MITIGATING GOAL MISGENERALIZATION VIA MINIMAX REGRET

Anonymous authors
Paper under double-blind review

ABSTRACT

Robustness research in reinforcement learning often focuses on ensuring that the policy consistently exhibits capable, goal-driven behavior. However, not every capable behavior is the intended behavior. *Goal misgeneralization* can occur when the policy generalizes capably with respect to a 'proxy goal' whose optimal behavior correlates with the intended goal on the training distribution, but not out of distribution. Though the intended goal would be ambiguous if they were perfectly correlated in training, we show progress can be made if the goals are only *nearly ambiguous*, with the training distribution containing a small proportion of *disambiguating levels*. We observe that the training signal from disambiguating levels could be amplified by regret-based prioritization. We formally show that approximately optimal policies on maximal-regret levels avoid the harmful effects of goal misgeneralization, which may exist without this prioritization. Empirically, we find that current regret-based Unsupervised Environment Design (UED) methods can mitigate the effects of goal misgeneralization, though do not always entirely eliminate it. Our theoretical and empirical results show that as UED methods improve they could further mitigate goal misgeneralization in practice.

1 INTRODUCTION

As reinforcement learning (RL) is increasingly applied in complex, open-ended, real-world environments, it becomes infeasible for training to comprehensively cover all situations that an agent will face in deployment. A particular challenge arises when insufficiently diverse training creates a 'proxy goal' that, compared to the true goal, induces similar behavior during training but radically different behavior out of distribution. In the face of proxy goals, standard RL methods sometimes find a policy that internalizes the wrong goal, completely ignoring the true goal in favor of capably pursuing the proxy goal out of distribution (Langosco et al., 2022; Shah et al., 2022). This phenomenon, known as goal misgeneralization, is a pressing problem in assuring the safety of RL agents (Ngo et al., 2023).

If the true goal and the proxy goal are perfectly correlated in the training distribution, then goal generalization comes down to the algorithm's inductive biases. However, we find that we can make progress on goal misgeneralization regardless of the algorithm's biases, by working in a relaxed setting where the training distribution is only *nearly ambiguous* providing a weak training signal in favor of optimizing the true goal. In particular, we model a complex environment as comprising a series of *levels* (Cobbe et al., 2020; Kirk et al., 2023) which can be designed during training (Dennis et al., 2021). We assume that while most training levels leave the true goal ambiguous, a small proportion of levels are available to disambiguate the goals by incentivizing different behavior. In this setting, we show that the standard RL training method of domain randomization (DR; Tobin et al., 2017), which optimizes for expected return on the training level distribution, may still induce goal misgeneralization if the proportion of disambiguating levels in training is sufficiently small.

However, we observe that even when following the proxy goal is approximately optimal in terms of maximum expected return on the training distribution, it is clearly sub-optimal in terms of minimax expected regret (MMER; Savage, 1951), since it leaves the true goal unfulfilled in disambiguating levels. We hypothesize that MMER-based training methods such as unsupervised environment design (UED; Dennis et al., 2021) should naturally mitigate this kind of goal misgeneralization, anticipating a possible distribution shift and amplifying the weak training signal from disambiguating levels. In this paper, we contribute the following theoretical and empirical results in support of this hypothesis:

- In Section 4, we introduce a theoretical model of the consequences of goal misgeneralization in RL based on the existence of a finite resource that can be allocated to either the true goal or the proxy goal. We prove that when the distribution of training levels is dominated by ambiguating levels, (A) approximate DR becomes susceptible to goal misgeneralization, but (B) approximately optimizing MMER is robust to goal misgeneralization.
- In Section 5, we complement our theoretical findings with experiments in custom JAX-based grid-world environments with procedurally-generated levels exhibiting proxy goals. We show that two UED methods—namely (PLR$^{\perp}$; Jiang et al., 2022) and (ACCEL; Parker-Holder et al., 2023)—are significantly more robust to goal misgeneralization than DR.

These results support regret-based UED methods as a promising approach for safe goal generalization. We note that this approach requires the ability to design training levels, but this requirement is often satisfied in practice, having a simulator in sim-to-real transfer (Tobin et al., 2017; Peng et al., 2018; Kumar et al., 2021; Makoviychuk et al., 2021; Muratore et al., 2022; Ma et al., 2024), having a generative environment model (Bruce et al., 2024), or having a world model (Ha & Schmidhuber, 2018; Hafner et al., 2019; Schrittwieser et al., 2020; Hafner et al., 2023; Valevski et al., 2024).

2 Preliminaries

2.1 Underspecified Markov decision processes

A reward-free *underspecified Markov decision process* (UMPD) is a tuple $M = \langle A, \Theta, S, \mathcal{I}, \mathcal{T} \rangle$ where A is the agent's action space, Θ is the space of the free parameters of the environment, S is a state space, $\mathcal{I} : \Theta \to \Delta(S)$ is an initial state distribution, and $\mathcal{T} : \Theta \times S \times A \to \Delta(S)$ is a conditional transition distribution. An agent's behavior in an UMDP is represented by a *policy*, a conditional action distribution $\pi : \Theta \times S \to \Delta(A)$. We denote by Π the set of all policies. Given a level $\theta \in \Theta$ we have a reward-free MDP $M_\theta = \langle A, S, \mathcal{I}(\theta), \mathcal{T}(\theta, -, -) \rangle$ and an agent's level-specific policy $\pi(\theta, -)$. We omit θ when not relevant or clear from the context. Together with the initial state distribution and transition distribution a policy π induces a distribution over *trajectories* $\tau = (s_0, a_0, s_1, a_1, \ldots)$ with $s_0 \sim \mathcal{I}(\theta)$, $a_t \sim \pi(\theta, s_t)$, and $s_{t+1} \sim \mathcal{T}(\theta, s_t, a_t)$. Similarly, a policy induces a distribution over *state trajectories* of the form $\tau = (s_0, s_1, \ldots)$. We denote by T the set of state trajectories with positive probability under some policy.

Given a reward-free UMDP M and a level $\theta \in \Theta$, let $R : S \times A \times S \to \mathbb{R}$ be a reward function, and let $\gamma \in [0, 1]$ be a discount factor. We define the *return (with respect to R)*, $U^R(\tau; \theta)$, as the discounted reward collected across trajectory τ, $U^R(\tau; \theta) = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})$. We define the expected return of a policy π under R as $U^R(\pi; \theta) = \mathbb{E}_\tau \big[U^R(\tau; \theta) \big]$ with the expectation taken over this induced distribution. Without loss of generality, we assume reward functions are normalized such that the returns lie in the range $[0, 1]$. Given a distribution over levels $\Lambda \in \Delta(\Theta)$, we further define the expected return of a policy over a UMDP as $U^R(\pi; \Lambda) = \mathbb{E}_{\theta \sim \Lambda} \big[U^R(\pi; \theta) \big]$.

Let $\varepsilon \geq 0$ be an approximation threshold. Given a reward function R and a level θ we define the *approximately optimal policy set* as $\Pi^\star_\varepsilon(R, \theta) = \{\pi : U^R(\pi; \theta) \geq \max_{\pi'} U^R(\pi'; \theta) - \varepsilon\}$. Analogously, given a distribution $\Lambda \in \Delta(\Theta)$, define $\Pi^\star_\varepsilon(R, \Lambda) = \{\pi : U^R(\pi; \Lambda) \geq \max_{\pi'} U^R(\pi'; \Lambda) - \varepsilon\}$.

2.2 Unsupervised environment design

The standard method of training in an UMDP is to use *domain randomization (DR),* training on levels $\theta \sim \Lambda$ sampled independently from a fixed level distribution $\Lambda \in \Delta(\Theta)$. This leads to a policy that maximizes the expected return given the distribution Λ. Formally, given a fixed training distribution Λ and reward function R, DR seeks a policy according to the objective $\max_{\pi \in \Pi} \mathbb{E}_{\theta \sim \Lambda} \big[U^R(\pi; \theta) \big]$.

Unsupervised environment design (*UED;* Dennis et al., 2021) proposes training in UMDPs via a two-player game, where an agent is trained on levels selected by an adversary. In *regret-based UED,* the agent tries to minimize *expected regret* while the adversary tries to maximize it, where the *regret* on a level is the shortfall of return achieved by the policy compared to an optimal policy, $\mathcal{G}^R(\pi; \theta) = \max_{\pi'} U^R(\pi'; \theta) - U^R(\pi; \theta)$. This is a zero-sum game, and at the Nash equilibrium, the agent plays a minimax expected regret (MMER) policy, achieving $\min_{\pi \in \Pi} \max_{\Lambda \in \Delta(\Theta)} \mathbb{E}_{\theta \sim \Lambda} \big[\mathcal{G}^R(\pi; \theta) \big]$.

