward. A single arrow coming out of a state indicates both a_1 and a_2 deterministically take the agent to the next state.

agent is modelling both mean and variance (for example, in distributional reinforcement learning), then the above result needs to be modified to disentangle the agent's epistemic and aleatoric uncertainties. For simplicity, we do not consider this case but note that it is straightforward to modify the proof below to apply to this setting.

Proof. To simplify notation, we let the random variable R be the return from the trajectory τ obtained as described in Secion 3.2. To show the result, we need to show that $H[Q(s,a)] = H[Q|\mathcal{D}] - \mathbb{E}_{\hat{R}\sim P_Q}[H[Q|\mathcal{D}, R]$. Doing so is a straightforward manipulation of known information quantities, leveraging the shared likelihood $P(R(s,a)|Q(s,a) = q) = \mathcal{N}(q, \sigma^2)\forall s, a$ between states and the Gaussian prior $\mathcal{N}(\mu_0, \sigma_0)$. We let the posterior distribution over Q-values at timestep h be denoted by $Q(s,a) \sim \mathcal{N}(\mu_h(s,a), \sigma_h^2(s,a))$.

$$\begin{aligned}
\text{arg max}_{s,a} IG(P_h, s, a) &= H[Q|\mathcal{D}] - \mathbb{E}[H[Q|\mathcal{D}, R] \\
&= \log c\sigma_h^2(s,a) - \log c\sigma_{h,R}^2(s,a) \\
&= \text{arg max}_{s,a} \log \sigma_h^2(s,a) - \log(\sigma_h(s,a) + \sigma_r)
\end{aligned}$$

Because $\log(x) - \log(x+a)$ is monotone with respect to x, we get

$$\begin{aligned}
&= \text{arg max}_{s,a} \log \sigma_h^2(s,a) \\
&= \text{arg max}_{s,a} H_P[Q_h(s,a)]
\end{aligned}$$

□

Observation 3. *For any $r_{\max} \in \mathbb{R}$, there exists an MDP M with rewards bounded in $[-r_{\max}, r_{\max}]$ and an initial distribution p_{train}, p_{test} such that with probability $p > 0$, for all timesteps t, the policy followed by Delayed Q-learning (Strehl et al., 2006) run from initial state distribution p_{train} attains (simple) regret $r_{\max}/2$ under the initial state distribution p_{test}.*

Proof. The family of MDPs we construct to demonstrate this result is visualized in Figure 7. For each $k \in \mathbb{N}$, we construct an MDP $\mathcal{M}_k$ consisting of $2^k + 3$ states and two actions. At the initial state S_0, the agent can choose between the MDP's two actions: a_1 and a_2. Action a_1 deterministically generates reward r_{max}, and transitions to a state which also deterministically generates reward r_{max} and then transitions back to state S_0, providing reward r_{max} again. This is the Good branch of the MDP, as cycling between S_0 and the green state provides the maximal possible value. Taking action a_2 from S_0 deterministically takes the agent to a state with reward $-r_{\text{max}}$. This state deterministically transitions to the root of the Maze component, a branching binary tree of height k in which a_1 takes the agent to a node's left child, and a_2 to the node's right child, and whose leaves all deterministically transition to S_{end} independent of the action taken. In one of the 2^{k-1} leaves of the binary tree s_l, the agent receives a reward $r_{\max}$. The only way to reach this state, however, requires correctly following a sequence of k actions. Finally, the MDP deterministically transitions back to S_{end}, which returns it to S_0. We set the initial state distributions $p_{\text{train}} = \delta_{S_0}$ and $p_{\text{test}} = \delta_{s_p}$, where s_p is the parent of the rewarding leaf s_l.

The optimal policy starting from the state S_0 is to take action a_1 and receive the reward of r_{max}. However, if the agent is placed at the root node of the Maze component of the MDP, then the optimal policy requires navigating the binary tree. The probability of reaching the rewarding state after a single random traversal of the Maze is $2^{-(k-1)}$.

Delayed Q-learning follows a greedy exploration policy with respect to an optimistically initialized value function. The algorithm updates state-action pairs after they have been visited sufficiently often so as to provide tight confidence intervals on the estimated value. Being a PAC-MDP algorithm, Delayed Q-learning takes as input a confidence parameter δ and an accuracy parameter ϵ. We then let $\epsilon_1 = \epsilon(1-\gamma)/9$, $\kappa = \frac{1}{(1-\gamma)\epsilon_1}$ and $m := \frac{\ln(3SA(1-SA\kappa)/\delta}{2\epsilon_1^2(1-\gamma)^2}$.

