Mitigating Goal Misgeneralization via Minimax Regret

Karim Abdel Sadek$^{=}$, Matthew Farrugia-Roberts$^{=}$, Usman Anwar, Hannah Erlebach, Christian Schroeder de Witt, David Krueger, Michael Dennis

Keywords: Goal Misgeneralization, Unsupervised Environment Design, AI Safety, AI Alignment.

Summary

Safe generalization in reinforcement learning requires not only that a learned policy *acts capably* in new situations, but also that it uses its capabilities *towards the pursuit of the designer's intended goal.* The latter requirement may fail when a *proxy goal* incentivizes similar behavior to the intended goal within the training environment, but not in novel deployment environments. In this setting, policies may behave as if in pursuit of the proxy goal in deployment—a phenomenon known as *goal misgeneralization.* In this paper, we theoretically investigate the possibility of goal misgeneralization under *maximum expected value (MEV)* and *minimax expected regret (MMER)* objectives, and empirically validate our results. Our findings underscore minimax expected regret as a promising principle for mitigating goal misgeneralization.

Contribution(s)

1. We introduce a problem setting called a *proxy-distinguishing distribution shift,* capturing a class of situations in which goal misgeneralization can be elicited and studied.
   Context: In a proxy-distinguishing distribution shift, optimizing a given proxy goal also optimizes the true goal in most training situations, but optimizing the proxy goal can be suboptimal under the true goal in most deployment situations (in particular, so-called *distinguishing levels*). We do not assume training methods have knowledge of the proxy goal.
2. We prove that, under a proxy-distinguishing distribution shift, approximately maximizing expected value on the training distribution permits a misgeneralizing solution if the proportion of distinguishing levels in the training distribution is low enough (Theorem 1).
   Context: *Exactly* maximizing expected value on the training distribution permits misgeneralization if *no* distinguishing levels are seen in training. We model *possible* goal misgeneralization; *actual* goal misgeneralization also depends on the agent's inductive biases.
3. We prove that, under a proxy-distinguishing distribution shift, no approximate solution of the minimax expected regret objective exhibits goal misgeneralization (Theorem 2).
   Context: Theorem 2 holds for fully observable environments; we include a generalization to partially observable environments in the supplementary materials (Theorem 3).
4. Experiments suggest (no statistical significance analysis) that, under conditions approximating a proxy-distinguishing distribution shift in procedurally generated grid-world environments, policies learned using domain randomization (DR; an MEV-based training method) exhibit goal misgeneralization when the proportion of distinguishing levels in the training distribution is low enough (§7.1).
   Context: Langosco et al. (2022) demonstrated goal misgeneralization with zero distinguishing levels, we extend this finding to the case with a small positive proportion.
5. Experiments suggest (no statistical significance analysis) that, under the same conditions, existing regret-based unsupervised environment design (UED) methods, PLR$^{\perp}$ (Jiang et al., 2021a) and ACCEL (Parker-Holder et al., 2022), (1) can detect rare distinguishing levels and increase their proportion in the training distribution, and (2) are more robust to goal misgeneralization than DR is (§7.2).
   Context: In some cases, less advanced UED methods fail to find MMER policies, and still exhibit goal misgeneralization (§7.3, §7.4), indicating that more mature UED methods are needed to achieve the potential of MMER for preventing goal misgeneralization in practice.

Mitigating Goal Misgeneralization via Minimax Regret

Karim Abdel Sadek[=,1,2], **Matthew Farrugia-Roberts**[=,3],
Usman Anwar[4], **Hannah Erlebach**[5], **Christian Schroeder de Witt**[3],
David Krueger[6], **Michael Dennis**[7]
Correspondence to karimabdel@berkeley.edu, matthew@far.in.net

[1]**University of California, Berkeley**
[2]**University of Amsterdam**
[3]**University of Oxford**
[4]**University of Cambridge**
[5]**University College London**
[6]**Mila, University of Montreal**
[7]**Google DeepMind**

[=]Equal contribution

Abstract

Safe generalization in reinforcement learning requires not only that a learned policy acts capably in new situations, but also that it uses its capabilities towards the pursuit of the designer's *intended goal.* The latter requirement may fail when a *proxy goal* incentivizes similar behavior to the intended goal within the training environment, but not in novel deployment environments. This creates the risk that policies will behave as if in pursuit of the proxy goal, rather than the intended goal, in deployment—a phenomenon known as *goal misgeneralization.* In this paper, we formalize this problem setting in order to theoretically study the possibility of goal misgeneralization under different training objectives. We show that goal misgeneralization is possible under approximate optimization of the maximum expected value (MEV) objective, but not the minimax expected regret (MMER) objective. We then empirically show that the standard MEV-based training method of domain randomization exhibits goal misgeneralization in procedurally-generated grid-world environments, whereas current regret-based unsupervised environment design (UED) methods are more robust to goal misgeneralization (though they don't find MMER policies in all cases). Our findings suggest that minimax expected regret is a promising approach to mitigating goal misgeneralization.

1 Introduction

As reinforcement learning (RL) is increasingly applied in complex, open-ended, real-world environments, it is becoming infeasible for training to comprehensively cover all situations an agent will face in deployment. We therefore need training methods to produce policies that *generalize*, behaving as intended when faced with a novel scenario (Kirk et al., 2023).

A particular challenge arises when incomplete coverage of the environment space during training creates a *proxy goal.* A proxy goal is a reward function that, compared to the true goal, induces similar optimal behavior in most situations encountered during training, but induces radically different behavior in some novel situations. Proxy goals create the risk of *goal misgeneralization*—learning

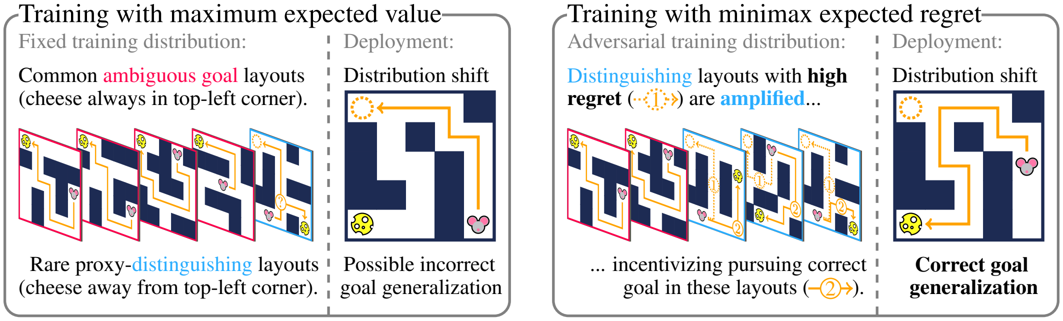
Training with maximum expected value
Fixed training distribution:
Deployment:
Common ambiguous goal layouts
(cheese always in top-left corner).
Distribution shift
Rare proxy-distinguishing layouts
(cheese away from top-left corner).
Possible incorrect
goal generalization
Training with minimax expected regret
Adversarial training distribution:
Deployment:
Distinguishing layouts with high
regret (1) are amplified...
Distribution shift
... incentivizing pursuing correct
goal in these layouts (2).
Correct goal
generalization

Figure 1: **Maximum expected value and minimax expected regret vs. goal misgeneralization.** A mouse searches a maze for cheese that is usually located in the top-left corner. There is a proxy goal (“go to the corner”) that mostly incentivizes the same optimal behavior as the true goal (“go to the cheese”). *(Left):* Standard RL methods that *approximately* maximize expected value/return could find a policy that behaves as if pursuing the proxy goal rather than the true goal, since layouts where this policy fails are rare in training. This would lead to incorrect generalization. *(Right):* If a policy ignores the cheese, a regret-maximizing adversary can move the cheese away from the corner until the agent internalizes the correct goal, leading to correct generalization.

a policy that retains its capabilities in novel situations, but behaves as if to pursue the proxy goal instead of the true goal when the two diverge (Langosco et al., 2022; Shah et al., 2022).

Goal misgeneralization can arise when such “proxy-distinguishing” situations—where the proxy goal diverges from the true goal—are rare within training, making policies that pursue the wrong goal approximately optimal in terms of the standard RL objective of maximum expected value (MEV). This motivates the need for training methods that can somehow identify proxy-distinguishing situations within a complex environment, and ensure they are adequately covered in the training distribution.

We observe that favoring the proxy goal in proxy-distinguishing situations leads to high *expected regret,* defined as the shortfall of expected return compared to that obtained by an optimal policy. An environment selected to maximize a policy’s expected regret will naturally include proxy-distinguishing situations as long as the policy ignores the true goal. Therefore, we propose mitigating goal misgeneralization via the *minimax expected regret (MMER)* objective (Savage, 1951).

In this paper, we conduct a theoretical and empirical investigation of the possibility of goal misgeneralization under the MEV and MMER objectives. An outline of our contributions is as follows.

1. In Section 4, we introduce a problem setting called a *proxy-distinguishing distribution shift*, formalizing a class of situations in which goal misgeneralization can arise.
2. In Section 5, we show formally that (1) approximately optimizing MEV is susceptible to goal misgeneralization if proxy-distinguishing situations are sufficiently rare (Theorem 1), and (2) approximately optimizing MMER is provably robust to goal misgeneralization (Theorem 2).
3. In Sections 6 and 7, we empirically study the robustness to goal misgeneralization of a standard MEV-based training method (domain randomization; Tobin et al., 2017), and recent MMER-based training methods (regret-based unsupervised environment design; Dennis et al., 2020; Jiang et al., 2021a; Parker-Holder et al., 2022) under a proxy-distinguishing distribution shift.

Our theoretical results show that, in the limit of idealized training methods, MMER-based training is guaranteed to be robust against goal misgeneralization, whereas MEV-based training is not. Our empirical results show that current MMER-based training methods are indeed more robust to goal misgeneralization than MEV-based training is, and, while they sometimes still exhibit goal misgeneralization, this happens less for more advanced methods. Together, these results establish MMER-based training as a promising approach to preventing goal misgeneralization.

2 Related work

Goal misgeneralization. Ensuring learned systems generalize as intended in novel situations is a perennial challenge for deep learning and deep RL (Kirk et al., 2023). Christiano (2018) distinguishes *benign* generalization failures, where an agent fails to behave capably in a novel situation, from *malign* generalization failures, where the agent demonstrates capable behavior towards the pursuit of an unintended objective. Langosco et al. (2022) and Shah et al. (2022) demonstrate behavioral examples of malign generalization failures in deep RL, introducing the term *goal misgeneralization.* Goal misgeneralization is similar to *shortcut learning* in supervised learning (Geirhos et al., 2020), but emphasizes shortcut *reward functions,* rather than shortcut policies (cf., Suau et al., 2024).

Recent work proposes complementary approaches to mitigating the risk of goal misgeneralization. Starace (2023) investigates influencing the agent's inductive bias in favor of correct goal generalization using goal-conditioned RL with natural language task descriptions. Trinh et al. (2024) studies methods for detecting when the agent is in an unfamiliar situation and choosing to ask an expert (at a cost) to clarify the optimal action.

Training in complex environments. The standard technique for RL in complex environments is to train on situations sampled from a fixed distribution, a technique known as domain randomization (e.g., Tobin et al., 2017; Peng et al., 2018). Maximizing expected return over such situations corresponds to pursuing the MEV objective with respect to the fixed training distribution.