3 PROBLEM SETTING

When the true goal and the proxy goal perfectly correlate on the entire training distribution, it is impossible to distinguish between polices optimizing for either goal. Hence, we propose to study goal misgeneralization in a relaxed setting with a *nearly ambiguous* training distribution—we assume that a small subset of training levels provide a weak training signal that disambiguates the true goal from the proxy goal. This relaxation mirrors the assumptions made in previous work on spurious correlations in supervised learning (Liu et al., 2021; Zhang et al., 2022).

To formalize our setting, we next introduce and define the concepts of proxy goals, what it means for a level to be disambiguating or ambiguating, and of a *proxy-distinguishing distribution shift*.

Definition 3.1 (Proxy goal)**.** *Given a reward-free UMDP M, a level distribution $\Lambda \in \Delta(\Theta)$, and a (true) reward function R, we say a reward function $\tilde{R} : S \times A \times S \to \mathbb{R}$ is a* proxy goal *if there exists a* proxy policy *$\tilde{\pi}$ which is approximately optimal with respect to both the given goal and the proxy goal: $\exists \tilde{\pi} \in \Pi^{\star}_{\varepsilon}(R, \Lambda) \cap \Pi^{\star}_{\varepsilon}(\tilde{R}, \Lambda)$.*

Definition 3.2 (Ambiguating and disambiguating levels)**.** *Given a reward-free UMDP, an approximation threshold $\epsilon \geq 0$, and two reward functions R and $\tilde{R}$, we define the following two classes of levels.*

1. *A level $\theta \in \Theta$ is* (goal) disambiguating *if optimizing either reward function leads to sub-optimal return from the other, that is: $\Pi^{\star}_{\varepsilon}(R, \theta) \cap \Pi^{\star}_{\varepsilon}(\tilde{R}, \theta) = \emptyset$.*
2. *A level $\theta \in \Theta$ is* (goal) ambiguating *if all approximately optimal policies with respect to one goal are approximately optimal with respect to the other as well: $\Pi^{\star}_{\varepsilon}(R, \theta) = \Pi^{\star}_{\varepsilon}(\tilde{R}, \theta)$.*

We note that some levels may be neither ambiguating nor disambiguating.

Definition 3.3 (Proxy-distinguishing distribution shift)**.** *Consider a UMDP, a pair of reward functions R and $\tilde{R}$, a pair of level distributions $\Lambda^{\text{Train}}, \Lambda^{\text{Test}} \in \Delta(\Theta)$, and a ratio $\alpha \in [0, 1]$. A distribution shift from Λ^{Train} to Λ^{Test} is* proxy distinguishing *if*

1. *Λ^{Train} has probability α on disambiguating levels and the rest on ambiguating levels, and*
2. *Λ^{Test} has probability 1 on disambiguating levels.*

We focus on the setting where α is small, so that the training distribution is nearly entirely ambiguous. In this setting, there exists a proxy policy $\tilde{\pi} \in \Pi^{\star}_{\varepsilon}(R, \Lambda^{\text{Train}}) \cap \Pi^{\star}_{\varepsilon}(\tilde{R}, \Lambda^{\text{Train}}) \cap \Pi^{\star}_{\varepsilon}(\tilde{R}, \Lambda^{\text{Test}})$ that optimizes $\tilde{R}$ on disambiguating levels. This proxy policy is approximately optimal under R on Λ^{Train} (based on its good behavior in ambiguating levels), however, it performs sub-optimally with respect to the true goal R after the distribution shift to Λ^{Test}. If an RL system learns the proxy policy $\tilde{\pi}$ and then is subject to this distribution shift, this is an example of goal misgeneralization.

4 THEORETICAL RESULTS

In this section, we present our theoretical results demonstrating that, compared to training on a fixed level distribution, regret-based prioritization of levels can mitigate goal misgeneralization. For clarity, we discuss this result in two parts.

In Section 4.1, we characterize when goal misgeneralization has negative consequences, namely when optimizing the proxy goal uses all of some abstract "resource" that could have been put towards optimizing the true goal. Since these consequences arise from the agent optimizing an incorrect objective function, this theory takes inspiration from prior work on the consequences of objective misspecification (Zhuang & Hadfield-Menell, 2020). However, we adapt and generalize these arguments to the sequential decision-making context.

In Section 4.2, we show that, under the conditions developed in Section 4.1, maximizing the expected value on the training distribution of levels can result in goal misgeneralization. On the other hand, we show that a policy that is approximately optimal with respect to the MMER objective mitigates the negative effects of goal misgeneralization. Intuitively, this is because these negative consequences generate regret and the adversary will incentivize the agent to avoid these negative effects.

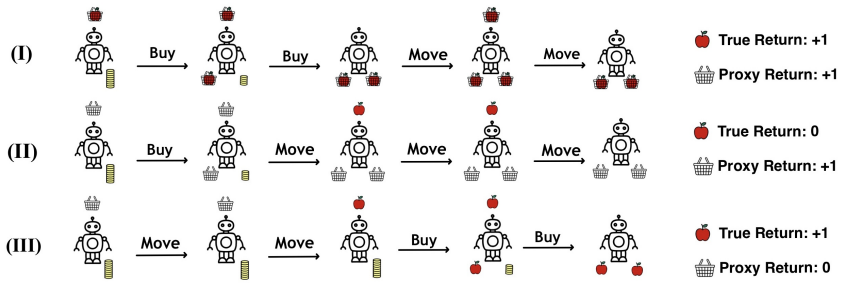

Figure 1: Example illustrating the concepts introduced in Section 3 and 4. At each time step, the agent is presented with an apple, a basket, both, or neither. The true reward R and the proxy $\tilde{R}$ correspond to buying apples and basket respectively. An agent has a certain amount of resources (money), which can be allocated towards R or $\tilde{R}$. (I) represent an ambiguating level. (II) is a disambiguating level where the agent fails. (III) is the same disambiguating level where the agent generalizes correctly.

4.1 Goal misgeneralization and its consequences

In this section we provide criteria which, if true about a domain with respect to a specific true goal R and proxy goal $\tilde{R}$, imply that goal misgeneralization by optimizing $\tilde{R}$ will lead to consequences that are negative as measured under R. Intuitively, this can be thought of as the consequences of optimizing any misspecified objective, as has been studied previously by Zhuang & Hadfield-Menell (2020). Our argument follows a similar structure to theirs—arguing that optimizing a misspecified objective, whose performance is constrained by a limited resource, will be forced to take that resource away from optimizing the true objective. We generalize this argument to the sequential RL setting, and characterize these resources as a feature of the local state trajectory.

We thus think of the negative consequences of pursuing the wrong objective as a misallocation of the agent's *resource*, whether it some physical resource explicit in the state, time in discounted MDPs, or simply the the opportunity cost of taking one action instead of another. At the outset of an episode the agent posseses some amount of this resource, and decides how to allocate it throughout the episode in order to achieve a high return. We always consider *limited* resources—once consumed, a resource cannot be regained. Goal misgeneralization has negative consequences when a resource cannot be allocated towards both rewards simultaneously and the agent incorrectly exhausts the resource in the pursuit of the proxy goal.

As a running example, consider Figure 1. The agent begins with a finite number of coins, which are the limited resource. At each time step, the agent may be presented with either apples, baskets, both, or neither and can choose from actions "buy" or "not buy". The "buy" action acquires all the objects presented. The cost for the "buy" action is always positive. The true reward function assigns +0.5 for an apple, while the proxy assigns +0.5 to the basket.

When we say that the number of coins the agent has remaining is a limited resource, we mean that it is a non-increasing function along a state trajectory.

Definition 4.1 (Limited resource)**.** *A* limited resource *is defined as* $F : S \times \mathrm{T} \to \mathbb{R}$ *such that for any state trajectory* $\tau \in \mathrm{T}$, $\forall i < j$, *it holds that* $F(s_j, \tau) \leq F(s_i, \tau)$.

While there may be many limited resources in any particular environment, such as time or materials, not all limited resources cause a trade-off between two reward functions. Intuitively, the strongest trade off between two reward functions would occur when a resource can be allocated to one reward function or the other but not both. This motivates the idea of a *resource allocation* which tracks how the resource is divided between two reward functions across a trajectory.