In particular, for fixed ϵ, γ, A, and δ, m grows as $O(\ln(S))$. We additionally observe that in any given traversal of the Maze before any Q-value is updated, the agent has probability $\frac{1}{2^{k-1}} \approx \frac{1}{S}$ of visiting any given leaf.

We first observe that the maze is traversed at most m times, as after m traversals the agent will update the state-action value $Q(S_0, a_2)$, after which it will forever choose action a_1. We now consider the simple regret of the policy followed by Delayed Q-learning at timestep t when initialized to the parent in the binary tree of s_l, denoted s_p (i.e. the *robust* performance). We suppose without loss of generality that s_l is a left child of s_p and so obtain the difference $Q^*(s_p, a_1) - Q^*(s_p, a_2) = r_{\max}$. Thus if the policy is uniform at s_p, it will attain simple regret $\frac{r_{\max}}{2}$. The probability of this occurring is equal to the probability that the agent updated the value $Q(s_p, a_2)$ in its mth traversal of the maze. An update to $Q(s_p, a_2)$ occurs only if (s_p, a_2) has been visited m times, previously. Assuming a random tiebreak rule, this event occurs independently on each trajectory with probability $\frac{1}{2^{k-1}} \approx \frac{1}{S}$ and so $P(\pi_t(s_p) = a_1) \approx \frac{1}{|S|}^m < \frac{1}{|S|}$.

Let T denote the timestep when $Q(S_0, a_2)$ is updated. For $t < T + k$, we have that π_t acts randomly on s_p and so attains simple regret at least $\frac{r}{2}$ with probability 1 . For $t \geq T + k$, we have that π_t acts randomly with probability at least $1 - \frac{1}{2^{k-1}}$. Thus, for any fixed probability $0 < p < 1$, setting $k \geq 1 - \log(1-p)$ yields

$$P(\mathbb{E}_{\mathcal{M}_k, p_0 = \delta_{s_p}}[R(\pi^*) - R(\pi_t)] > \frac{r_{\text{max}}}{2}) \geq p \ . \tag{11}$$

□

B INTERVENTIONS

The definition we use to define agent-interventions in an environment is distinct from the notion of causal graph manipulation used by Pearl (2009), and so we provide some discussion of and motivation for this distinction. Do-interventions require direct manipulations to variables in the causal graph, correspnoding to a model of the world in which the agent can fix variables to specific values without changing anything else in the data-generating process. While this is practical in some scenarios such as treatment decisions in medical trials, it is problematic when we translate this notion to the RL setting. Indeed, a do-intervention on the SCM described by Figure 1 would entail setting the value of a state variable at a specific timestep to take a specific value while leaving both the agent's policy and the remaining environment dynamics unchanged. While such an intervention may be desirable (for example, if a developer intervenes on the game state of a chess-playing agent to see how it will respond to a piece being taken off the board), it is by definition not something that can be performed by the agent.

Instead, our notion of agent-interventions considers only how changes to the agent's policy affect the data-generating process of the MDP – i.e. we do not assume that there is some omnipotent entity that can arbitrarily modify variables in the world, but rather that changes to these variables must occur through the actions of the agent. This is equivalent to performing a do-intervention on a subset of the action variables $A_{t:t+k}$ in the MDP's graphical model in the traditional language of interventions on a causal graph. However, we cannot directly translate this change in the agent's policy to a do-intervention of the form do$(X_i = x_i)$, where X_i is some dimension of the state. In particular, this means that while the agent can act in such a way that $X_i^t = x_i^t$ for some timestep t, the sequence of actions it takes to achieve this outcome may change the distribution over states such that the distribution over the remaining components of the state $(X_j^t)_{j \neq i}$ differs from what would be

obtained by the agent following its default policy. Indeed, there may be many interventional policies the agent can follow which lead to the outcome $X_i^t = x_i$ for some t, and each of these interventional policies may induce a different distribution over the state $(X_j^t)_{j \neq i}$.