Dennis et al. (2020) proposed *regret-based unsupervised environment design (UED)*, an RL training technique featuring an adversarial environment designer that continually adapts the training distribution aiming to maximize the agent's expected regret. Maximizing expected return on this adversarial distribution corresponds to the MMER objective (Dennis et al., 2020). UED has been promoted as a technique for (1) improving sample efficiency by creating an emergent curriculum; and (2) improving capability generalization via adversarial robustness (Dennis et al., 2020; Jiang et al., 2021a; Parker-Holder et al., 2022). We show that UED also helps to mitigate goal misgeneralization.

Alternative adversarial approaches, such as maximin expected value (Dennis et al., 2020; Wang et al., 2023), maximizing diversity (OpenAI et al., 2019), or maximizing learnability (Rutherford et al., 2024), have not been studied in the context of goal misgeneralization. These approaches may also mitigate goal misgeneralization to the extent that they promote training in proxy-distinguishing situations, incentivizing the agent to internalize the true goal. We show that directly optimizing the training environment for regret is sufficient. Appendix K shows that minimax expected value can exhibit goal misgeneralization when some situations have low maximum expected return.

3 Preliminaries

A (reward-free) underspecified Markov decision process (UMDP) is a tuple $M = \langle \Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$ where Θ is a space of free parameters (also called **levels**), $\mathcal{A}$ is the agent's action space, $\mathcal{S}$ is a state space, $\mathcal{I} : \Theta \to \Delta(\mathcal{S})$ is an initial state distribution, and $\mathcal{T} : \Theta \times \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is a conditional transition distribution. For simplicity, we assume Θ, $\mathcal{S}$, $\mathcal{A}$ are finite. Given a level $\theta \in \Theta$ we have a fully-specified (reward-free) MDP $\langle \mathcal{A}, \mathcal{S}, \mathcal{I}(\theta), \mathcal{T}(\theta, -, -) \rangle$. We aggregate these MDPs into a single complex environment using a **level distribution** $\Lambda \in \Delta(\Theta)$. A **reward function** (or **goal**) is a function $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$. Taken together, M and R define a proper (non-reward-free) UMDP. We define reward functions and reward-free UMDPs separately to facilitate considering multiple goals for an otherwise fixed environment. We usually denote by R and $\tilde{R}$ the **true goal** and the **proxy goal,** respectively.

An agent's **policy** is a conditional action distribution $\pi : \Theta \times \mathcal{S} \to \Delta(\mathcal{A})$. Note that we assume the policy observes the level (we consider the partially observable case in Appendix E). The set of all policies is denoted by Π. We define the **expected return** (or **expected value**) of policy π in level θ under goal R as the discounted cumulative reward $V^R(\pi; \theta) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})]$ where $\gamma \in (0, 1)$ is a discount factor and the expectation is over $s_0 \sim \mathcal{I}(\theta)$, $a_t \sim \pi(\theta, s_t)$, and

$s_{t+1} \sim \mathcal{T}(\theta, s_t, a_t)$. We lift this definition to level distributions as $V^R(\pi;\Lambda) = \mathbb{E}_{\theta\sim\Lambda}\big[\,V^R(\pi;\theta)\,\big]$. A **normalized** goal is one such that the return has support in $[0,1]$.

We define the **expected regret** of a policy π in the level θ under a goal R as the shortfall of expected value achieved by the policy compared to an optimal policy for that level,

$$G^R(\pi;\theta) = \max_{\pi'\in\Pi} V^R(\pi';\theta) - V^R(\pi;\theta). \tag{1}$$

Once again, we lift this definition to level distributions as $G^R(\pi;\Lambda) = \mathbb{E}_{\theta\sim\Lambda}\big[\,G^R(\pi;\theta)\,\big]$. Since the policy is conditioned on θ, we also have the following identity (see Appendix B):

$$G^R(\pi;\Lambda) = \max_{\pi'\in\Pi} V^R(\pi';\Lambda) - V^R(\pi;\Lambda). \tag{2}$$

4 Problem setting

Langosco et al. (2022) and Shah et al. (2022) provide case studies of several situations in which goal misgeneralization arises. In order to theoretically study goal misgeneralization, we formalize an abstract class of situations in which goal misgeneralization can arise as a problem setting called a *proxy-distinguishing distribution shift*.

4.1 Level classification

The problem setting is defined in terms of the following classification of levels, based on whether a given proxy goal incentivizes optimal or suboptimal behavior under a true goal.

Definition 1 (Proxy non-distinguishing level)**.** *Consider an UMDP $\langle\Theta,\mathcal{A},\mathcal{S},\mathcal{I},\mathcal{T}\rangle$, a true goal R, and a proxy goal $\tilde{R}$. A level $\theta\in\Theta$ is* proxy non-distinguishing *with respect to R and $\tilde{R}$ if all optimal policies with respect to $\tilde{R}$ are also optimal with respect to R, that is,*

$$\arg\max_{\pi\in\Pi} V^{\tilde{R}}(\pi;\theta) \subseteq \arg\max_{\pi\in\Pi} V^R(\pi;\theta).$$

Definition 2 (Possibly proxy C-distinguishing level)**.** *Consider an UMDP $\langle\Theta,\mathcal{A},\mathcal{S},\mathcal{I},\mathcal{T}\rangle$, a true goal R, a proxy goal $\tilde{R}$, and a constant $C\geq 0$. A level $\theta\in\Theta$ is* possibly proxy C-distinguishing *with respect to R and $\tilde{R}$ if some policy that is optimal with respect to $\tilde{R}$ achieves sufficiently suboptimal expected return with respect to R, that is,*

$$\arg\max_{\pi\in\Pi} V^{\tilde{R}}(\pi;\theta) \not\subseteq \text{arg-}C\text{-max}_{\pi\in\Pi} V^R(\pi;\theta)$$

where $\text{arg-}C\text{-max}_{\pi\in\Pi} V^R(\pi;\theta) = \{\pi\in\Pi \mid V^R(\pi;\theta) \geq \max_{\pi'\in\Pi} V^R(\pi';\theta) - C\}$.

For brevity, we usually drop the prefixes "proxy" and "possibly proxy" and refer simply to "non-distinguishing" and "distinguishing" levels. The prefix "proxy" signifies that the two goals play asymmetric roles in the definitions, because our interest is in whether training in a level disincentivizes policies that pursue the proxy goal at the expense of the true goal, and not the other way around. Proxy non-distinguishing levels never offer a training signal against optimally pursuing the proxy goal, because all such policies are necessarily also optimal under the true goal. In contrast, in proxy distinguishing levels, there exist policies that pursue the proxy goal at the expense of the true goal. These policies are disincentivized when training in these levels. Note that while this is the case for some policies that are optimal under the proxy goal, it may not be the case for a given policy, hence "possibly."

If $C=0$, the classification is exhaustive. Figure 2 gives several examples. If $C>0$, the classification is not necessarily exhaustive: there may exist levels for which it is possible to optimally pursue the proxy goal while remaining approximately optimal under the true goal. However, it is often true that optimizing a misspecified goal leads to arbitrarily low return (cf., Zhuang & Hadfield-Menell, 2020). Therefore, possibly proxy C-distinguishing levels represent an important class of levels.

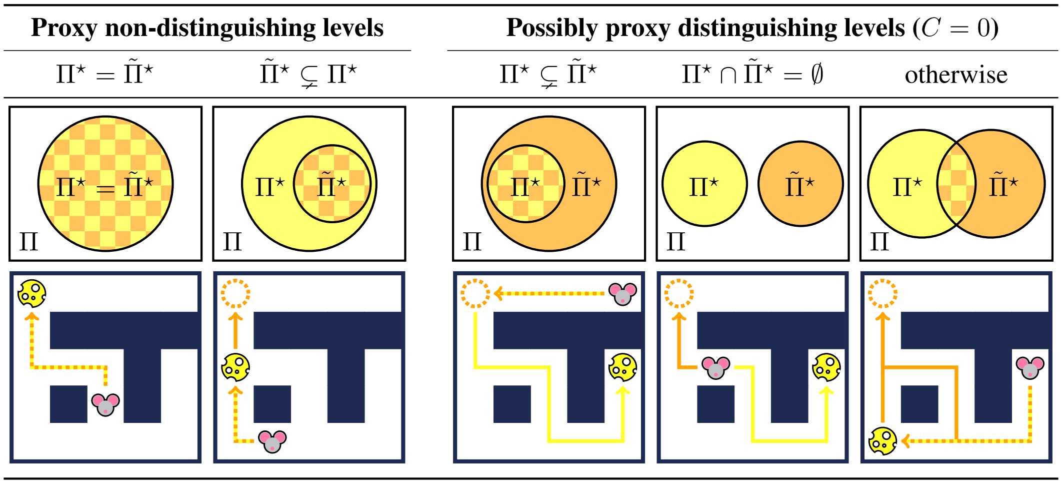
Proxy non-distinguishing levels
Possibly proxy distinguishing levels ($C = 0$)
$\Pi^\star = \tilde{\Pi}^\star$
$\tilde{\Pi}^\star \subsetneq \Pi^\star$
$\Pi^\star \subsetneq \tilde{\Pi}^\star$
$\Pi^\star \cap \tilde{\Pi}^\star = \emptyset$
otherwise
$\Pi^\star = \tilde{\Pi}^\star$
Π
$\Pi^\star$
$\tilde{\Pi}^\star$
Π
$\Pi^\star$
$\tilde{\Pi}^\star$
Π
$\Pi^\star$
$\tilde{\Pi}^\star$
Π
$\Pi^\star$
$\tilde{\Pi}^\star$
Π

Figure 2: **Illustration and examples of non-distinguishing and distinguishing levels.** A level $\theta \in \Theta$ can be classified as *non-distinguishing* or 0-*distinguishing* (Definitions 1 and 2). *(1st row):* Possible relationships between sets $\Pi^\star = \arg\max_{\pi\in\Pi} V^R(\pi;\theta)$ and $\tilde{\Pi}^\star = \arg\max_{\pi\in\Pi} V^{\tilde{R}}(\pi;\theta)$. *(2nd row):* Example levels for a navigation environment. Yellow and orange arrows show optimal behaviors for the true goal ("go to the cheese") and a proxy goal ("go to the corner"), respectively.

4.2 Proxy-distinguishing distribution shift

We model a shift from training to deployment as a change in the distribution of available levels (from a *training distribution* to a *deployment distribution*). The distribution shift is *proxy-distinguishing* when the training distribution concentrates mostly on non-distinguishing levels, but the deployment distribution concentrates mostly on distinguishing levels.

Definition 3 (Proxy-distinguishing distribution shift)**.** *Consider an UMDP $\langle\Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T}\rangle$, a true goal R, and a proxy goal $\tilde{R}$. A* proxy-distinguishing distribution shift *is a tuple $\langle\alpha, \beta, C, \Lambda^{\text{Train}}, \Lambda^{\text{Deploy}}\rangle$ where α, β are ratios such that $0 \le \alpha < \beta \le 1$, $C \ge 0$ is a constant, and $\Lambda^{\text{Train}}, \Lambda^{\text{Deploy}} \in \Delta(\Theta)$ are distributions over levels with the following classifications (with respect to R and $\tilde{R}$):*

1. *Λ^{Train} has probability α on C-distinguishing levels and the rest on non-distinguishing levels.*
2. *Λ^{Deploy} has probability β on C-distinguishing levels and the rest on non-distinguishing levels.*

We are mainly interested in the case where α is very close to zero (where goal misgeneralization is a particular risk) and β is very close to one (where goal misgeneralization is a particular concern).