Definition 4.2 (Resource allocation)**.** *A* resource allocation *of a resource* F *is defined by a tuple* $\langle G_R, G_{\tilde{R}}, G, f_1, f_2 \rangle$ *where* $G_R, G_{\tilde{R}}, G : S \times A \times S \to \mathbb{R}_{\geq 0}$ *are functions describing how the resources allocated at* $S \times A \times S$ *are divided amongst reward functions, and where* $f_1, f_2 : S \times A \times S \times \mathbb{R} \to \mathbb{R}$, *are conversion functions, which describe how spent resources correspond to increased value for their reward.*

This tuple must satisfy the following conditions:

- f_1, f_2 *are monotonically non-decreasing in their last input arguments.*
- $G_R(s_t, a_t, s_{t+1}) + G_{\tilde{R}}(s_t, a_t, s_{t+1}) + G(s_t, a_t, s_{t+1}) = F(s_t, \tau) - F(s_{t+1}, \tau)$
- $\sum_{t=0}^{|\tau|-1} f_1(s_t, a_t, s_{t+1}, G_R(s_t, a_t, s_{t+1})) = U_R(\tau)$
- $\sum_{t=0}^{|\tau|-1} f_2(s_t, a_t, s_{t+1}, G_{\tilde{R}}(s_t, a_t, s_{t+1})) = U_{\tilde{R}}(\tau)$

For example, in Figure 1(II) the resource is the number of coins remaining, and in the first two time steps the resource is allocated only to the proxy reward of buying baskets, with a conversion rate of 4 coins per basket. In Figure 1(III), the resources are instead allocated to the true reward of buying apples at a rate of 4 coins per apple. In Figure 1(I), where the agent can only ever buy baskets and apples simultaneously, there are many valid resource allocations. For instance, every buy action could allocate 2 coins to the true reward of apples and 2 coins to the proxy reward of baskets, each at an exchange rate of 0.5 reward per 2 coins.

There is a tension between maximizing R and $\tilde{R}$ when there is a limited resource which is *critical* for good performance with respect to R but is *marginally useful* to an agent pursuing $\tilde{R}$. Thus the agent is motivated to completely exhaust the resource in pursuit of $\tilde{R}$. For instance, in Figure 1(II), coins are needed to buy apples and do well by R, but are also useful to buy baskets and maximize $\tilde{R}$. We formalize these below.

Definition 4.3 (Critical resource)**.** *A resource is* critical *w.r.t. a reward function R and resource allocationof a resource F if*

$$\forall \pi, \tau \sim \pi \text{ s.t. } \sum_{s_t \in \tau} G_R(s_t, a_t, s_{t+1}) = 0, \exists \pi' \text{ s.t. } U^R(\pi') > U^R(\pi) + C$$

for some $C > 0$.

Definition 4.4 (Marginally useful resource)**.** *A resource is* marginally useful *w.r.t. R and resource allocation of a resource F if*

$$\forall \pi, \tau \sim \pi \text{ s.t. } \sum_{s_t \in \tau} G_R(s_t, a_t, s_{t+1}) < F(s_0, \tau) - F(s_{|\tau|}, \tau), \exists \pi' \in \Pi \text{ s.t. } U^R(\pi) < U^R(\pi')$$

Finally, we argue that if there is a resource that is critical to R and marginally useful to $\tilde{R}$, then optimizing $\tilde{R}$ will result in low reward according to R.

Theorem 4.5 (C-distinguishing resource)**.** *Consider all specified MDPs with parameter θ, reward functions $R, \tilde{R}$. If there exists a resource F critical w.r.t R for some $C > 0$, and marginally useful w.r.t. $\tilde{R}$, then for any policies $\pi^R, \pi^{\tilde{R}}$ optimal w.r.t. $R, \tilde{R}$ respectively,*

$$U^R(\pi^R, \theta) - U^R(\pi^{\tilde{R}}, \theta) > C$$

*We say such a resource C-*distinguishes *$\tilde{R}$ from R.*

Proof sketch (full proof in Appendix A). Since resource is marginally useful with respect to $\tilde{R}$, an optimal policy with respect to $\tilde{R}$ will use all resource towards reward function $\tilde{R}$. However, since resource is critical for R, $\pi^{\tilde{R}}$ will achieve sub-optimal return under R. □

Returning to the running example, if the agent is pursuing baskets, as long as there is a way to spend coins to buy more of them, even for a significant cost, then coins are marginally useful for $\tilde{R}$. Thus the agent would want to spend all the coins on buying baskets. However, as long as there are many apples available to buy that do not come with those baskets, the coins are critical for R—the agent will not have any coins left to spend on the apples, resulting in low reward under R.

4.2 MINIMAX EXPECTED REGRET MITIGATES GOAL MISGENERALIZATION

In the last section we showed that, when there is a resource that distinguishes $\tilde{R}$ from R, optimal policies for the proxy will perform poorly under the true reward. In this section we will use this to study the ability of MMER to mitigate the consequences of goal misgeneralization. We will do this by presenting general conditions under which goal misgeneralization can have negative consequences and then show that an MMER policy will avoid these consequences.

Intuitively, consequential goal misgeneralization can occur when there is a proxy-distinguishing distribution shift and the disambiguating levels contain a resource that distinguishes $\tilde{R}$ from R to a sufficient degree. Rather than modeling the reasons why an optimization algorithm might prefer a proxy policy over a correctly generalizing policy, we characterize the possibility for some optimizer to select a proxy policy. Along these lines, we show that (A) there exists a proxy policy that is about optimal under DR, indicating a risk that DR will suffer goal misgeneralization; and (B) in contrast, all approximately optimal policies under the MMER objective generalize correctly, indicating that MMER training is robust to goal misgeneralization regardless of inductive biases. In other words, our result does not depend on the mechanisms by which the optimization algorithm misgeneralizes.

Theorem 4.6. *For all UMDPs, reward functions $R, \tilde{R}$, distributions $\Lambda^{\text{Train}}, \Lambda^{\text{Test}} \in \Delta(\Theta)$, and approximation thresholds $\epsilon > 0$ and $\delta \geq 0$, let π^R be any optimal policy w.r.t. R. If*

- *$\exists \bar{\Theta} \subset \Theta$ s.t. $\mathbb{P}_{\Lambda^{\text{Train}}}(\bar{\Theta}) < \epsilon$ and $\mathbb{P}_{\Lambda^{\text{Test}}}(\bar{\Theta}) \geq 1 - \delta$;*
- *On levels $\theta \in \big(\text{supp}(\Lambda^{\text{Train}}) \setminus \bar{\Theta}\big)$, we have $\Pi_0^\star(R, \theta) = \Pi_0^\star(\tilde{R}, \theta)$;*
- *For all levels $\theta \in \bar{\Theta}$, M_θ has a resource which C-distinguishes $\tilde{R}$ from R;*

then

(A) $\exists \pi^{\text{DR}} \in \Pi_\varepsilon^\star(R, \Lambda^{\text{Train}})$ *s.t.* $U^R(\pi^R, \Lambda^{\text{Test}}) - U^R(\pi^{\text{DR}}, \Lambda^{\text{Test}}) > (1 - \delta) \cdot C$

(B) $\forall \pi^{\text{MMER}}, \Lambda^{\text{MER}}$ *satisfying*

(i) $\pi^{\text{MMER}} \in \Pi_\varepsilon^\star(R, \Lambda^{\text{MER}})$

(ii) $\Lambda^{\text{MER}} \in \arg\max_{\Lambda \in \Delta(\Theta)} \mathbb{E}_{\theta \sim \Lambda}\big[\, \mathcal{G}^R(\pi^{\text{MMER}}; \theta)\,\big]$

we have that $U^R(\pi^R, \Lambda^{\text{Test}}) - U^R(\pi^{\text{MMER}}, \Lambda^{\text{Test}}) \leq \epsilon$

The proof can be found in Appendix A.

To return to our running example, in a setting where the apples are almost always sold in baskets, it would be approximately optimal for the agent to learn a policy which always buys baskets. However, in the transfer levels in Figure 1 (II) and (III), coins are a resource which distinguishes $\tilde{R}$ from R. Thus by Theorem 4.5, goal misgeneralization has negative consequences (in this case the agent buying baskets and no apples). Therefore, by Theorem 4.6(A) there is a risk that a policy trained by DR will misgeneralize and achieve low return in the transfer levels.