In ergodic environments, it is always possible to construct an option with policy $\pi_{\text{do}(X_i=x)}$ which induces the distribution $p_0(X|\text{do}(X_i = x))$ in the state space on termination. This can be achieved by an appropriate weighting of the goal-reaching policies $\pi_{\mathbf{x}}$ for each $\mathbf{x}$ in the support of $p_0(X|\text{do}(X_i = x))$. In non-ergodic environments, however, this may be impossible as some states may be unreachable by the agent. As an illustrative example, consider the problem of a robot exploring a room with a timer on the wall. Because it takes time for the robot to reach a given state, it is impossible to follow a policy which replicates the distribution induced by the intervention $\text{do}(X, Y = x, y)$ which sets the agent to a particular (x, y)-position. This problem also arises in any attempt to replicate the distribution $\text{do}(X_i^t = x)$ for a specific timestep t: while in an ergodic MDP for sufficiently large t (i.e. greater than the mixing time of the induced Markov chain) there may exist an intervention policy which induces the same distribution over states as the evaluation policy, this will not in general hold for small t or for non-ergodic environments.

Given the impossibility of constructing such policies, along with the fact that the set of interventional distributions which maximize the value approximation error of a given estimated value function contains at least one Dirac delta distribution, we conclude that the class of agent-interventions we consider is both a realistic model of agent interaction with the world, and sufficient to identify sources of causal confusion.

C Experiment Details

C.1 MountainCar

We follow the experimental procedure described by de Haan et al. (2019) with respect to the training data and the policy model, basing our implementation on the open-source code provided by the authors. Because the expert policy is deterministic and so the imitation learning dataset does not cover the joint state-action space, we pretrain the exploration policy online in the environment for 100 episodes before deploying it for evaluation. Because the ensemble is given by a set of policies rather than value functions, we apply an unnormalized version of the disagreement bonus given in Algorithm 1 to the probability distributions output by the ensemble heads. We additionally modify the environment for evaluation so that we can deterministically initialize the environment to a given initial state. We evaluate our method, target-then-execute, and the approach of de Haan et al. (2019) on a range of these initial states, running the policy induced by each possible feature mask for 10 random seeds from each state and performing the posterior weight updating based on the average return of this policy. We determine whether the agent has correctly inferred the causal structure based on whether the posterior weight assigned to the spurious previous action variable is equal to the weight assigned to the position and velocity.

C.2 MiniGrid

To simulate the motivating autonomous vehicle example, we construct a MiniGrid environment in which the agent must pass through a bottleneck square in order to reach its goal state. The bottleneck square (i.e. the rail crossing) can be passed through normally when the marker on this square is green, but terminates the episode and yields a reward of -1 if the square is red. The agent must also avoid colliding with other cars, modelled as blue circles. In the training environment, the rail crossing is green at initialization, and at any given timestep has probability of switching color from green to red or vice versa is equal to 0.1. In most instances when the color switches to red, the agent is behind one of the obstacles, mimicking the setting where an autonomous vehicle never or almost-never sees a red rail crossing during training. Additionally, the agent in this setting is initialized to be behind the obstacles. In the validation environment, we change the initialization distribution so that now the agent is at the front of the line of cars. Otherwise, the environment parameters are identical to the environment used for offline data collection. This setup means that some fraction of the time, the agent will find itself at the red light purely by chance, though this fraction will be quite small. We observe qualitatively that the uncertainty value obtained by the agent at this state is large relative to when the agent is in front of the green light. We pretrain the exploration policy to maximize the

expected sum of discounted uncertainty bonuses on the offline data used to train the ensemble, and allow the exploration policy to continue updating during the validation phase. The expert data used to train the ensemble and exploration policy is generated by a pretrained model which following an ϵ-greedy policy for $\epsilon = 0.2$.

C.3 GRIDWORLD

We use a 15×15 gridworld in which the agent is initialized in position $(1, 1)$. We place goal states in locations $(1, h-1)$ and $(w, 1)$. The agent receives reward $+1$ upon reaching a goal state, terminating the episode, and zero reward otherwise. The episode terminates after 128 steps if the goal has not been reached. To evaluate the Q-learning agents, we run ϵ-greedy exploration with $\epsilon = 0.1$. The 'random' initialization is sampled from a normal distribution with mean zero and variance $\frac{1}{|S|}$ for each state-action pair, while the optimistic initialization is fixed at 1. We use $\gamma = 0.99$, and train each agent for 200 episodes.