4.3 Assumptions

We don't assume prior knowledge of the proxy goal or distinguishing levels. However, we *do* assume the ability to train in distinguishing levels, once identified. In practice, one can train in a very wide space of situations, whether via a simulator (Tobin et al., 2017; Peng et al., 2018; Kumar et al., 2021; Makoviychuk et al., 2021; Muratore et al., 2022; Ma et al., 2024), a generative environment model (Bruce et al., 2024), or a world model (Ha & Schmidhuber, 2018; Hafner et al., 2020; Schrittwieser et al., 2020; Hafner et al., 2025; Valevski et al., 2025). If *all* levels accessible before deployment are non-distinguishing, we may require alternative assumptions (see, e.g., Trinh et al., 2024).

Moreover, we assume access to a reliable reward signal in favor of the true goal in distinguishing levels. This mirrors assumptions made in work on spurious correlations in supervised learning (e.g., Liu et al., 2021; Zhang et al., 2022). However, in practice, reward functions may be subject to misspecification in such corner cases (cf., Hadfield-Menell et al., 2017). Future work could develop methods that treat the true goal as underspecified in rare, distinguishing levels and find ways to incentivize safe generalization behavior despite this uncertainty.

5 Theoretical results

In this section, we prove that under a proxy-distinguishing distribution shift, the maximum expected value (MEV) objective permits an approximately optimal policy that exhibits goal misgeneralization. On the other hand, we show that any policy that is approximately optimal with respect to minimax expected regret (MMER) must avoid goal misgeneralization. All proofs are in Appendix A.

We consider *approximately* optimal policies because, in practice, training uses finite optimization power and will not always find policies that are exactly optimal. We model approximate optimization as instead finding an arbitrary policy within a small threshold of optimal for the given objective. We use the notation arg-ε-$\max_{x\in\mathcal{X}} f(x) = \{x \in \mathcal{X} \mid f(x) \geq \max_{\xi\in\mathcal{X}} f(\xi) - \varepsilon\}$ (likewise, arg-ε-min) for approximate optimization of a function $f : \mathcal{X} \to \mathbb{R}$ with approximation threshold $\varepsilon \geq 0$.

5.1 Approximate maximum expected value is susceptible to goal misgeneralization

The standard objective used in RL is the maximum expected value (MEV) objective with respect to the fixed training distribution. We formalize approximate solutions under this objective as follows.

Definition 4 (Approximate MEV)**.** *Consider an UMDP $\langle \Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$, a goal R, an approximation threshold $\varepsilon \geq 0$, and a fixed level distribution $\Lambda \in \Delta(\Theta)$. The* approximate MEV policy set *with respect to Λ is then*

$$\Pi_\varepsilon^{\text{MEV}}(R, \Lambda) = \underset{\pi\in\Pi}{\text{arg-}\varepsilon\text{-max}}\, V^R(\pi; \Lambda).$$

The MEV objective permits goal misgeneralization under a proxy-distinguishing distribution shift if the proportion of distinguishing levels in training is too small. Intuitively, a policy pursuing the proxy goal in all levels achieves enough return on non-distinguishing levels to be approximately optimal. Note: rather than modeling inductive bias, we characterize the *possibility* of goal misgeneralization.

Theorem 1 (MEV is susceptible to goal misgeneralization)**.** *Consider an UMDP $\langle \Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$, a pair of normalized goals $R, \tilde{R}$, a proxy-distinguishing distribution shift $\langle \alpha, \beta, C, \Lambda^{\text{Train}}, \Lambda^{\text{Deploy}} \rangle$, and an approximation threshold $\varepsilon \geq 0$. If $\alpha \leq \varepsilon$, then there exists $\pi^{\text{MEV}} \in \Pi_\varepsilon^{\text{MEV}}(R, \Lambda^{\text{Train}})$ such that*

$$\pi^{\text{MEV}} \in \underset{\pi\in\Pi}{\arg\max}\, V^{\tilde{R}}(\pi; \Lambda^{\text{Deploy}}) \setminus \underset{\pi\in\Pi}{\text{arg-}\beta C\text{-max}}\, V^R(\pi; \Lambda^{\text{Deploy}}). \qquad \text{[proof]}$$

5.2 Approximate minimax expected regret is robust to goal misgeneralization

The MMER objective is to *minimize* the expected regret assuming an *adversarially-chosen* (*maximum* expected regret) level distribution for each policy. We consider approximate minimization against an exactly optimal adversary, but a similar robustness property holds when using an approximate adversary (the bound worsens linearly in the suboptimality of the adversary, see Appendix C).

Definition 5 (Approximate MMER)**.** *Consider an UMDP $\langle \Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$, a goal R, and an approximation threshold $\varepsilon \geq 0$. The* approximate MMER policy set *is then*

$$\Pi_\varepsilon^{\text{MMER}}(R) = \underset{\pi\in\Pi}{\text{arg-}\varepsilon\text{-min}} \max_{\Lambda\in\Delta(\Theta)} G^R(\pi; \Lambda).$$

The MMER objective does not permit goal misgeneralization under any distribution shift within the specified level space. Intuitively, the adversarial level distribution for a misgeneralizing policy would concentrate on distinguishing levels, generating high expected regret.

Theorem 2 (MMER is robust to goal misgeneralization)**.** *Consider an UMDP $\langle \Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$, a pair of goals $R, \tilde{R}$, a proxy-distinguishing distribution shift $\langle \alpha, \beta, C, \Lambda^{\text{Train}}, \Lambda^{\text{Deploy}} \rangle$, and an approximation threshold $\varepsilon \geq 0$. Then*

$$\forall \pi^{\text{MMER}} \in \Pi_\varepsilon^{\text{MMER}}(R), \text{ we have } \pi^{\text{MMER}} \in \underset{\pi\in\Pi}{\text{arg-}\varepsilon\text{-max}}\, V^R(\pi; \Lambda^{\text{Deploy}}). \qquad \text{[proof]}$$

Remarks. As a corollary, any policy that is robust against all possible distribution shifts must be an MMER policy (see Appendix D). A slightly modified bound holds for partially observable environments after accounting for the minimum expected regret *realizable by a fixed policy* (see Appendix E).

6 Experimental methods

In this section, we outline our methods for investigating the robustness to goal misgeneralization of MEV-based and MMER-based training methods. We construct three custom procedurally-generated grid-world environments approximating proxy-distinguishing distribution shifts (Section 6.1). We compare a standard MEV-based training method and two recently proposed MMER-based methods with adversaries of varying flexibility (Section 6.2), paired with regret estimators that leverage varying amounts of domain knowledge (either using the ground truth maximum return, or estimating it from samples; Section 6.3). Section 7 presents the results of our experiments.

6.1 Procedurally-generated grid-world environments

Langosco et al. (2022) exhibited goal misgeneralization in several environments from OpenAI Procgen (Cobbe et al., 2020), suitably modified to implement a proxy-distinguishing distribution shift with $\alpha = 0$. We implement three similar procedurally-generated grid-world environments in JAX (Bradbury et al., 2018), allowing us to more easily implement custom level generation and analysis.

For each environment, we construct two procedural level generators $\Lambda_{\neg\text{Distg.}}, \Lambda_{\text{Distg.}} \in \Delta(\Theta)$, approximately concentrated on non-distinguishing and distinguishing levels, respectively. From these, we define training distributions $\Lambda^{\text{Train}}_{\alpha} = (1-\alpha)\Lambda_{\neg\text{Distg.}} + \alpha\Lambda_{\text{Distg.}}$ where α is the proportion of distinguishing levels. In our experiments, we vary α from 10^{-5} to 10^{-1}, with $\alpha \in \{0, 1\}$ as baselines. We evaluate on $\Lambda^{\text{Deploy}} = \Lambda_{\text{Distg.}}$, approximating a proxy-distinguishing distribution shift.

The three environments are as follows. Figure 3 illustrates example levels (note we use Boolean observations). Appendix F comprehensively documents each environment, including the details of classifying levels as non-distinguishing or distinguishing and procedural level generation.

1. **Cheese in the corner.** A mouse navigates a maze. The true goal assigns $+1$ reward for reaching a piece of cheese, while a proxy goal assigns $+1$ reward for reaching the top left corner for the first time. Levels with the cheese in the top left corner are non-distinguishing and levels with the cheese away from the corner are (in most cases) distinguishing.
2. **Cheese on a dish.** This time the mouse navigates a maze containing cheese and also a dish. The true goal assigns $+1$ reward for reaching the cheese, while a proxy goal assigns $+1$ reward for reaching the dish. Levels with the cheese and dish co-located are non-distinguishing, and levels with the cheese and dish separated are (in most cases) distinguishing.
3. **Keys and chests.** A more complex, multi-stage task, in which the mouse navigates a maze, collects keys, and spends keys to open chests. Levels with 3 keys and 10 chests are approximately non-distinguishing. Levels with 10 keys and 3 chests are mostly distinguishing—a misgeneralizing policy would overprioritize key collection beyond what is necessary for opening chests.

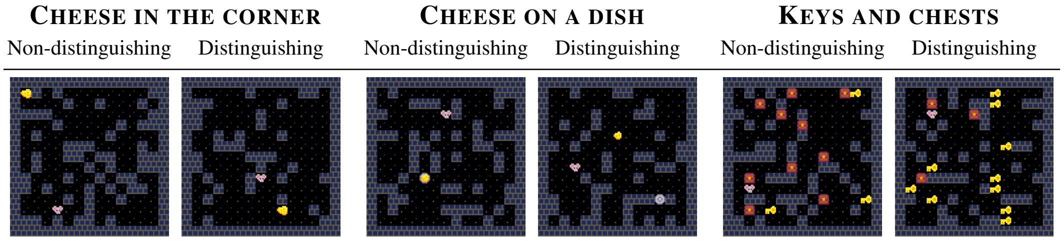

Figure 3: Example procedurally-generated non-distinguishing/distinguishing levels. The agent's observation is a $15 \times 15 \times c$ Boolean grid (where c is an environment-dependent number of channels).

6.2 Training methods

For both MEV-based and MMER-based training, we follow Langosco et al. (2022) and use an agent network architecture based on that of IMPALA (Espeholt et al., 2018) with a dense feed-forward layer replacing the LSTM block. We perform policy updates with PPO (Schulman et al., 2017) and GAE (Schulman et al., 2015). We document hyperparameters and compute usage in Appendix G.

For MEV, we use a standard method for training in UMDPs given a fixed level distribution.

1. **Domain randomization** (DR; Tobin et al., 2017)**.** For each iteration of PPO, we sample (procedurally generate) a new batch of levels from the fixed training level distribution $\Lambda_\alpha^{\text{Train}}$, collect experience in this batch of levels, and then train on the collected experience.

For MMER, we use two methods of regret-based unsupervised environment design (UED; Dennis et al., 2020). UED methods implement the two-level optimization from Definition 5 by training the policy on levels selected from a distribution chosen by a regret-maximizing **adversary**. The first UED method is a regret-based form of prioritized level replay (PLR; Jiang et al., 2021b).

2. **Robust prioritized level replay** (PLR$^\perp$; Jiang et al., 2021a)**.** The adversary parametrizes its level distribution using a fixed-size **level buffer.** Throughout training, the adversary refines the buffer by either (1) sampling a new batch of levels from the underlying training distribution $\Lambda_\alpha^{\text{Train}}$ and estimating the expected regret of the current policy on these levels; or (2) sampling from the current buffer, conducting a PPO training step with the chosen levels, and updating their expected regret estimates; keeping the highest-regret levels in the buffer.