In contrast, if levels are prioritized to maximize regret, then Theorem 4.6(B) shows that the only policies even approximately optimal in these high-regret levels have to behave in a way near-optimal under R (in this case buying apples rather than baskets).

5 EXPERIMENTS

Regret-based UED methods have been motivated by their potential to improve sample efficiency and robustness. We have shown that, in theory, the MMER objective is also well-suited to mitigating goal misgeneralization in the face of a proxy-distinguishing distribution shift. In this section, we validate that existing UED methods are empirically capable of mitigating goal misgeneralization, showing that their level-generating adversaries are able to locate the region of the level space containing disambiguating levels, recognize when the policy has high regret in those levels, and amplify the weak training signal provided by these levels. We call this the *amplification effect*.

We construct two custom procedurally-generated grid-world environments that exhibit a proxy-distinguishing distribution shift. For each environment we construct a distribution of ambiguating levels Λ_1, a distribution of (primarily) disambiguating levels Λ_2, and from these a training distribution $\Lambda_\alpha^{\text{Train}} = (1-\alpha)\Lambda_1 + \alpha\Lambda_2$. We select training levels with DR in addition to a range of UED methods described in Section 5.1. We describe the environment constructions in Sections 5.2 and 5.3.

For all training runs we use a network architecture based on that of IMPALA (Espeholt et al., 2018) with a dense feed-forward layer replacing the LSTM block (following Langosco et al., 2022). We truncate rollouts at a large finite time horizon and perform policy updates with PPO (Schulman et al., 2017) and GAE (Schulman et al., 2015). We document all hyperparameters in Table 1.

5.1 Unsupervised environment design methods

We train policies with DR and also with the following two existing regret-based UED methods.

1. *PLR$^\perp$* (*robust prioritized level replay;* Jiang et al., 2022): Explores the level space by sampling levels from an underlying level distribution, estimating the regret of the current policy on each level, and keeping high-regret levels in a finite *level buffer*. The policy is trained on levels sampled from the buffer rather than from the underlying distribution.
2. *ACCEL* (*adversarially compounding complexity by editing levels;* Parker-Holder et al., 2023): Extends PLR$^\perp$ by, in addition to occasionally sampling new levels from an underlying level distribution, also exploring the level space in the neighborhood of the levels already in the buffer using a *mutation operation* $\mu : \Theta \to \Delta(\Theta)$ to generate similar levels. As with PLR$^\perp$, we keep high-regret levels in a level buffer used to train the policy.

Both PLR$^\perp$ and ACCEL involve estimating regret. Several estimators were previously proposed by Jiang et al. (2022), but we found that they were outperformed by a simple and flexible estimator which we call the *max-latest* regret estimator, defined as

$$\hat{\mathcal{G}}^R_{\text{max-latest}}(\pi;\theta) = \hat{U}^R_{\max}(\theta) - \hat{U}^R_{\text{latest}}(\pi;\theta)$$

where $\hat{U}^R_{\max}(\theta)$ is the highest empirical return ever achieved for level θ throughout training (during rollouts collected with the current policy or any previous policy since the start of training), and $\hat{U}^R_{\text{latest}}(\pi;\theta)$ is the empirical average return achieved on θ in the latest batch of rollouts with the current policy π only. We use this regret estimator for both PLR$^\perp$ and ACCEL.

ACCEL additionally requires specifying a mutation operation that makes n various environment-specific edits to the level (e.g. moving walls or changing the agent's starting position; n is a hyperparameter). We consider three distinct classes of mutation operations that differ primarily in the way they affect the balance between ambiguating and disambiguating levels, as follows.

1. *Constant goal mutation operation* (ACCEL$_\text{c}$): We make $n-1$ random edits that do not affect goal ambiguity, followed by one edit that transforms the level into a disambiguating version of the level with probability α or an ambiguating version with probability $1-\alpha$ (irrespective of whether the input level is disambiguating or ambiguating). Applying this operation to any distribution of levels results in a distribution with the same proportion of disambiguating and ambiguating levels as the training distribution.
2. *Identity goal mutation operation* (ACCEL$_\text{id}$): We make n random edits that do not affect the goal ambiguity of the level, and no other edits. Applying this operation to any distribution of levels results in a distribution with the same proportion of disambiguating and ambiguating levels as the input distribution.
3. *Binomial goal mutation operation* (ACCEL$_\text{bin}$): We make a fixed number of edits n where each edit is independently chosen to be either a random edit that does not affect goal ambiguity (with probability $1-\frac{1}{n}$) or otherwise (with probability $\frac{1}{n}$) an edit that transforms the level into a disambiguating version of the level with probability α or an ambiguating version with probability $1-\alpha$. This has essentially the same effect as ACCEL$_\text{c}$, except in the event that no goal ambiguity edits are sampled (with probability $(1-\frac{1}{n})^n$) in which case it has the same effect as ACCEL$_\text{id}$.

Thus we experiment with a total of five methods: DR, PLR$^\perp$, ACCEL$_\text{c}$, ACCEL$_\text{id}$, and ACCEL$_\text{bin}$.

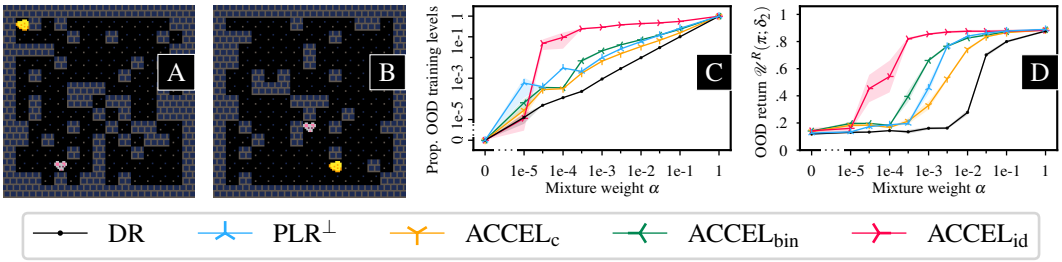

Figure 2: Cheese in the corner. We construct training distributions with both ambiguating levels (e.g., A) and disambiguating levels (e.g., B). We vary the proportion of disambiguating levels in the training distribution and report both an average of the proportion of disambiguating levels sampled during training (C) and the generalization performance on a fixed batch of disambiguating levels after training on approximately 250 million environment steps (D). We plot the mean value over three seeds with the shaded region marking one standard error (individual runs in Appendix B.1). Except for the vertical axis in (D) we use clipped log scales with values below a given threshold labeled 0.

5.2 Environment 1: Cheese in the corner

This environment is inspired by the Maze 1 environment from Langosco et al. (2022). The agent navigates a maze as a mouse looking for cheese. Observations are Boolean grids with one channel for the maze layout and one each for the mouse and cheese positions. The true goal assigns $+1$ reward when the mouse reaches the cheese, while the proxy goal assigns $+1$ reward the first time the mouse reaches the top left corner. Given these goals, levels with the cheese in the top left corner are ambiguating and levels with the cheese away from the corner are disambiguating.

We procedurally generate levels by randomly placing walls and the mouse. For ambiguating levels we place the cheese in the top left corner (e.g. Figure 2(A)). We generate disambiguating levels by randomly placing the cheese anywhere (e.g. Figure 2(B)). Our mutation operations delete or insert a number of walls and sometimes randomize the position of the mouse, and may toggle the level's goal ambiguity by moving the cheese into or out of the corner, as described in Section 5.1.

Figure 2(D) shows that DR is susceptible to goal misgeneralization until the training distribution has 3–10% of its mass on disambiguating levels. In contrast, all of the UED methods exhibit the amplification effect (C) and correct goal misgeneralization (D) at a much lower α. The best performing UED algorithm, ACCEL$_{\text{id}}$ is capable of fully correcting goal misgeneralization at a very low $\alpha = 0.03\%$. Note that some of the levels in the disambiguating test distributions are unsolvable, hence why none of the methods achieve perfect return.

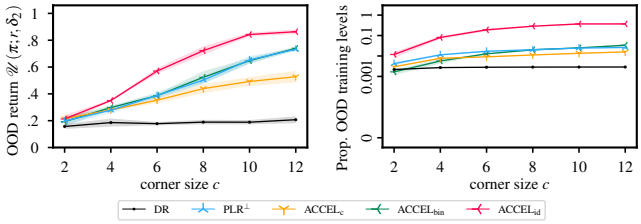

Figure 3: Robustness of methods in *cheese in the corner*. The respective agents are trained on a distribution comprised of ambiguating levels (cheese spawns in the corner) and disambiguating levels where the cheese spawn with a restricted corner size of c. α is set to 0.3%. Runs are averaged across 3 seeds. *Left*: Return on disambiguating levels where the cheese spawns anywhere in the maze. *Right*: Proportion of disambiguating levels sampled during training.