C.4 PROCGEN

The target-then-execute method is implemented as an additional ϵ-greedy addition to PLR (Jiang et al., 2021). We took the original best hyperparameters from PLR and tuned just three hyperparameters for our target-then-execute method, shown in Table 2. Staleness refers to the type of prioritization used in the original PLR method. The default prioritization method is "staleness-aware," or how long ago a specific level was sampled. The sampling distribution is defined as a mixture of two distributions based on the level scores P_S and how long ago each level was last sampled P_C:

$$P_{\text{replay}} = (1 - \rho) \cdot P_S + \rho \cdot P_C.$$

Parameter name	Value
Exploration parameter ϵ	0.05
Temperature parameter β	0.08
Staleness coefficient ρ	0.05

Table 2: A complete overview of used hyper-parameters.

Results over all 16 environments are shown in Figure 8.

We also provide results of our sweeps in Figure 9, Figure 10, and Figure 11. We see that performance across most environments is quite robust to changes to these hyperparameters, with the exception of `bigfish` and `heist`.

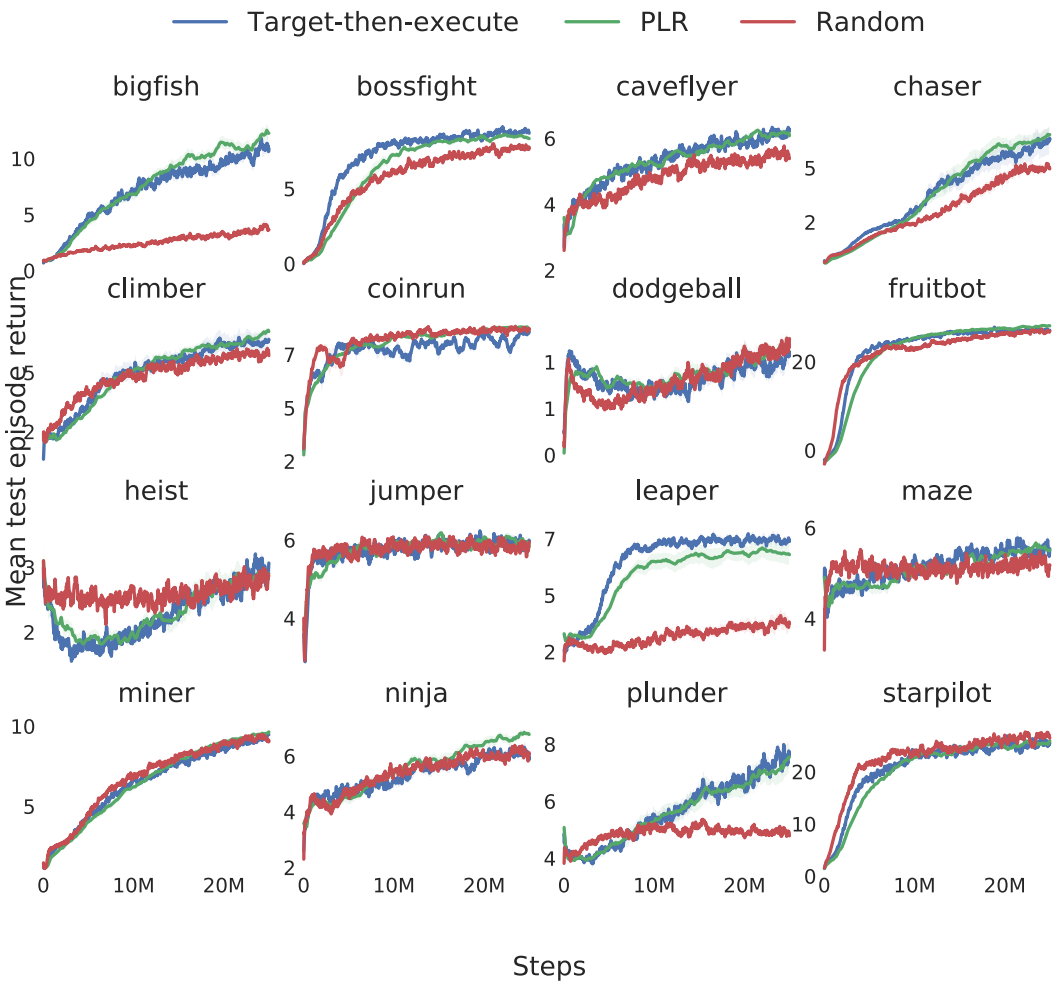

Figure 8: Method in multi-environment, rich-observation setting. Blue is target-then-execute. Averaged over 10 seeds, with 1 standard error shaded.