PLR$^\perp$ has the advantage of being domain-agnostic, but has the disadvantage of only being able to *replay* levels once they have been sampled from the underlying distribution. We also consider a more advanced adversary with an independent means of exploring the space of level distributions.

3. **Adversarially compounding complexity by editing levels** (ACCEL; Parker-Holder et al., 2022)**.** The adversary continually refines a level buffer with steps (1) and (2) from PLR$^\perp$, and additionally by (3) applying stochastic **edits** to the levels used for PPO training to generate similar levels, and estimating the expected regret of the current policy on these new levels.

ACCEL additionally requires an **edit distribution.** We edit levels by sampling a sequence of random elementary level modifications, none of which change whether the level is non-distinguishing or distinguishing. Appendix J details this edit distribution and compares it to edit distributions with more or less ability to introduce distinguishing levels.

6.3 Expected regret estimation methods

Both UED methods require an **(expected) regret estimator** for deciding which levels to keep in the buffer. To represent the current capabilities of UED methods, we use the following domain-agnostic estimator, similar to the MaxMC estimator proposed by Jiang et al. (2021a).

1. **Max-latest estimator.** We estimate the expected regret of policy π in level θ under goal R as

$$\hat{G}^R_{\text{max-latest}}(\pi;\theta) = \hat{V}^R_{\text{max}}(\theta) - \hat{V}^R_{\text{latest}}(\pi;\theta) \tag{3}$$

where $\hat{V}^R_{\text{max}}(\theta)$ is the highest empirical return ever achieved for this level throughout training; and $\hat{V}^R_{\text{latest}}(\pi;\theta)$ is the empirical average return achieved by the current policy.

To simulate a more advanced regret estimator than is currently available in practice, we also consider a domain-specific estimator that solves each procedurally-generated level using a graph algorithm to compute the exact maximum expected return (details in Appendix F).

2. **Oracle-latest estimator.** We estimate the expected regret of policy π in level θ under goal R as

$$\hat{G}^R_{\text{oracle-latest}}(\pi;\theta) = \max_{\pi'} V^R(\pi';\theta) - \hat{V}^R_{\text{latest}}(\pi;\theta) \tag{4}$$

where $\max_{\pi'} V^R(\pi';\theta)$ is the maximum expected return for the level; and $\hat{V}^R_{\text{latest}}(\pi;\theta)$ is as above.

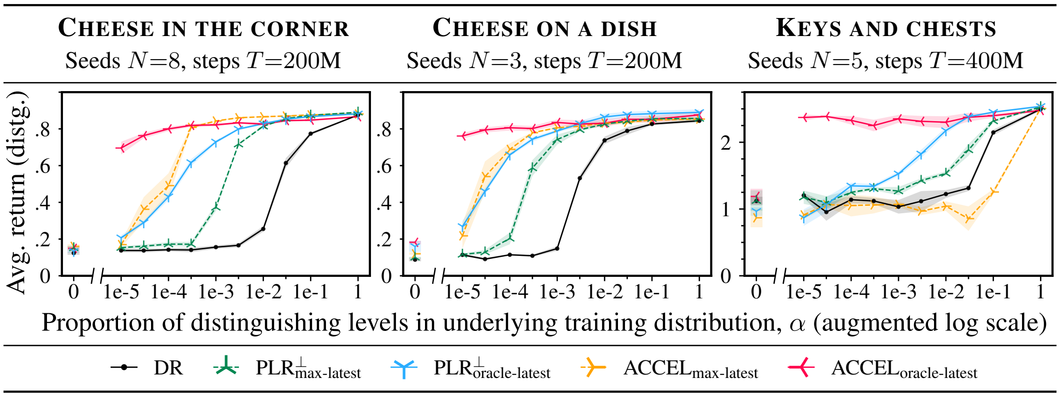

Figure 4: **Distribution shift performance for various training distributions.** Average return over 512 steps for an evaluation batch of 256 distinguishing levels sampled from $\Lambda^{\text{Deploy}} = \Lambda_{\text{Distg.}}$. High performance indicates policies generalizing as intended; low performance indicates goal misgeneralization. Each policy is trained on T environment steps using the indicated training method with underlying training distribution $\Lambda^{\text{Train}}_{\alpha} = (1-\alpha)\Lambda_{\neg\text{Distg.}} + \alpha\Lambda_{\text{Distg.}}$. Mean over N seeds, shaded to one standard error. Note the split in the horizontal axis used to show zero on the log scale.

7 Experimental results

In this section, we report the results of our main experiments. Consistent with Theorem 1, MEV-based training is susceptible to goal misgeneralization unless the proportion of distinguishing levels in the training distribution is sufficiently high (Section 7.1). Consistent with Theorem 2, MMER-based training methods are typically capable of identifying and increasing the proportion of rare, high-regret, distinguishing levels, thereby preventing goal misgeneralization in many situations where MEV-based training misgeneralizes (Section 7.2). In some cases, UED methods fail to find MMER policies, and exhibit goal misgeneralization. We see generally that the more advanced UED methods are more robust to goal misgeneralization (Section 7.3). In KEYS AND CHESTS, DR outperforms ACCEL with max-latest regret estimation, underscoring reliable regret estimation as a particular challenge for future work on MMER-based training (Section 7.4).

7.1 Domain randomization exhibits goal misgeneralization with rare distinguishing levels

Theorem 1 says that if the proportion of distinguishing levels in the fixed training distribution is small enough, then approximately optimizing MEV *possibly* leads to goal misgeneralization. Our experiments show that DR, an MEV-based training method, indeed exhibits goal misgeneralization when the proportion of distinguishing levels in the training distribution is small enough. Figure 4 shows end-of-training performance on distinguishing levels. There is a threshold below which DR's performance on distinguishing levels falls. DR achieves high return on non-distinguishing levels and high proxy return on distinguishing levels (Appendix H), indicating a case of goal misgeneralization.

In CHEESE IN THE CORNER and KEYS AND CHESTS, DR exhibits goal misgeneralization until there is around α = 1e-1 (10%) mass on distinguishing levels. For CHEESE ON A DISH, DR is robust to goal misgeneralization from as low as α = 1e-2 (1%) (see also Appendix N). Appendix L shows that training for substantially longer slightly increases DR's robustness in CHEESE ON A DISH.

We note that Langosco et al. (2022) previously demonstrated goal misgeneralization while training with DR without distinguishing levels in similar environments. Moreover, Langosco et al. (2022) demonstrated that for a modified version of OpenAI ProcGen's COINRUN environment (Cobbe et al., 2019; 2020), training with α = 2e-2 (2%) prevents goal misgeneralization. They did not experiment with smaller proportions of distinguishing levels. We show that with small but nonzero proportions of distinguishing levels, DR can still exhibit goal misgeneralization.

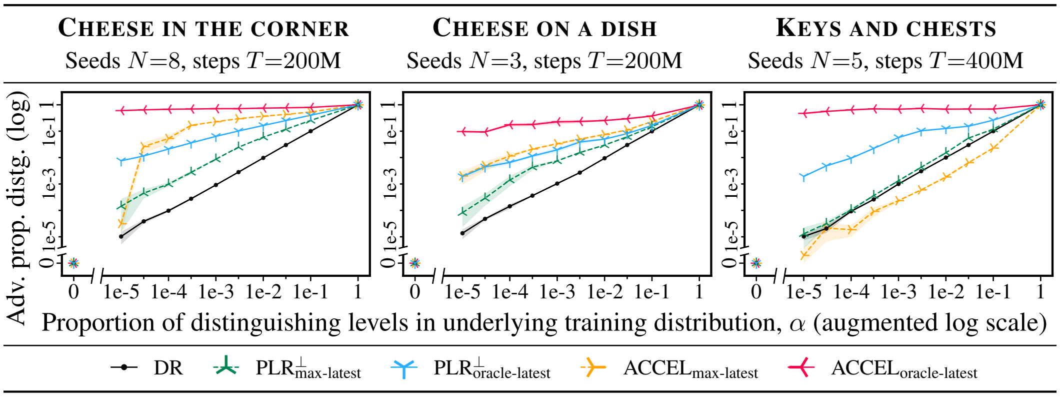

Figure 5: **Rate at which adversary plays distinguishing levels.** We plot the proportion of adversarially sampled levels classified as distinguishing across training for T environment steps. The diagonal represents the proportion from the underlying training distribution $\Lambda^{\text{Train}}_{\alpha} = (1-\alpha)\Lambda_{\neg\text{Distg.}} + \alpha\Lambda_{\text{Distg.}}$ (as used in DR). Points above the diagonal indicate the adversary increasing the proportion of distinguishing levels relative to the underlying training distribution. Mean over N seeds, shaded to one standard error. Note the splits in *both* axes used to show zero on the log scales.

7.2 Regret-based prioritization amplifies distinguishing levels, mitigating misgeneralization

Theorem 2 says that, if a policy pursues the proxy goal at the expense of the true goal in distinguishing levels, then the adversary should select a distribution of distinguishing levels that generates high regret. Figure 5 shows the average proportion of distinguishing levels selected from the adversary throughout training, showing that, with the exception of max-latest estimation in the KEYS AND CHESTS environment, the adversary selects distinguishing levels disproportionately often compared to sampling from the underlying distribution, thereby incentivizing policies that pursue the true goal.

Figure 4 shows that this increase in the proportion of training levels is, in most cases, enough to lead to correct generalization. In each environment, MMER-based training methods are robust to goal misgeneralization at α values for which DR exhibits goal misgeneralization. For example, in CHEESE IN THE CORNER, all UED methods are robust to goal misgeneralization at α = 1e-2 (1%), and some remain robust for even lower α. Note that some evaluation levels are unsolvable—the highest return to be expected is given by the agents trained with $\alpha = 1$.

7.3 Increasingly advanced UED methods are more robust to goal misgeneralization

Theorem 2 says that MMER-based training should be robust to goal misgeneralization regardless of the distribution shift. In contrast, in our experiments, the proportion of distinguishing levels played by the adversary decreases as we decrease α (Figure 5), and each UED method exhibits a threshold below which it fails to converge to an MMER policy, and exhibits goal misgeneralization (Figure 4).

This performance trend reflects how the adversaries construct level distributions. When distinguishing levels are very rare (or never arise), the adversary is hindered (prevented) from increasing the number of distinguishing levels in the buffer. Compared to PLR$^{\perp}$, ACCEL can replicate similar levels throughout its buffer through edits, but we used edits that don't create new distinguishing levels. Appendix J investigates ACCEL variants with different edit distributions, showing that ACCEL can prevent goal misgeneralization even when $\alpha = 0$ if edits can introduce distinguishing levels.

Overall, robustness correlates with how advanced the adversaries are. Our most flexible adversary (ACCEL) paired with our most powerful regret estimator (oracle-latest) is remarkably robust to goal misgeneralization in all environments for all positive α tested. The less flexible PLR$^{\perp}$ using the less powerful max-latest expected regret estimator is the least robust. This motivates work pursuing regret-based UED methods with better convergence properties (cf. Monette et al., 2025).