In Figure 3, we design an experiment to investigate the robustness of the agents trained via the methods considered. We train our agent with the disambiguating mixture weight $\alpha = 0.03\%$. However, we do not train on 'fully disambiguating' levels. The training distribution comprised of ambiguating levels and 'restricted' disambiguating levels where the cheese spawns in a top-left corner region of size c times c. The return is evaluated on fully disambiguating levels, where the cheese spawns anywhere in the maze. We observe in Figure 3 (left) that all of the UED methods are able to highly benefit from even a lower training signal, while DR achieves very low return across all corner sizes c. All of the UED methods also exhibit the amplification effect.

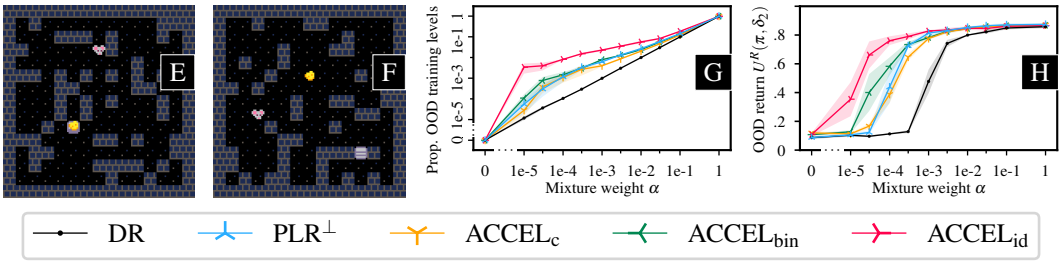

Figure 4: Cheese on a dish. We train with both ambiguating levels (e.g., E) and disambiguating levels (e.g., F). We vary the proportion of disambiguating levels and report the proportion of disambiguating levels sampled during training (G) and the generalization performance on a fixed batch of disambiguating levels after training for approx. 500 million environment steps (H). Means over three or more runs, shaded region marks one standard error (individual runs in Appendix B.1). Except for the vertical axis in (H) we use clipped log scales with values below a given threshold labeled 0.

5.3 Environment 2: Cheese on a dish

This environment is inspired by the Maze 2 environment from Langosco et al. (2022). This time the mouse navigates a maze containing both cheese and a secondary object—a dish. The true goal assigns $+1$ reward for reaching the cheese, while the proxy goal assigns $+1$ reward for reaching the dish. Episodes terminate when the mouse hits either object (or after a fixed time horizon). Levels with the cheese and dish co-located are ambiguating, and levels with the cheese and dish separated are disambiguating. The observations are Boolean grids with six additional channels redundantly coding the dish position (breaking symmetry to elicit a clearer case of goal misgeneralization).

We procedurally generate levels by randomly placing walls, the mouse, and the dish. For ambiguating levels, we place the cheese on the dish (e.g. Figure 4(E)). For disambiguating levels we randomly place the cheese and the dish independently (e.g. Figure 4(F)). Our mutation operations delete or insert a number of walls and sometimes randomize the position of the mouse, and may toggle the level's goal ambiguity by moving the cheese onto or away from the dish, as described in Section 5.1.

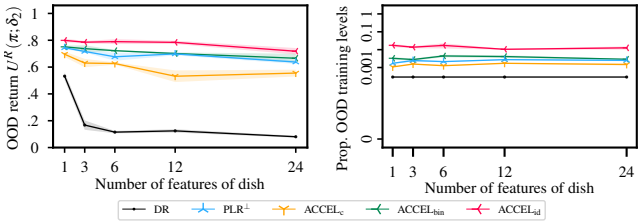

Figure 5: Robustness of our methods in the Cheese on a dish. The respective agents are trained on a distribution of levels with p features encoding the dish position. α is set to 0.03%. *Left*: Return on disambiguating levels. *Right*: Proportion of disambiguating levels sampled during training.

Figure 4(H) shows that DR is susceptible to goal misgeneralization unless the training distribution has around 0.3% of its mass on disambiguating levels. In contrast, all UED methods exhibit a strong amplification effect (G), and reduce goal misgeneralization (H) at significantly lower α. ACCEL$_{\text{id}}$ corrects goal misgeneralization at a notably low $\alpha = 0.01\%$.

In Figure 5, we vary the inductive bias by changing the number of channels coding the dish position. The UED methods are significantly more robust than DR. Additional channels significantly increase DR's susceptibility to goal misgeneralization. On the other hand, UED methods retain comparably similar performance. The amplification effect appears essentially constant. We hypothesize that the number of channels does not affect the ability for the UED methods to notice high-regret disambiguating levels, though it does affect the tendency for the policy to respond to them.

6 Related work

Unsupervised environment design (UED). Our work shows that UED methods are powerful tools to improve *inner alignment* of reinforcement learning agents. While it is well-known that UED methods improve capabilities generalization and general robustness of RL agents, this work is the first

demonstration of their power for improving alignment of RL agents as well. UED was formalized by Dennis et al. (2021). Following works have since proposed additional UED algorithms, like PLR$^{\perp}$ and ACCEL used in this work (Jiang et al., 2022; Parker-Holder et al., 2023).

Goal misgeneralization. Alignment of reinforcement learning agents is a very challenging problem. A distinction can be made between *outer alignment* and *inner alignment* (Ngo et al., 2023). Inner misalignment occurs when an agent fails to robustly pursue the preferences of the principal despite *correct* specification of the reward function (Hubinger et al., 2019). Goal misgeneralization can be considered a type of inner alignment failure. Inner alignment failures can also arise due to lack of adversarial robustness (Lu et al., 2023). Compared to outer alignment, very few works exist that are explicitly focused on assuring inner alignment of RL agents. Specific to goal misgeneralization, Langosco et al. (2022) and Shah et al. (2022) demonstrated the occurance of goal misgeneralization in various setups, but do not propose any methods to mitigate it. Starace (2023) approaches goal misgeneralization as 'task-underspecification' and conditions decision transformer style models on natural language descriptions of goals instead of scalar reward values. This is a complementary approach aimed at influencing the inductive biases in RL to favor the true goal in more circumstances.

Underspecification. Goal misgeneralization in some sense arises from underspecification present in the training setup (Shah et al., 2022). When a machine learning problem is underspecified, multiple models may exist that behave similarly on in-distribution data but behave in qualitatively different ways on out-of-distribution data (Teney et al., 2022). Prior works have discussed how underspecification can lead to erroneous evaluation of reinforcement learning (Jayawardana et al., 2022), and machine learning models in general (D'Amour et al., 2022). Underspecification underlying goal misgeneralization is also related to the identifiability of reward functions. Goal misgeneralization occurs when there exist proxies that highly correlate with the true reward function on the training domain, thus, posing a challenge for the learning agent to correctly identify the *right* reward function. A reward function is generally not identifiable from behavioral data within a single environment (Ng et al., 2000; Skalse et al., 2023; Schlaginhaufen & Kamgarpour, 2023). However, in multi-environment setups, it is sometimes possible to uniquely identify the reward function (Amin et al., 2017; Cao et al., 2021; Büning et al., 2022; Rolland et al., 2022).

7 Conclusions

We studied goal misgeneralization arising from a nearly ambiguous training distribution followed by a proxy-distinguishing distribution shift. In this setting, we have contributed a combination of theoretical and empirical results characterizing the ability of the minimax expected regret (MMER) objective and regret-based unsupervised environment design (UED) methods to mitigate goal misgeneralization. Our results also highlight the pitfalls of training with fixed level distributions.

Our theoretical results have provided a precise characterization of the potential negative consequences of goal misgeneralization in sequential decision-making. Using this characterization, we have proven the first theoretical guarantees of the effectiveness of different training methods in avoiding these negative consequences in the setting of proxy-distinguishing distribution shifts.

We have also empirically established that existing regret-based UED methods are capable of amplifying weak training signals favoring correct goal generalization via the amplification effect. These results signal UED's potential as a defense against goal misgeneralization—the amplification effect is in a position to become increasingly powerful as UED methods improve since more successful UED methods should be even more capable of detecting and amplifying rare, high-regret levels. Moreover, while prioritization-based UED methods like PLR$^{\perp}$ are confined to the support of the training distribution, ACCEL is only limited by the span of its mutation operations, and more powerful UED methods could in principle surface disambiguating levels from anywhere in the level space.