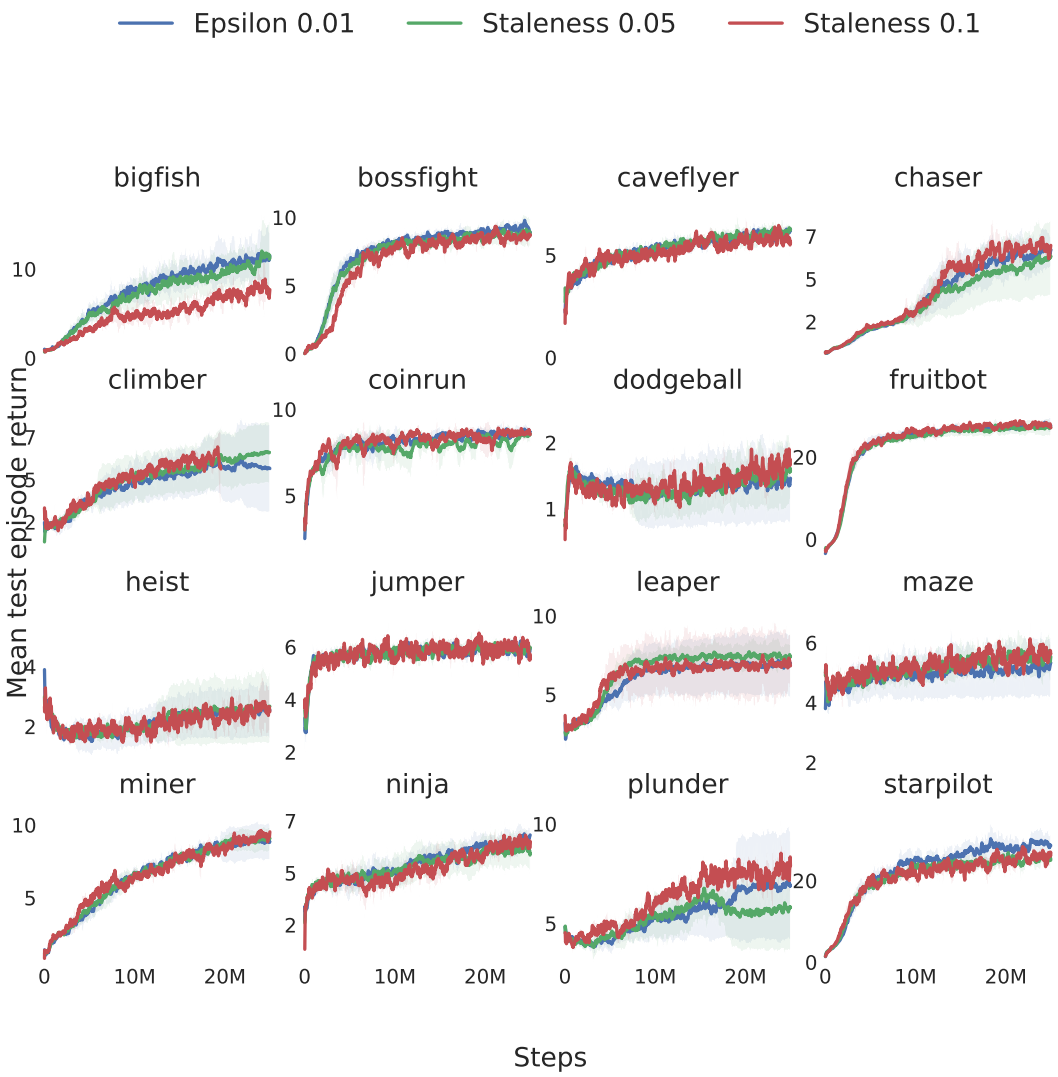

Figure 9: Sweep on exploration parameter ϵ for ProcGen.