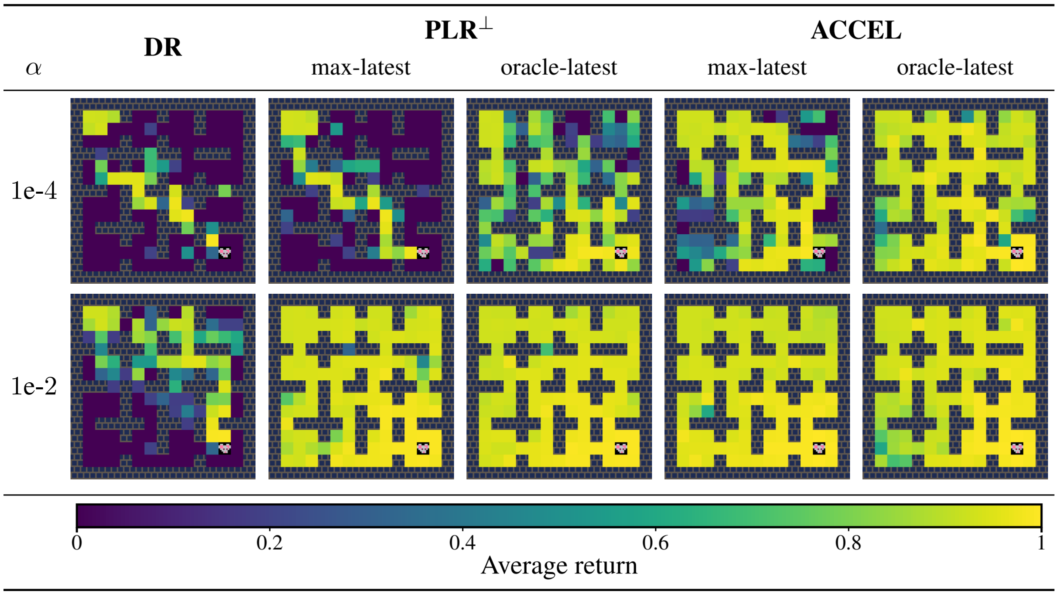

Figure 6: **Performance on CHEESE IN THE CORNER levels with varying cheese position.** For each training configuration, we evaluate the trained policy (first of 8 seeds) on a batch of 122 levels with shared wall layout and mouse spawn position but different cheese positions. We indicate average return on levels with different cheese positions by the color of the corresponding grid square. We see a progression whereby for more advanced algorithms or higher α, the agent is robust to a greater proportion of cheese positions. See Appendix I for more details and full range of α values.

7.4 Biased regret estimation can undermine UED in more complex environments

The poor performance of ACCEL with max-latest the estimator in the KEYS AND CHESTS environment underscores the challenge of expected regret estimation. Estimating maximum return from samples is particularly challenging in this environment, where high return is unlikely to be achieved in distinguishing levels by chance, since chests are substantially rarer than in non-distinguishing levels. It appears that the increased flexibility of ACCEL in this case works as a disadvantage, leading to the adversary being led astray by biased regret estimates even more so than PLR$^{\perp}$.

The challenges of regret estimation are known, and are an active area of research (cf. Rutherford et al., 2024). Our results highlight the importance of future work on reliable regret estimation methods, towards achieving the improved performance shown by our domain-specific oracle-latest estimator. Such work could investigate using a separate policy network to estimate the maximum return (cf. Dennis et al., 2020), or incorporating the predictions of a value network (cf. Jiang et al., 2021a).

8 Conclusion

In this paper, we introduce the setting of a proxy-distinguishing distribution shift, and offer a theoretical and empirical investigation of the robustness of MEV-based and MMER-based training to goal misgeneralization. We show theoretically and empirically that MEV-based training on a fixed training distribution can lead to goal misgeneralization. In contrast, we show that MMER-based training is provably robust against goal misgeneralization in the limit of idealized training methods, and regret-based unsupervised environment design (UED) methods are empirically more robust than MEV-based training. Current UED methods do not find MMER policies and prevent goal misgeneralization in all the cases we studied, indicating there is still room for improvement between current methods and the theoretical ideal. These findings highlight MMER-based training as a promising approach to preventing goal misgeneralization.

A Proofs for theoretical results from Section 5

Proof of Theorem 1. Assume $\alpha \leq \varepsilon$. Construct a policy π^{MEV} such that, for all levels $\theta \in \Theta$,

$$\begin{array}{ll} \pi^{\text{MEV}} \in \operatorname*{arg\,max}_{\pi \in \Pi} V^{\tilde{R}}(\pi; \theta) \setminus \operatorname*{arg\text{-}\mathit{C}\text{-}max}_{\pi \in \Pi} V^{R}(\pi; \theta) & \text{if } \theta \text{ is } C\text{-distinguishing, or} \\ \pi^{\text{MEV}} \in \operatorname*{arg\,max}_{\pi \in \Pi} V^{\tilde{R}}(\pi; \theta) & \text{otherwise.} \end{array}$$

The former is non-empty by Definition 2. By construction, $\pi^{\text{MEV}} \in \operatorname{arg\,max}_{\pi \in \Pi} V^{\tilde{R}}(\pi; \Lambda^{\text{Deploy}})$. It remains to prove (i) $\pi^{\text{MEV}} \in \Pi^{\text{MEV}}_{\varepsilon}(R, \Lambda^{\text{Train}})$ and (ii) $\pi^{\text{MEV}} \notin \text{arg-}\beta C\text{-max}_{\pi \in \Pi} V^{R}(\pi; \Lambda^{\text{Deploy}})$.

For notational convenience, construct a policy $\pi^{\star} \in \Pi$ that is optimal under R in all levels. Moreover, let $\Lambda^{\text{Train}}_{\text{Distg.}}, \Lambda^{\text{Train}}_{\neg\text{Distg.}}, \Lambda^{\text{Deploy}}_{\text{Distg.}}, \Lambda^{\text{Deploy}}_{\neg\text{Distg.}} \in \Delta(\Theta)$ be Λ^{Train} and Λ^{Deploy} conditioned on the level being C-distinguishing or non-distinguishing, respectively.

Then for condition (i), we have

$$\begin{aligned} V^{R}(\pi^{\text{MEV}}; \Lambda^{\text{Train}}) &= \alpha V^{R}(\pi^{\text{MEV}}; \Lambda^{\text{Train}}_{\text{Distg.}}) + (1-\alpha) V^{R}(\pi^{\text{MEV}}; \Lambda^{\text{Train}}_{\neg\text{Distg.}}) && \text{(by Definition 3)} \\ &= \alpha V^{R}(\pi^{\text{MEV}}; \Lambda^{\text{Train}}_{\text{Distg.}}) + (1-\alpha) V^{R}(\pi^{\star}; \Lambda^{\text{Train}}_{\neg\text{Distg.}}) && \text{(by Definition 1)} \\ &\geq \alpha \cdot 0 + (1-\alpha) V^{R}(\pi^{\star}; \Lambda^{\text{Train}}_{\neg\text{Distg.}}) && (\text{since } V^{R} \geq 0) \\ &= V^{R}(\pi^{\star}; \Lambda^{\text{Train}}) - \alpha V^{R}(\pi^{\star}; \Lambda^{\text{Train}}_{\text{Distg.}}) && \text{(by Definition 3)} \\ &\geq V^{R}(\pi^{\star}; \Lambda^{\text{Train}}) - \varepsilon \cdot 1. && (\text{since } \alpha \leq \varepsilon;\ V^{R} \leq 1) \end{aligned}$$

For condition (ii), we have

$$\begin{aligned} V^{R}(\pi^{\text{MEV}}; \Lambda^{\text{Deploy}}) &= \beta V^{R}(\pi^{\text{MEV}}; \Lambda^{\text{Deploy}}_{\text{Distg.}}) + (1-\beta) V^{R}(\pi^{\text{MEV}}; \Lambda^{\text{Deploy}}_{\neg\text{Distg.}}) && \text{(by Definition 3)} \\ &= \beta V^{R}(\pi^{\text{MEV}}; \Lambda^{\text{Deploy}}_{\text{Distg.}}) + (1-\beta) V^{R}(\pi^{\star}; \Lambda^{\text{Deploy}}_{\neg\text{Distg.}}) && \text{(by Definition 1)} \\ &< \beta\big(V^{R}(\pi^{\star}; \Lambda^{\text{Deploy}}_{\text{Distg.}}) - C\big) + (1-\beta) V^{R}(\pi^{\star}; \Lambda^{\text{Deploy}}_{\neg\text{Distg.}}) \\ & && \Big(\pi^{\text{MEV}} \notin \operatorname*{arg\text{-}\mathit{C}\text{-}max}_{\pi \in \Pi} V^{R}(\pi; \theta)\Big) \\ &= V^{R}(\pi^{\star}; \Lambda^{\text{Deploy}}) - \beta C. && \text{(by Definition 3)} \quad \square \end{aligned}$$

Proof of Theorem 2. Suppose $\pi^{\text{MMER}} \in \Pi^{\text{MMER}}_{\varepsilon}(R)$. Then we have the following bound on expected regret:

$$\begin{aligned} G^{R}(\pi^{\text{MMER}}; \Lambda^{\text{Deploy}}) &\leq \max_{\Lambda \in \Delta(\Theta)} G^{R}(\pi^{\text{MMER}}; \Lambda) && (\Lambda^{\text{Deploy}} \in \Delta(\Theta)) \\ &\leq \min_{\pi \in \Pi} \max_{\Lambda \in \Delta(\Theta)} G^{R}(\pi; \Lambda) + \varepsilon. && \text{(by Definition 5)} \end{aligned}$$

We can convert this upper bound on expected regret to a lower bound on expected return:

$$\begin{aligned} V^{R}(\pi^{\text{MMER}}; \Lambda^{\text{Deploy}}) &= \max_{\pi \in \Pi} V^{R}(\pi; \Lambda^{\text{Deploy}}) - G^{R}(\pi^{\text{MMER}}; \Lambda^{\text{Deploy}}) && \text{(by equation 2)} \\ &\geq \max_{\pi \in \Pi} V^{R}(\pi; \Lambda^{\text{Deploy}}) - \min_{\pi \in \Pi} \max_{\Lambda \in \Delta(\Theta)} G^{R}(\pi; \Lambda) - \varepsilon. && \text{(by above bound)} \end{aligned}$$

The theorem follows, since, for all $\Lambda \in \Delta(\Theta)$, $\min_{\pi \in \Pi} G^{R}(\pi; \Lambda)$ vanishes by equation (2):

$$\begin{aligned} \min_{\pi \in \Pi} G^{R}(\pi; \Lambda) &= \min_{\pi \in \Pi} \left(\max_{\pi' \in \Pi} V^{R}(\pi'; \Lambda) - V^{R}(\pi; \Lambda) \right) && \text{(by equation 2)} \\ &= \max_{\pi' \in \Pi} V^{R}(\pi'; \Lambda) - \max_{\pi \in \Pi} V^{R}(\pi; \Lambda) && (\text{max term is constant wrt. } \pi) \\ &= 0. && \square \end{aligned}$$

Broader impact statement

Ngo et al. (2024) cast goal misgeneralization as a key risk mechanism for advanced deep learning systems, noting that techniques that improve capability robustness without preventing goal misgeneralization could *worsen* outcomes, since the system's greater capabilities would then be devoted to the pursuit of an incorrect goal. Preventing this dangerous mode of generalization failure is a key challenge in assuring the safety of advanced RL agents.