Future work can address the setting of fully ambiguous training distributions—for example, distribution shifts beyond the reach of the UED algorithm or even beyond the level space. In this case, we again face the challenge of understanding the internal dynamics of our learning algorithms and our learned policies. We are hopeful that our work will instigate further research on the problem of goal misgeneralization, which remains a critical, open problem in alignment and safe generalization of reinforcement learning agents.

REFERENCES

Kareem Amin, Nan Jiang, and Satinder Singh. Repeated inverse reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

Thomas Kleine Büning, Anne-Marie George, and Christos Dimitrakakis. Interactive inverse reinforcement learning for cooperative games. In *International Conference on Machine Learning*, pp. 2393–2413. PMLR, 2022.

Haoyang Cao, Samuel Cohen, and Lukasz Szpruch. Identifiability in inverse reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12362–12373, 2021.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2048–2056. PMLR, 2020.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research*, 23(226):1–61, 2022.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design, 2021.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Evan Hubinger, Chris van Merwijk, Vladimir Mikulik, Joar Max Victor Skalse, and Scott Garrabrant. Risks from learned optimization in advanced machine learning systems, 2019.

Vindula Jayawardana, Catherine Tang, Sirui Li, Dajiang Suo, and Cathy Wu. The impact of task underspecification in evaluating deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:23881–23893, 2022.

Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design, 2022.

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.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

Lauro Langosco Di Langosco, Jack Koch, Lee D Sharkey, Jacob Pfau, and David Krueger. Goal misgeneralization in deep reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 12004–12019. PMLR, July 2022.

Evan Z Liu, Behzad Haghgoo, Annie S Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. Just train twice: Improving group robustness without training group information. In *International Conference on Machine Learning*, pp. 6781–6792. PMLR, 2021.

Chris Lu, Timon Willi, Alistair Letcher, and Jakob Nicolaus Foerster. Adversarial cheap talk. In *International Conference on Machine Learning*, pp. 22917–22941. PMLR, 2023.

Yecheng Jason Ma, William Liang, Hung-Ju Wang, Sam Wang, Yuke Zhu, Linxi Fan, Osbert Bastani, and Dinesh Jayaraman. Dreureka: Language model guided sim-to-real transfer. *arXiv preprint arXiv:2406.01967*, 2024.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

Fabio Muratore, Fabio Ramos, Greg Turk, Wenhao Yu, Michael Gienger, and Jan Peters. Robot learning from randomized simulations: A review. *Frontiers in Robotics and AI*, 9:799893, 2022.

Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Richard Ngo, Lawrence Chan, and Sören Mindermann. The alignment problem from a deep learning perspective, 2023.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design, 2023.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3803–3810. IEEE, 2018.

Paul Rolland, Luca Viano, Norman Schürhoff, Boris Nikolov, and Volkan Cevher. Identifiability and generalizability from multiple experts in inverse reinforcement learning. *Advances in Neural Information Processing Systems*, 35:550–564, 2022.

Leonard J. Savage. The theory of statistical decision. *Journal of the American Statistical Association*, 46(253):55–67, 1951.

Andreas Schlaginhaufen and Maryam Kamgarpour. Identifiability and generalizability in constrained inverse reinforcement learning. In *International Conference on Machine Learning*, pp. 30224–30251. PMLR, 2023.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

Rohin Shah, Vikrant Varma, Ramana Kumar, Mary Phuong, Victoria Krakovna, Jonathan Uesato, and Zac Kenton. Goal misgeneralization: Why correct specifications aren't enough for correct goals, 2022.

Joar Max Viktor Skalse, Matthew Farrugia-Roberts, Stuart Russell, Alessandro Abate, and Adam Gleave. Invariance in policy optimisation and partial identifiability in reward learning. In *International Conference on Machine Learning*, pp. 32033–32058. PMLR, 2023.

G. Starace. Addressing goal misgeneralization with natural language interfaces, 2023.

Damien Teney, Maxime Peyrard, and Ehsan Abbasnejad. Predicting is not understanding: Recognizing and addressing underspecification in machine learning. In *European Conference on Computer Vision*, pp. 458–476. Springer, 2022.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. *arXiv preprint arXiv:2408.14837*, 2024.

Michael Zhang, Nimit S Sohoni, Hongyang R Zhang, Chelsea Finn, and Christopher Ré. Correct-n-contrast: A contrastive approach for improving robustness to spurious correlations. *arXiv preprint arXiv:2203.01517*, 2022.

Simon Zhuang and Dylan Hadfield-Menell. Consequences of misaligned ai. *Advances in Neural Information Processing Systems*, 33:15763–15773, 2020.

APPENDIX

A THEORETICAL RESULTS AND PROOFS

A.1 PROOF OF THE RESOURCE THEOREM (THEOREM 4.5)

Theorem 4.5 (C-distinguishing resource)**.** *Consider all specified MDPs with parameter θ, reward functions $R, \tilde{R}$. If there exists a resource F critical w.r.t R for some $C > 0$, and marginally useful w.r.t. $\tilde{R}$, then for any policies $\pi^R, \pi^{\tilde{R}}$ optimal w.r.t. $R, \tilde{R}$ respectively,*

$$U^R(\pi^R, \theta) - U^R(\pi^{\tilde{R}}, \theta) > C$$

*We say such a resource C-*distinguishes *$\tilde{R}$ from R.*

Proof. By definition, $U^R(\pi^R)$ is the optimal utility achievable under R. Thus, we want to show that $U^R(\pi^{\tilde{R}})$ achieves low performance. By definition, it must be that since the resource is marginally useful for $\tilde{R}$, that $\pi^{\tilde{R}}$ is one of the policies under which

$$\sum_{s_t \in \tau \sim \pi^{\tilde{R}}} G_{\tilde{R}}(s_t, a_t, s_{t+1}) = F(s_0) - F(s_T)$$

But if that is the case, by the fact that the resource is critical for R, it must hold that

$$\sum_{s_t \in \tau \sim \pi^{\tilde{R}}} G_R(s_t, a_t, s_{t+1}) = 0$$

Note that the condition above implies that $\exists \pi$ such that $U^R(\pi) > U^R(\pi^{\tilde{R}}) + C$. Clearly, the optimal policy π^R is one of these. Thus, it follows that

$$U^R(\pi^R) - U^R(\pi^{\tilde{R}}) > C$$

This concludes the proof. □

A.2 PROOF OF THE DR AND MMER THEOREM (THEOREM 4.6)

Theorem 4.6. *For all UMDPs, reward functions $R, \tilde{R}$, distributions $\Lambda^{\text{Train}}, \Lambda^{\text{Test}} \in \Delta(\Theta)$, and approximation thresholds $\epsilon > 0$ and $\delta \geq 0$, let π^R be any optimal policy w.r.t. R. If*

- *$\exists \bar{\Theta} \subset \Theta$ s.t. $\mathbb{P}_{\Lambda^{\text{Train}}}(\bar{\Theta}) < \epsilon$ and $\mathbb{P}_{\Lambda^{\text{Test}}}(\bar{\Theta}) \geq 1 - \delta$;*
- *On levels $\theta \in \left(\text{supp}(\Lambda^{\text{Train}}) \setminus \bar{\Theta}\right)$, we have $\Pi_0^\star(R, \theta) = \Pi_0^\star(\tilde{R}, \theta)$;*
- *For all levels $\theta \in \bar{\Theta}$, M_θ has a resource which C-distinguishes $\tilde{R}$ from R;*

then

(A) $\exists \pi^{\text{DR}} \in \Pi_\varepsilon^\star(R, \Lambda^{\text{Train}})$ *s.t.* $U^R(\pi^R, \Lambda^{\text{Test}}) - U^R(\pi^{\text{DR}}, \Lambda^{\text{Test}}) > (1 - \delta) \cdot C$

(B) $\forall \pi^{\text{MMER}}, \Lambda^{\text{MER}}$ *satisfying*

(i) $\pi^{\text{MMER}} \in \Pi_\varepsilon^\star(R, \Lambda^{\text{MER}})$

(ii) $\Lambda^{\text{MER}} \in \arg\max_{\Lambda \in \Delta(\Theta)} \mathbb{E}_{\theta \sim \Lambda}\left[\, \mathcal{G}^R(\pi^{\text{MMER}}; \theta) \,\right]$

we have that $U^R(\pi^R, \Lambda^{\text{Test}}) - U^R(\pi^{\text{MMER}}, \Lambda^{\text{Test}}) \leq \epsilon$

Proof. We will prove each part separately.