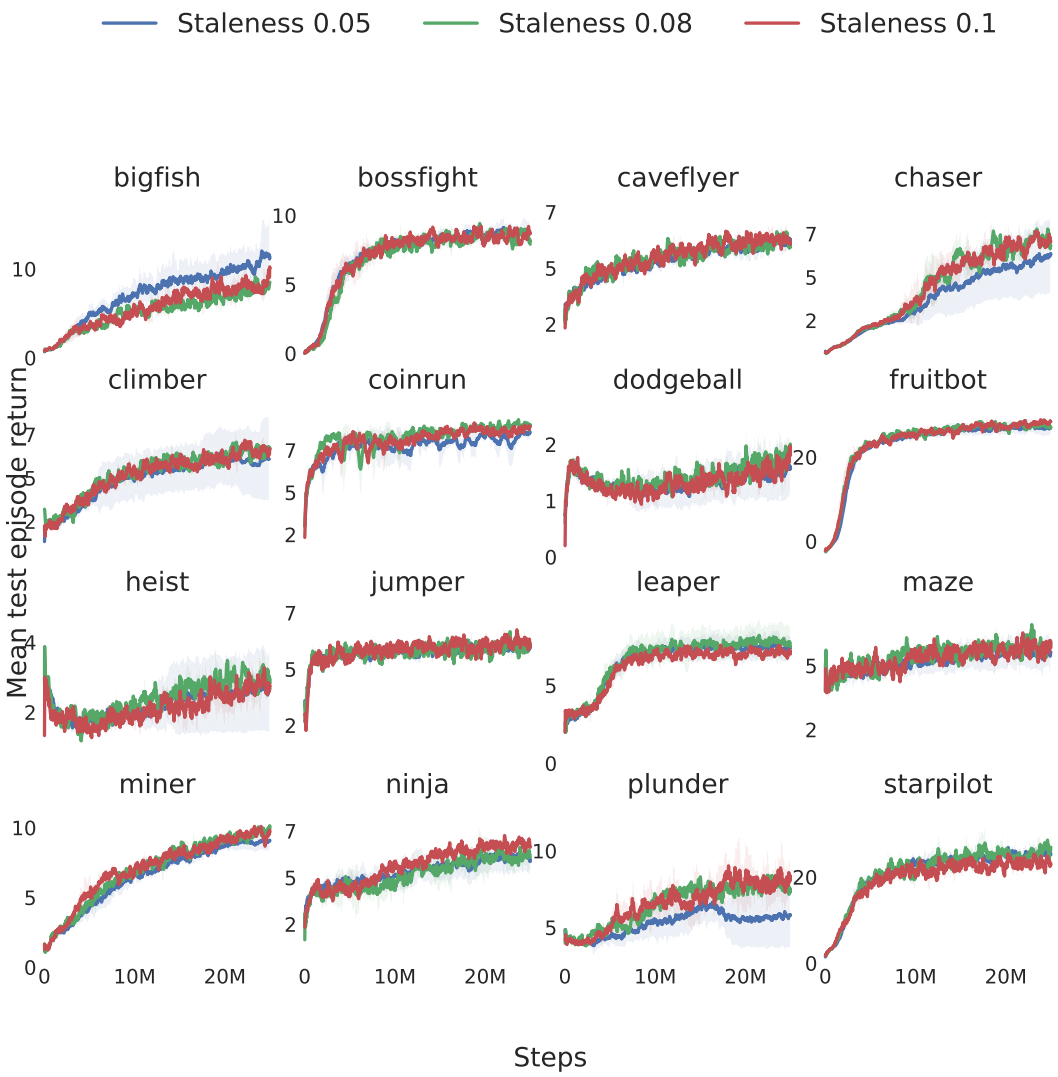

Figure 10: Sweep on staleness coefficient ρ for ProcGen.