In this section, we briefly note that minimax expected regret appears to be well-suited in principle to mitigating goal misgeneralization as deep learning systems become increasingly capable. This is because more generally capable deep learning systems should also be more capable regret-maximizing adversaries in particular. A more capable adversary will, in turn, be better at detecting or synthesizing rare, high-regret training situations, and then amplifying the training signal from these situations so as to induce correct generalization in an advanced deep RL agent (cf., Appendix J).

Our work highlights training with the minimax expected regret (MMER) objective as a promising avenue for preventing goal misgeneralization. This objective has desirable theoretical properties, and we have found promising initial empirical results, though current MMER-based training techniques are not mature enough to prevent goal misgeneralization in all cases. As MMER-based training methods improve and as goal misgeneralization leads to more severe consequences, the ability of MMER-based training to mitigate goal misgeneralization should also improve.

Ultimately, we are hopeful that our work will instigate further research on the problem of goal misgeneralization, which remains a critical, open problem in the alignment and safe generalization of future advanced reinforcement learning agents.

Acknowledgments

We thank Jason Brown, Robert Kirk, and Lauro Langosco for helpful discussions. We thank Stephen Chung and Samuel Coward for advice with training algorithms. We thank Micah Carroll, Dmitrii Krasheninnikov, Niklas Lauffer, Michelle Li, Clare Lyle, Benjamin Plaut, Rohin Shah, and our anonymous reviewers (especially Reviewer DDwf) for helpful feedback on the research and the manuscript. KAS was supported in part by the Cambridge ERA:AI Fellowship.

References

Matthew Barnett. A simple environment for showing mesa misalignment. Alignment Forum, September 2019. URL https://www.alignmentforum.org/posts/AFdRGfYDWQqmkdhFq.

Michael Beukman, Samuel Coward, Michael Matthews, Mattie Fellows, Minqi Jiang, Michael Dennis, and Jakob N. Foerster. Refining minimax regret for unsupervised environment design. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 3637–3657. PMLR, 2024.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C. Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando De Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623. PMLR, 2024.

Paul Christiano. Techniques for optimizing worst-case performance. AI Alignment (Blog), February 2018. URL https://ai-alignment.com/techniques-for-optimizing-worst-case-performance-39eafec74b99.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1282–1289. PMLR, 2019.

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.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart J. Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In *Advances in Neural Information Processing Systems 33*, pp. 13049–13061. Curran Associates, Inc., 2020.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad 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. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1407–1416. PMLR, 2018.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2:665–673, 2020.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, pp. 2450–2462. Curran Associates, Inc., 2018.

Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J. Russell, and Anca Dragan. Inverse reward design. In *Advances in Neural Information Processing Systems 30*, pp. 6765–6774. Curran Associates, Inc., 2017.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *8th International Conference on Learning Representations*. OpenReview, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.

Evan Hubinger. Towards an empirical investigation of inner alignment. Alignment Forum, September 2019. URL https://www.alignmentforum.org/posts/2GycxikGnepJbxfHT.

Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob N. Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. In *Advances in Neural Information Processing Systems 34*, pp. 1884–1897. Curran Associates, Inc., 2021a.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4940–4950. PMLR, 2021b.

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. In *Proceedings of Robotics: Science and Systems XVII*, 2021.

Lauro 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, 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 *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6781–6792. PMLR, 2021.

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. In *Proceedings of Robotics: Science and Systems XX*, 2024.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance GPU based physics simulation for robot learning. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Nathan Monette, Alistair Letcher, Michael Beukman, Matthew Thomas Jackson, Alexander Rutherford, Alexander David Goldie, and Jakob N. Foerster. An optimisation framework for unsupervised environment design. *Reinforcement Learning Journal*, 2025. To appear.

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.

Richard Ngo, Lawrence Chan, and Sören Mindermann. The alignment problem from a deep learning perspective. In *12th International Conference on Learning Representations*. OpenReview, 2024.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik's Cube with a robot hand. Preprint arXiv:1910.07113 [cs.LG], 2019.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob N. Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. Preprint arXiv:2203.01302 [cs.LG], 2022.

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*, pp. 3803–3810. IEEE, 2018.

Alexander Rutherford, Michael Beukman, Timon Willi, Bruno Lacerda, Nick Hawes, and Jakob Foerster. No regrets: Investigating and improving regret approximations for curriculum discovery. In *Advances in Neural Information Processing Systems 37*, pp. 16071–16101. Curran Associates, Inc., 2024.

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

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. 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. Preprint arXiv:1506.02438 [cs.LG], 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. Preprint arXiv:1707.06347 [cs.LG], 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. Preprint arXiv:2210.01790 [cs.LG], 2022.

Giulio Starace. Addressing goal misgeneralization with natural language interfaces. Master's thesis, University of Amsterdam, 2023.

Miguel Suau, Matthijs T. J. Spaan, and Frans A. Oliehoek. Bad habits: Policy confounding and out-of-trajectory generalization in reinforcement learning. *Reinforcement Learning Journal*, 4: 1711–1732, 2024.

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*, 2017.

Tu Trinh, Mohamad H. Danesh, Nguyen X. Khanh, and Benjamin Plaut. Getting by goal misgeneralization with a little help from a mentor. In *The First Workshop on Safe & Trustworthy Agents*, 2024. Workshop at NeurIPS 2024. Preprint arXiv:2410.21052 [cs.LG].

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *13th International Conference on Learning Representations*. OpenReview, 2025.

Shengbo Wang, Nian Si, Jose Blanchet, and Zhengyuan Zhou. On the foundation of distributionally robust reinforcement learning. Preprint arxiv:2311.09018 [cs.LG], 2023.

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. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 26484–26516. PMLR, 2022.

Simon Zhuang and Dylan Hadfield-Menell. Consequences of misaligned AI. In *Advances in Neural Information Processing Systems 33*, pp. 15763–15773. Curran Associates, Inc., 2020.

Supplementary Materials

The following content was not necessarily subject to peer review.

Contents

B Expected regret identity for UMDPs

In this section, we prove equation (2) for UMDPs. Recall the following definitions from Section 3.

$$V^R(\pi;\theta) = \mathbb{E}_{s_0\sim\mathcal{I}(\theta),a_t\sim\pi(\theta,s_t),s_{t+1}\sim\mathcal{T}(\theta,s_t,a_t)}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t,s_{t+1})\right] \tag{5}$$

$$V^R(\pi;\Lambda) = \mathbb{E}_{\theta\sim\Lambda}\left[\,V^R(\pi;\theta)\,\right] \tag{6}$$

$$G^R(\pi;\theta) = \max_{\pi'\in\Pi} V^R(\pi';\theta) - V^R(\pi;\theta) \tag{7}$$

$$G^R(\pi;\Lambda) = \mathbb{E}_{\theta\sim\Lambda}\left[\,G^R(\pi;\theta)\,\right] \tag{8}$$

In Section 3, we observe that, for UMDPs, we have the additional basic identity

$$G^R(\pi;\Lambda) = \max_{\pi'\in\Pi} V^R(\pi';\Lambda) - V^R(\pi;\Lambda). \qquad \text{(this is equation 2)}$$

This is a nontrivial identity that does not hold for partially observable underspecified environments in which the level is not observable to the policy (see Appendix E.1). However, for policies that are conditioned on the level, the identity holds, as we now prove.

Proposition 1 (Expected regret identity for UMDPs)**.** *Consider an UMDP $\langle\Theta,\mathcal{A},\mathcal{S},\mathcal{I},\mathcal{T}\rangle$, a goal R, and a level distribution $\Lambda\in\Delta(\Theta)$. Let Π be the set of all policies of the form $\pi:\Theta\times\mathcal{S}\to\Delta(\mathcal{A})$. Then we have*

$$G^R(\pi;\Lambda) = \max_{\pi'\in\Pi} V^R(\pi';\Lambda) - V^R(\pi;\Lambda).$$

Proof.

$$\begin{aligned}
G^R(\pi;\Lambda) &= \mathbb{E}_{\theta\sim\Lambda}\left[\,G^R(\pi;\theta)\,\right] && \text{(equation 8)}\\
&= \mathbb{E}_{\theta\sim\Lambda}\left[\max_{\pi'\in\Pi} V^R(\pi';\theta) - V^R(\pi;\theta)\right] && \text{(by equation 7)}\\
&= \max_{\pi'\in\Pi} \mathbb{E}_{\theta\sim\Lambda}\left[\,V^R(\pi';\theta)\,\right] - \mathbb{E}_{\theta\sim\Lambda}\left[\,V^R(\pi;\theta)\,\right] && \text{(by Proposition 2, below)}\\
&= \max_{\pi'\in\Pi} V^R(\pi';\Lambda) - V^R(\pi;\Lambda). && \text{(by equation 6)} \quad \square
\end{aligned}$$

The above proof relies on Proposition 2, which says that we can exchange expectation and maximization for the expected return since the policy is conditioned on the level.

Proposition 2 (Expectation and maximization of expected return commute for UMDPs)**.** *Consider an UMDP $\langle\Theta,\mathcal{A},\mathcal{S},\mathcal{I},\mathcal{T}\rangle$, a goal R, and a level distribution $\Lambda\in\Delta(\Theta)$. Let Π be the set of all policies of the form $\pi:\Theta\times\mathcal{S}\to\Delta(\mathcal{A})$. Then we have*

$$\mathbb{E}_{\theta\sim\Lambda}\left[\max_{\pi\in\Pi} V^R(\pi;\theta)\right] = \max_{\pi\in\Pi}\mathbb{E}_{\theta\sim\Lambda}\left[\,V^R(\pi;\theta)\,\right].$$

Proof. *($\geq$):* Note that this direction holds regardless of whether we condition policies on the level. Let $\pi^\star \in \arg\max_{\pi\in\Pi}\mathbb{E}_{\theta\sim\Lambda}\left[\,V^R(\pi;\theta)\,\right]$. Then we have

$$\max_{\pi\in\Pi}\mathbb{E}_{\theta\sim\Lambda}\left[\,V^R(\pi;\theta)\,\right] = \mathbb{E}_{\theta\sim\Lambda}\left[\,V^R(\pi^\star;\theta)\,\right] \leq \mathbb{E}_{\theta\sim\Lambda}\left[\max_{\pi\in\Pi} V^R(\pi;\theta)\right].$$

($\leq$): Observe that, per equation (5), $V^R(\pi;\theta)$ depends only on π through $\pi(\theta,-):\mathcal{S}\to\Delta(\mathcal{A})$, that is, through the policy conditioned on the fixed level θ. Therefore, we can construct a single policy that achieves the maximum expected return under all levels. For $\theta\in\Theta$, let $\pi^\star_\theta\in\arg\max_{\pi\in\Pi}V^R(\pi;\theta)$. Then define $\pi^\star:\Theta\times\mathcal{S}\to\Delta(\mathcal{A})$ such that for $\theta\in\Theta$, $s\in\mathcal{S}$, and $a\in\mathcal{A}$, $\pi^\star(a\mid\theta,s)=\pi^\star_\theta(a\mid\theta,s)$. By construction, we have $\pi^\star_\theta\in\arg\max_{\pi\in\Pi}V^R(\pi;\theta)$ for all $\theta\in\Theta$. It follows that

$$\mathbb{E}_{\theta\sim\Lambda}\left[\max_{\pi\in\Pi} V^R(\pi;\theta)\right] = \mathbb{E}_{\theta\sim\Lambda}\left[\,V^R(\pi^\star;\theta)\,\right] \leq \max_{\pi\in\Pi}\mathbb{E}_{\theta\sim\Lambda}\left[\,V^R(\pi;\theta)\,\right]. \qquad \square$$

C Approximate relaxations of the minimax expected regret decision rule

The minimax expected regret decision rule says to choose a policy that *minimizes* the expected regret with respect to the *maximum* expected regret level distribution for the given policy. In Section 5.2, we consider one possible approximate relaxation of this decision rule, where we replace only the minimization step with approximate minimization (but retain the exact maximization step).