Part A

For simplicity we assume Θ is discrete. First consider the utility function optimized by DR. Given

the probability distribution Λ^{Train}, we have (defining r^K as the expected return of a policy on a given level w.r.t. to the reward function K),

$$U^K_{\text{DR}}(\pi, \Lambda^{\text{Train}}) = \sum_{\theta \in \text{supp}(\Lambda^{\text{Train}})} \mathbb{P}_{\Lambda^{\text{Train}}}(\theta) \cdot r^K(\pi, \theta)$$

Additionally, note that the utility of DR during training can be re-written as (assuming normalized and non-negative returns)

We will prove this by constructing π^{DR} such that (A) is satisfied.

Let π^{DR} be any perfectly optimal policy w.r.t. $\tilde{R}$ in all levels.

We first need to prove that $\pi^{\text{DR}} \in \Pi^\star_\varepsilon(R, \Lambda^{\text{Train}})$. We can see that

$$\begin{aligned} U^R_{\text{DR}}(\pi^{\text{DR}}, \Lambda^{\text{Train}}) &= \sum_{\theta_i \in \left(\text{supp}(\Lambda^{\text{Train}}) \setminus \bar{\Theta}\right)} \mathbb{P}_{\Lambda^{\text{Train}}}(\theta_i) \cdot r^R(\pi^{\tilde{R}}, \theta_i) + \sum_{\theta_j \in \bar{\Theta}} \mathbb{P}_{\Lambda^{\text{Train}}}(\theta_j) \cdot r^R(\pi^{\tilde{R}}, \theta_j) \\ &\leq \sum_{\theta_i \in \left(\text{supp}(\Lambda^{\text{Train}}) \setminus \bar{\Theta}\right)} \mathbb{P}_{\Lambda^{\text{Train}}}(\theta_i) \cdot r^R(\pi^{\tilde{R}}, \theta_i) + \epsilon \\ &= \sum_{\theta_i \in \left(\text{supp}(\Lambda^{\text{Train}}) \setminus \bar{\Theta}\right)} \mathbb{P}_{\Lambda^{\text{Train}}}(\theta_i) \cdot r^R(\pi^{R}, \theta_i) + \epsilon \end{aligned}$$

where the third equality holds by our assumption that $\Pi^\star_0(R, \theta) = \Pi^\star_0(\tilde{R}, \theta)$ Thus, this satisfies the definition of $\Pi^\star_\varepsilon(R, \Lambda^{\text{Train}})$, and thus it follows that $\pi^{\text{DR}} \in \Pi^\star_\varepsilon(R, \Lambda^{\text{Train}})$.

Now, we are left to show that $U^R(\pi^R, \Lambda^{\text{Test}}) - U^R(\pi^{\text{DR}}, \Lambda^{\text{Test}}) > (1 - \delta) \cdot C$.

Rewrite the utility on test of π^{DR} as

$$U^R_{\text{DR}}(\pi^{\text{DR}}, \Lambda^{\text{Test}}) = \sum_{\theta_i \in \left(\text{supp}(\Lambda^{\text{Test}}) \setminus \bar{\Theta}\right)} \mathbb{P}_{\Lambda^{\text{Test}}}(\theta_i) \cdot r^R(\pi^{\tilde{R}}, \theta_i) + \sum_{\theta_j \in \bar{\Theta}} \mathbb{P}_{\Lambda^{\text{Test}}}(\theta_j) \cdot r^R(\pi^{\tilde{R}}, \theta_j)$$

Now recall that there exists a resource that C-distinguishes $\tilde{R}$ from R on $\theta \in \bar{\Theta}$. Then, it must be the case that on all of these levels, $U^R(\pi^R, \theta) - U^R_{\text{DR}}(\pi^{\text{DR}}, \theta) > C$. So, since this set $\bar{\Theta}$ has probability $\mathbb{P}_{\Lambda^{\text{Test}}} \geq 1 - \delta$, the set of levels $\theta_i \in \left(\text{supp}(\Lambda^{\text{Test}}) \setminus \bar{\Theta}\right)$ must have probability less or equal than δ.

Thus, since we are guaranteed of a difference of at least C in $\bar{\Theta}$, it holds that

$$U^R(\pi^R, \Lambda^{\text{Test}}) - U^R(\pi^{\text{DR}}, \Lambda^{\text{Test}}) > \sum_{\theta_j \in \bar{\Theta}} \mathbb{P}_{\Lambda^{\text{Test}}}(\theta_j) \cdot C \quad \geq (1 - \delta) \cdot C$$

This concludes the proof.

Part B

Suppose for purposes of contradiction that

$$U^R(\pi^R, \Lambda^{\text{Test}}) - U^R(\pi^{\text{MMER}}, \Lambda^{\text{Test}}) > \epsilon$$

Then by (ii)

$$\mathbb{E}_{\theta \sim \Lambda^{\text{MER}}}\left[\, \mathcal{G}^R(\pi^{\text{MMER}}; \theta) \,\right] \geq \mathbb{E}_{\theta \sim \Lambda^{\text{Test}}}\left[\, \mathcal{G}^R(\pi^{\text{MMER}}; \theta) \,\right] > \varepsilon$$

But by (i) we have

$$\mathbb{E}_{\theta \sim \Lambda^{\text{MER}}}\left[\, \mathcal{G}^R(\pi^{\text{MMER}}; \theta) \,\right] \leq \varepsilon$$

which is a contradiction. This concludes the proof for (B).

□

B ADDITIONAL EXPERIMENTS

B.1 INDIVIDUAL RUNS

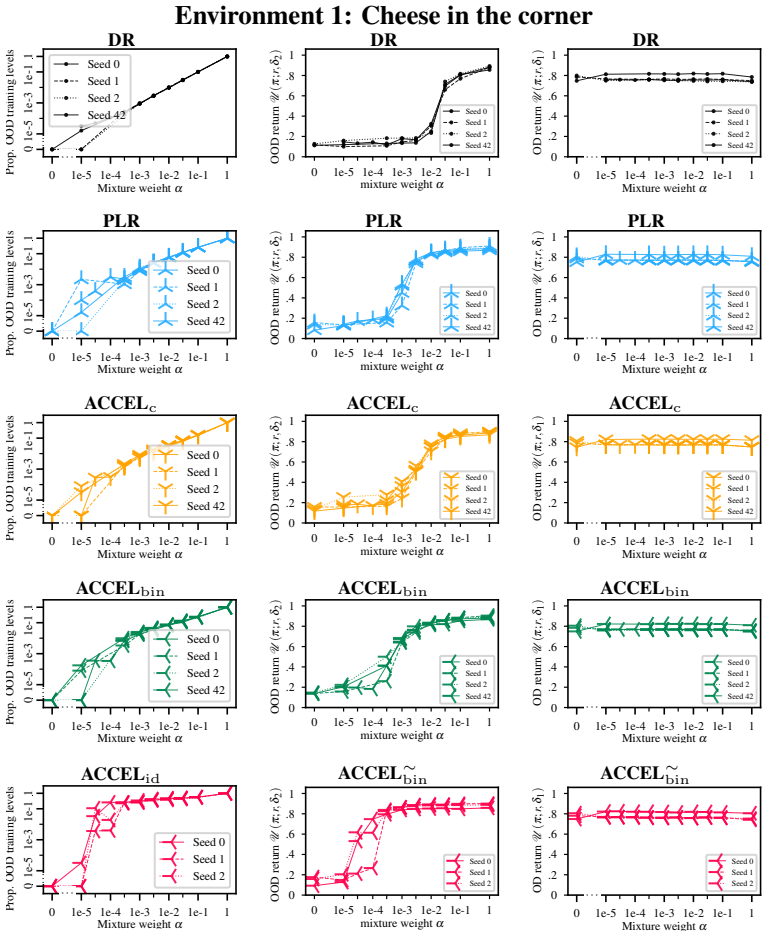

Figure 6: Individual runs for Cheese in the corner. Each row corresponds to a training algorithm, displaying the proportion of disambiguating levels during training (left), the average return on disambiguating levels (OOD levels) at the end of training (center), and the the average return on ambiguating levels (OD levels) at the end of training (right).