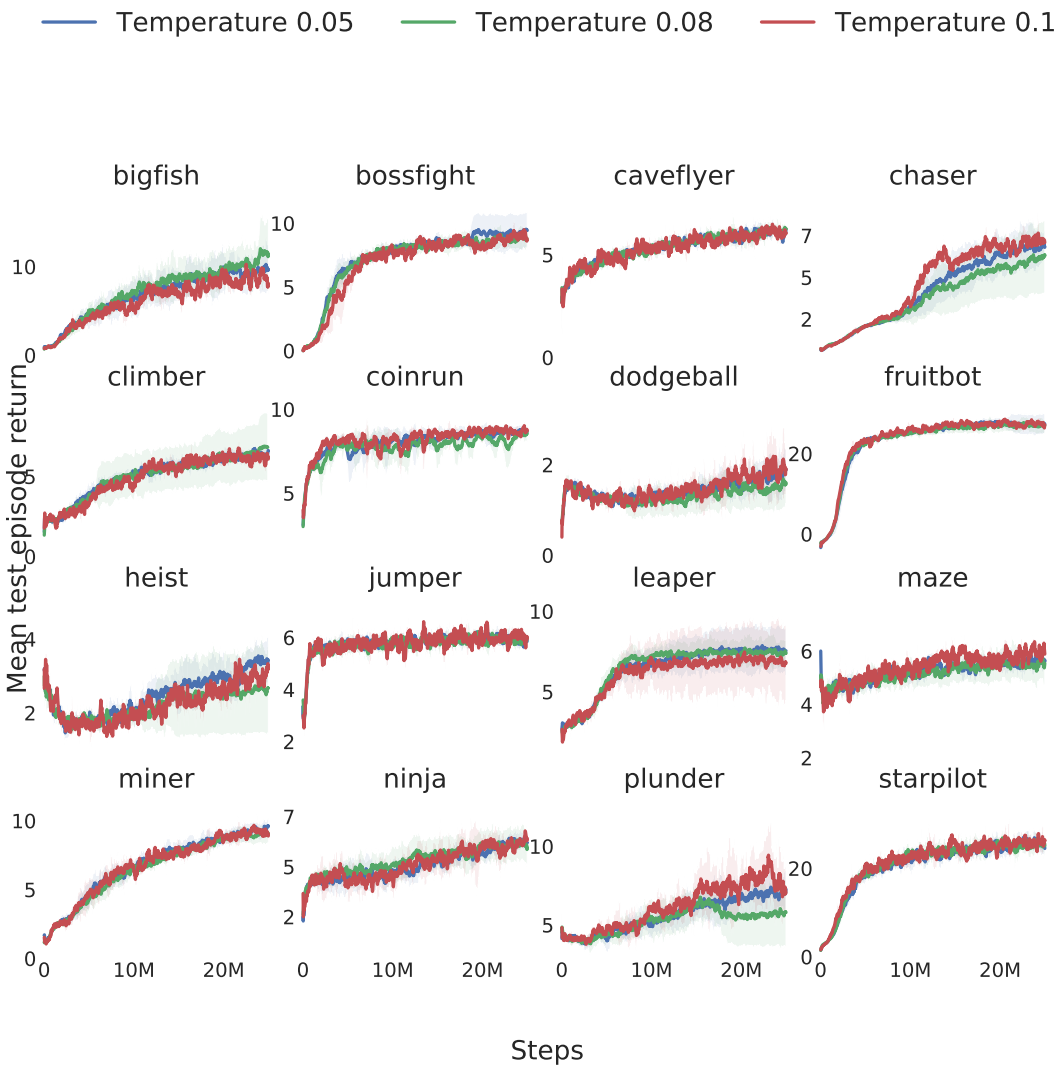

Figure 11: Sweep on temperature parameter β for ProcGen.

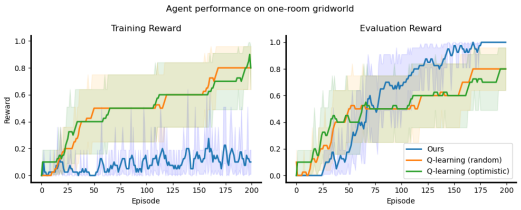

Figure 12: Return from train and test phases in a 14×14 Gridworld. Stochasticity in returns comes from randomness in the initialized Q-functions. Results are averaged over 10 ensemble initializations.

D ADDITIONAL GRIDWORLD EVALUATIONS

In addition to the results in the main body, we show that target-then-execute can be used in conjunction with online RL to encourage state coverage during training in hard-exploration environments, even in the tabular setting where spurious correlations cannot exist. In this experiment, shown in Figure 12 ,we consider a large gridworld with a single goal state. The agent is initialized in the top left corner of the grid, and the goal state is in the bottom right corner. We evaluate ensemble PSRL, epsilon-greedy, and target-then-execute, and find that target-then-execute consistently outperforms both on average, producing agents that consistently find the goal state in a limited-data setting. Our approach exhibits less dependence on the value function initialization than its competitors, and attains higher performance in the medium-data regime.