In this appendix, we formulate two alternative approximate relaxations of the minimax expected regret decision rule that also relax the maximization step (Appendix C.1). We also show that these three formulations are asymptotically equivalent (Appendix C.2), and we derive robustness guarantees akin to Theorem 2 corresponding to the two new definitions (Appendix C.3).

C.1 Alternative definitions of approximate minimax expected regret

First, we restate the approximate MMER definition from Section 5.2. The only difference is that we add a qualifier "(1)" in preparation for distinguishing this definition from the two alternatives to follow, and suppress the dependence on R in the notation for brevity.

Definition 6 (Approximate MMER(1), restating Definition 5)**.** *Consider an UMDP $\langle \Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$, a goal R, and an approximation threshold $\varepsilon \geq 0$. The* approximate MMER(1) policy set *is then*

$$\Pi_{\varepsilon}^{\text{MMER(1)}} = \underset{\pi \in \Pi}{\text{arg-}\varepsilon\text{-min}} \max_{\Lambda \in \Delta(\Theta)} G^R(\pi; \Lambda).$$

This decision rule is approximate in that we don't assume we can find a policy that achieves the true *minimum* of the maximum expected regret. However, we still assume we can find the true *maximum* expected regret for each policy. We consider next two approaches for relaxing this assumption.

The first approach casts the MMER objective as finding a Nash equilibrium of a two-player, simultaneous-play zero-sum game in which the first player is the agent selecting a policy and the second player is an adversary selecting a level distribution. We can therefore relax both the minimization and the maximization simultaneously by using the concept of an *approximate Nash equilibrium,* in which each player plays an *approximate* best response.

Definition 7 (Approximate MMER(2))**.** *Consider an UMDP $\langle \Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$, a goal R, and approximation thresholds $\varepsilon, \delta \geq 0$. Consider the two-player zero-sum game $\big\langle \langle \Pi, \Delta(\Theta) \rangle, \langle -G^R, G^R \rangle \big\rangle$, where an* agent *plays a policy $\pi \in \Pi$ and an* adversary *plays a level distribution $\Lambda \in \Delta(\Theta)$, aiming to minimize or maximize $G^R(\pi; \Lambda)$ respectively. A pair (π, Λ) is an* (ε, δ)-equilibrium *if*

$$\pi \in \underset{\pi' \in \Pi}{\text{arg-}\varepsilon\text{-min}}\, G^R(\pi'; \Lambda) \qquad \textit{and} \qquad \Lambda \in \underset{\Lambda' \in \Delta(\Theta)}{\text{arg-}\delta\text{-max}}\, G^R(\pi; \Lambda').$$

The approximate MMER(2) policy set *is then*

$$\Pi_{\varepsilon,\delta}^{\text{MMER(2)}} = \big\{ \pi \in \Pi \;\big|\; \exists \Lambda \in \Delta(\Theta) \textit{ such that } (\pi, \Lambda) \textit{ is an } (\varepsilon, \delta)\textit{-equilibrium} \big\}.$$

Our second approach conditions on a concrete mapping capturing an approximately optimal response from the adversary to each possible policy, with respect to which we require the chosen MMER policy to be approximately optimal. This definition could alternatively be formulated in terms of a *sequential* zero-sum game where the agent chooses the policy and reveals it to the adversary prior to the adversary choosing a level distribution aiming to maximize expected regret.

Definition 8 (Approximate MMER(3))**.** *Consider an UMDP $\langle \Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$, a goal R, and approximation thresholds $\varepsilon, \eta \geq 0$. Let $\lambda : \Pi \to \Delta(\Theta)$ be a function such that for all $\pi \in \Pi$,*

$$\lambda(\pi) \in \underset{\Lambda \in \Delta(\Theta)}{\text{arg-}\eta\text{-max}}\, G^R(\pi; \Lambda).$$

*Call a function with this property an η-*approximate adversarial map*. The* approximate MMER(3) policy set *with respect to λ is then*

$$\Pi_{\varepsilon,\lambda}^{\text{MMER(3)}} = \underset{\pi \in \Pi}{\text{arg-}\varepsilon\text{-min}}\, G^R(\pi; \lambda(\pi)).$$

C.2 Asymptotic equivalence of the definitions

Definitions 6, 7, and 8 do not necessarily define equal sets of policies. However, the three sets of policies are closely related. Proposition 3, below, shows that each set is contained in the others for appropriately-chosen values of the approximation thresholds $\varepsilon, \delta, \eta \geq 0$.

Proposition 3 (Asymptotic equivalence of approximate MMER definitions)**.** *Consider an UMDP $\langle \Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$, a goal R, approximation thresholds $\varepsilon, \delta, \eta \geq 0$, and an η-approximate adversarial map λ. We have the following relations:*

$$\begin{aligned}
\Pi^{\text{MMER(1)}}_{\varepsilon} &\subseteq \Pi^{\text{MMER(2)}}_{\varepsilon,\varepsilon} & \qquad \Pi^{\text{MMER(2)}}_{\varepsilon,\delta} &\subseteq \Pi^{\text{MMER(1)}}_{\varepsilon+\delta} \\
\Pi^{\text{MMER(1)}}_{\varepsilon} &\subseteq \Pi^{\text{MMER(3)}}_{\varepsilon+\eta,\lambda} & \qquad \Pi^{\text{MMER(3)}}_{\varepsilon,\lambda} &\subseteq \Pi^{\text{MMER(1)}}_{\varepsilon+\eta} \\
\Pi^{\text{MMER(2)}}_{\varepsilon,\delta} &\subseteq \Pi^{\text{MMER(3)}}_{\varepsilon+\delta+\eta,\lambda} & \qquad \Pi^{\text{MMER(3)}}_{\varepsilon,\lambda} &\subseteq \Pi^{\text{MMER(2)}}_{\varepsilon+\eta,\varepsilon+\eta}.
\end{aligned}$$

Proof. We prove the top four subset relationships. The remaining two relationships follow. Before proceeding, we note that since we assume $\Delta(\Theta)$, $\mathcal{S}$, and $\mathcal{A}$ are finite and our environments are fully observable, we have

$$\min_{\pi \in \Pi} \max_{\Lambda \in \Delta(\Theta)} G^R(\pi; \Lambda) = \max_{\Lambda \in \Delta(\Theta)} \min_{\pi \in \Pi} G^R(\pi; \Lambda). \tag{9}$$

In the general case, modified bounds can be derived by accounting for the value of the difference $\min_{\pi \in \Pi} \max_{\Lambda \in \Delta(\Theta)} G^R(\pi; \Lambda) - \max_{\Lambda \in \Delta(\Theta)} \min_{\pi \in \Pi} G^R(\pi; \Lambda)$ (see the remark after this proof).

($1 \subseteq 2$): Suppose $\pi^{(1)} \in \Pi^{\text{MMER(1)}}_{\varepsilon} = \operatorname{arg-}\varepsilon\text{-}\min_{\pi \in \Pi} \max_{\Lambda \in \Delta(\Theta)} G^R(\pi; \Lambda)$. Put $\Lambda^{(*)} \in \arg\max_{\Lambda \in \Delta(\Theta)} \min_{\pi \in \Pi} G^R(\pi; \Lambda)$. Then, we have the following cycle of relations.

$$\begin{aligned}
G^R(\pi^{(1)}; \Lambda^{(*)}) &\leq \max_{\Lambda \in \Delta(\Theta)} G^R(\pi^{(1)}; \Lambda) && \text{(by definition of max)} \\
&\leq \min_{\pi \in \Pi} \max_{\Lambda \in \Delta(\Theta)} G^R(\pi; \Lambda) + \varepsilon && (\pi^{(1)} \in \operatorname{arg-}\varepsilon\text{-}\min_{\pi \in \Pi} \max_{\Lambda \in \Delta(\Theta)} G^R(\pi; \Lambda)) \\
&= \max_{\Lambda \in \Delta(\Theta)} \min_{\pi \in \Pi} G^R(\pi; \Lambda) + \varepsilon && \text{(by equation 9)} \\
&= \min_{\pi \in \Pi} G^R(\pi; \Lambda^{(*)}) + \varepsilon && (\Lambda^{(*)} \in \arg\max_{\Lambda \in \Delta(\Theta)} \min_{\pi \in \Pi} G^R(\pi; \Lambda)) \\
&\leq G^R(\pi^{(1)}; \Lambda^{(*)}) + \varepsilon. && \text{(by definition of min)}
\end{aligned}$$

Comparing the first and second-last terms, we have that $\pi^{(1)} \in \operatorname{arg-}\varepsilon\text{-}\min_{\pi \in \Pi} G^R(\pi; \Lambda^{(*)})$. Comparing the second and the last terms, we have $\Lambda^{(*)} \in \operatorname{arg-}\varepsilon\text{-}\max_{\Lambda \in \Delta(\Theta)} G^R(\pi^{(1)}; \Lambda)$. It follows that $(\pi^{(1)}, \Lambda^{(*)})$ is an $(\varepsilon, \varepsilon)$-equilibrium, which means $\pi^{(1)} \in \Pi^{\text{MMER(2)}}_{\varepsilon,\varepsilon}$.

($2 \subseteq 1$): Suppose $\pi^{(2)} \in \Pi^{\text{MMER(2)}}_{\varepsilon,\delta}$. Then by definition there exists $\Lambda^{(2)} \in \Delta(\Theta)$ such that both $\pi^{(2)} \in \operatorname{arg-}\varepsilon\text{-}\min_{\pi \in \Pi} G^R(\pi; \Lambda^{(2)})$ and $\Lambda^{(2)} \in \operatorname{arg-}\delta\text{-}\max_{\Lambda \in \Delta(\Theta)} G^R(\pi^{(2)}; \Lambda)$. Then we have

$$\begin{aligned}
\max_{\Lambda \in \Delta(\Theta)} G^R(\pi^{(2)}; \Lambda) &\leq G^R(\pi^{(2)}; \Lambda^{(2)}) + \delta && (\Lambda^{(2)} \in \operatorname{arg-}\delta\text{-}\max_{\Lambda \in \Delta(\Theta)} G^R(\pi^{(2)}; \Lambda)) \\
&\leq \min_{\pi \in \Pi} G^R(\pi; \Lambda^{(2)}) + \varepsilon + \delta && (\pi^{(2)} \in \operatorname{arg-}\varepsilon\text{-}\min_{\pi \in \Pi} G^R(\pi; \Lambda^{(2)})) \\
&\leq \max_{\Lambda \in \Delta(\Theta)} \min_{\pi \in \Pi} G^R(\pi; \Lambda) + \varepsilon + \delta && \text{(by definition of max)} \\
&= \min_{\pi \in \Pi} \max_{\Lambda \in \Delta(\Theta)} G^R(\pi; \Lambda) + \varepsilon + \delta. && \text{(by equation 9)}
\end{aligned}$$

Therefore, $\pi^{(2)} \in \operatorname{arg-}(\varepsilon{+}\delta)\text{-}\min_{\pi \in \Pi} \max_{\Lambda \in \Delta(\Theta)} G^R(\pi; \Lambda) = \Pi^{\text{MMER(1)}}_{\varepsilon+\delta}$.