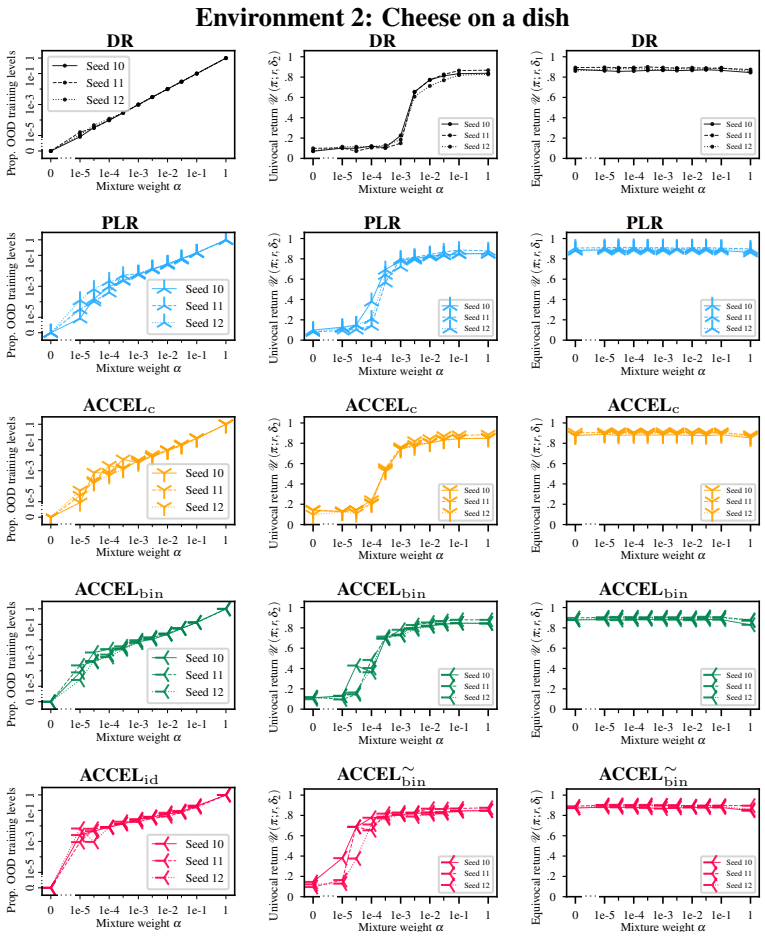

Figure 7: Individual runs for Cheese on a dish. Each row corresponds to a training algorithm, displaying the proportion of disambiguating levels during training (left), the average return on disambiguating levels at the end of training (center), and the average return on ambiguating levels at the end of training (right). The axis labels are from an old version of the terminology. Equivocal levels = ambiguating levels and univocal levals = disambiguating levels.

C ADDITIONAL ROBUSTNESS RESULTS

Cheese in the corner We provide additional experiments similar to Figure 3 under different hyperameters

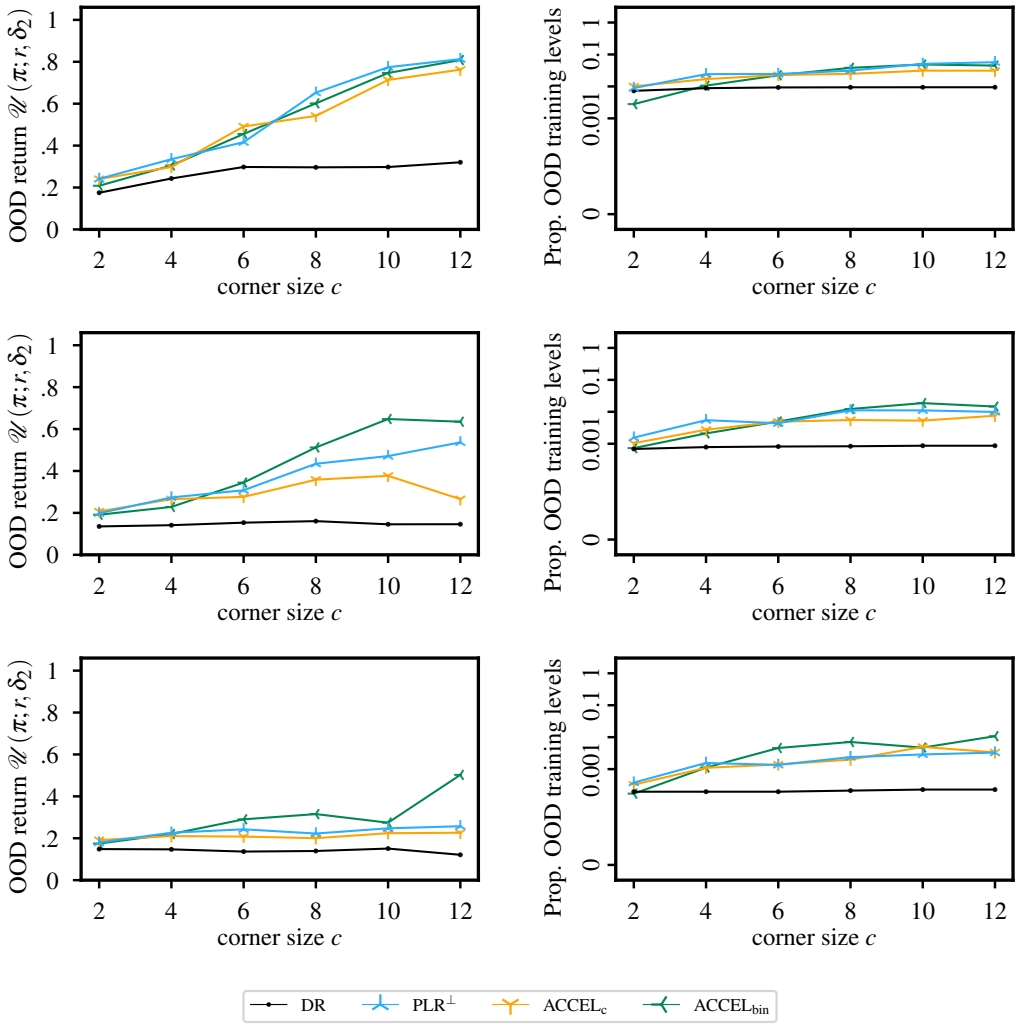

Figure 8: Additional robustness experiment, same setting as Figure 3. *Left Column*: Return on disambiguating levels. *Right Column*: Average proportion of disambiguating levels during training. *First row:* α set to 0.01. *Second row:* α set to 0.001. *Third row:* α set to 0.0003

D HYPERPARAMETERS AND IMPLEMENTATION DETAILS

We report all the relevant hyperparameters for our experiments in Table 1. The values have been chosen according to previous work, together with an exploratory analysis which considered individually the following subsets of parameters:

- learning rate in {1e-5, 5e-5, 1e-4, 5e-4}
- discount factor γ in $\{0.999, 0.99, 0.995\}$
- PPO entropy coefficient in $\{0.0001, 0.001\}$
- replay rate for ACCEL in $\{0.5, 0.7, 0.9\}$
- replay rate for PLR in $\{0.33, 0.5\}$
- regret estimators { PVL, MaxMC, max-latest }
- PLR staleness coefficient in $\{0.1, 0.3\}$
- PLR temperature in $\{0.1, 0.3\}$
- ACCEL number of mutations in $\{2, 4, 6, 12, 16, 32, 64\}$

When only the first value is compiled for a row, it indicates that all the methods used the same hyperparameter.

Table 1: Hyperparameters used for training each method.

Parameter	DR	PLR$^{\perp}$	ACCEL$_{\text{c}}$	ACCEL$_{\text{id}}$	ACCEL$_{\text{bin}}$	
PPO						
γ	0.999					
λ_{GAE}	0.95					
PPO rollout length	128					
PPO epochs	5					
PPO minibatches per epoch	4					
PPO clip range	0.1					
PPO # parallel environments	256					
Adam learning rate	5e-5					
PPO max gradient norm	0.5					
PPO value clipping	yes					
PPO critic coefficient	0.5					
PPO entropy coefficient	1e-3					
UED						
Replay rate, p	–	0.33	0.5	0.5	0.5	
Buffer size, K	–	4096	4096	4096	4096	
Regret estimator	–	max-latest	max-latest	max-latest	max-latest	
Prioritization	–	rank	rank	rank	rank	
Temperature, β	–	0.1	0.1	0.1	0.1	
Staleness coefficient	–	0.1	0.1	0.1	0.1	
ACCEL						
# Mutations per step	–	-	-	12	12	12