(1 ⊆ 3): Suppose $\pi^{(1)} \in \Pi^{\text{MMER(1)}}_{\varepsilon} = \underset{\pi\in\Pi}{\text{arg-}\varepsilon\text{-min}} \max_{\Lambda\in\Delta(\Theta)} G^R(\pi;\Lambda)$. Note also that, by definition, for all $\pi \in \Pi$, $\lambda(\pi) \in \underset{\Lambda\in\Delta(\Theta)}{\text{arg-}\eta\text{-max}}\, G^R(\pi;\Lambda)$. Then we have

$$\begin{aligned}
G^R(\pi^{(1)};\lambda(\pi^{(1)})) &\le \max_{\Lambda\in\Delta(\Theta)} G^R(\pi^{(1)};\Lambda) && \text{(by definition of max)}\\
&\le \min_{\pi\in\Pi}\max_{\Lambda\in\Delta(\Theta)} G^R(\pi;\Lambda) + \varepsilon && (\pi^{(1)} \in \underset{\pi\in\Pi}{\text{arg-}\varepsilon\text{-min}} \max_{\Lambda\in\Delta(\Theta)} G^R(\pi;\Lambda))\\
&\le \min_{\pi\in\Pi}\left(G^R(\pi;\lambda(\pi)) + \eta\right) + \varepsilon && \text{(by definition of } \lambda\text{; monotonicity of min)}\\
&\le \min_{\pi\in\Pi} G^R(\pi;\lambda(\pi)) + \eta + \varepsilon. && (\eta \text{ constant wrt. } \pi)
\end{aligned}$$

Therefore, $\pi^{(1)} \in \text{arg-}(\varepsilon{+}\eta)\text{-min}_{\pi\in\Pi}\, G^R(\pi;\lambda(\pi)) = \Pi^{\text{MMER(3)}}_{\varepsilon+\eta,\lambda}$.

(3 ⊆ 1): Suppose $\pi^{(3)} \in \Pi^{\text{MMER(3)}}_{\varepsilon,\lambda} = \underset{\pi\in\Pi}{\text{arg-}\varepsilon\text{-min}}\, G^R(\pi;\lambda(\pi))$. Then we have

$$\begin{aligned}
\max_{\Lambda\in\Delta(\Theta)} G^R(\pi^{(3)};\Lambda) &\le G^R(\pi^{(3)};\lambda(\pi^{(3)})) + \eta && \text{(by definition of } \lambda)\\
&\le \min_{\pi\in\Pi} G^R(\pi;\lambda(\pi)) + \varepsilon + \eta && (\pi^{(3)} \in \underset{\pi\in\Pi}{\text{arg-}\varepsilon\text{-min}}\, G^R(\pi;\lambda(\pi)))\\
&\le \min_{\pi\in\Pi}\max_{\Lambda\in\Delta(\Theta)} G^R(\pi;\Lambda) + \varepsilon + \eta.\\
& && \text{(by definition of max; monotonicity of min)}
\end{aligned}$$

Therefore, $\pi^{(3)} \in \text{arg-}(\varepsilon{+}\eta)\text{-min}_{\pi\in\Pi}\max_{\Lambda\in\Delta(\Theta)} G^R(\pi;\Lambda) = \Pi^{\text{MMER(1)}}_{\varepsilon+\eta}$.

(2 ⊆ 3): follows from (2 ⊆ 1) and (1 ⊆ 3).

(3 ⊆ 2): follows from (3 ⊆ 1) and (1 ⊆ 2). □

Remarks (Generalization to non-finite environments)**.** Unlike our other results, Proposition 3 relies on the assumption that the UMDP is finite. This assumption guarantees that there exists a Nash equilibrium for the game in Definition 7. The definitions and results can be generalized to infinite environments, so long as the necessary minima and maxima are defined. However, if there do not exist (approximate) Nash equilibria at some approximation thresholds, then it becomes necessary to account for the possibility that the set of MMER(2) policies is empty. We can generalize the relations in terms of the **minimax gap**,

$$\Delta = \min_{\pi\in\Pi}\max_{\Lambda\in\Delta(\Theta)} G^R(\pi;\Lambda) - \max_{\Lambda\in\Delta(\Theta)}\min_{\pi\in\Pi} G^R(\pi;\Lambda) \ge 0.$$

The minimax gap is always non-negative when it is well-defined, due to the max–min inequality. If the game permits Nash equilibria, the max–min inequality is an equality and $\Delta = 0$. In general, if the game permits an (ε,δ)-equilibrium for some $\varepsilon,\delta \ge 0$, then $\Delta \le \varepsilon + \delta$, and there exists $\varepsilon,\delta \ge 0$ and an (ε,δ)-equilibrium such that $\Delta = \varepsilon + \delta$.

We can now extend the proof of Proposition 3 to derive the following relations:

$$\begin{aligned}
\Pi^{\text{MMER(1)}}_{\varepsilon} &\subseteq \Pi^{\text{MMER(2)}}_{\varepsilon+\Delta,\varepsilon+\Delta} & \qquad \Pi^{\text{MMER(2)}}_{\varepsilon,\delta} &\subseteq \Pi^{\text{MMER(1)}}_{\varepsilon+\delta-\Delta}\\
\Pi^{\text{MMER(2)}}_{\varepsilon,\delta} &\subseteq \Pi^{\text{MMER(3)}}_{\varepsilon+\delta-\Delta+\eta,\lambda} & \qquad \Pi^{\text{MMER(3)}}_{\varepsilon,\lambda} &\subseteq \Pi^{\text{MMER(2)}}_{\varepsilon+\eta+\Delta,\varepsilon+\eta+\Delta}.
\end{aligned}$$

The relations between MMER(1) and MMER(3), not involving MMER(2) or the existence of equilibria, are unchanged.

C.3 Generalizing the robustness result

In this section, we combine Theorem 2 and Proposition 3 to show robustness results for the two alternative definitions of approximate MMER policies (Definitions 7 and 8).

Corollary 1 (MMER(2) is robust to goal misgeneralization)**.** *Consider an UMDP $\langle \Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$, a pair of goals $R, \tilde{R}$, a proxy-distinguishing distribution shift $\langle \alpha, \beta, C, \Lambda^{\text{Train}}, \Lambda^{\text{Deploy}} \rangle$, and approximation thresholds $\varepsilon, \delta \geq 0$. Then*

$$\forall \pi^{\text{MMER}} \in \Pi^{\text{MMER(2)}}_{\varepsilon,\delta}, \text{ we have } \pi^{\text{MMER}} \in \text{arg-}(\varepsilon{+}\delta)\text{-max}_{\pi \in \Pi} V^R(\pi; \Lambda^{\text{Deploy}}).$$

Proof. Let $\pi^{\text{MMER}} \in \Pi^{\text{MMER(2)}}_{\varepsilon,\delta}$. We have $\pi^{\text{MMER}} \in \Pi^{\text{MMER(1)}}_{\varepsilon+\delta}$ by Proposition 3. The corollary follows by Theorem 2. □

Corollary 2 (MMER(3) is robust to goal misgeneralization)**.** *Consider an UMDP $\langle \Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$, a pair of goals $R, \tilde{R}$, a proxy-distinguishing distribution shift $\langle \alpha, \beta, C, \Lambda^{\text{Train}}, \Lambda^{\text{Deploy}} \rangle$, approximation thresholds $\varepsilon, \eta \geq 0$, and an η-approximate adversarial map λ. Then*

$$\forall \pi^{\text{MMER}} \in \Pi^{\text{MMER(3)}}_{\varepsilon,\lambda}, \text{ we have } \pi^{\text{MMER}} \in \text{arg-}(\varepsilon{+}\eta)\text{-max}_{\pi \in \Pi} V^R(\pi; \Lambda^{\text{Deploy}}).$$

Proof. Let $\pi^{\text{MMER}} \in \Pi^{\text{MMER(3)}}_{\varepsilon,\lambda}$. We have $\pi^{\text{MMER}} \in \Pi^{\text{MMER(1)}}_{\varepsilon+\eta}$ by Proposition 3. The corollary follows by Theorem 2. □

D Optimizing minimax expected regret is necessary if you want to prevent misgeneralization under all possible distribution shifts

In Section 5.2, we show that approximately optimizing the MMER objective is sufficient for preventing misgeneralization under a distribution shift. In this appendix, we show that if we want policies that are robust to *all possible* distribution shifts, then finding an approximate MMER policy is both sufficient *and also necessary.*

Corollary 3. *Consider an UMDP $\langle \Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$, a goal R, and an approximation threshold $\varepsilon \geq 0$. We have that*

$$\left(\forall \Lambda \in \Delta(\Theta), \pi \in \underset{\pi' \in \Pi}{\text{arg-}\varepsilon\text{-max}}\, V^R(\pi'; \Lambda) \right) \qquad \textit{if and only if} \qquad \pi \in \Pi^{\text{MMER}}_{\varepsilon}(R).$$

Proof. ($\Rightarrow$): Suppose $\forall \Lambda \in \Delta(\Theta), \pi \in \text{arg-}\varepsilon\text{-max}_{\pi' \in \Pi} V^R(\pi'; \Lambda)$. Then

$$\max_{\Lambda \in \Delta(\Theta)} G^R(\pi; \Lambda) = \max_{\Lambda \in \Delta(\Theta)} \left(\max_{\pi' \in \Pi} V^R(\pi'; \Lambda) - V^R(\pi; \Lambda) \right) \leq \varepsilon.$$

Since regret is non-negative it follows that $\pi \in \underset{\pi' \in \Pi}{\text{arg-}\varepsilon\text{-min}} \max_{\Lambda \in \Delta(\Theta)} G^R(\pi'; \Lambda) = \Pi^{\text{MMER}}_{\varepsilon}(R)$.

($\Leftarrow$): Apply Theorem 2 for each $\Lambda \in \Delta(\Theta)$. □

While we prove a general result that holds for all possible level distributions, we are mainly interested in its implications for goal misgeneralization. In particular, imagine having any sort of distribution shift from training to deployment. This theorem implies that any policy that generalizes in this case is an MMER policy. In principle, it would be possible to find some of the generalizing policies in this set by other training methods. However, it is clear that MMER (or any refinement) should be the training objective utilized if we want to be certain to be able to identify the entire set of solutions that *never* suffer from goal misgeneralization.

E Partially observable environments

In this section, we generalize equation (2) and Theorem 2 to partially observable environments, with a slight modification of the bound to account for the fact that it may no longer be possible for any policy to achieve zero expected regret on a given distribution of levels.

Rather than defining underspecified partially observable MDPs in detail, we consider an arbitrary subset of the space of policies $\Phi \subseteq \Pi$. We can model partial observability by restricting to policies with tied outputs within any given partition of $\Theta \times \mathcal{S}$ into information sets.

We define the **expected restricted regret** of a policy $\pi \in \Phi$ in a level $\theta \in \Theta$ under a goal R based on the return of the best available policy within such a subset of policies:

$$G^R_\Phi(\pi;\theta) = \max_{\pi' \in \Phi} V^R(\pi';\theta) - V^R(\pi;\theta). \tag{10}$$

As before, we lift this definition to a level distribution Λ by taking the expectation

$$G^R_\Phi(\pi;\Lambda) = \mathbb{E}_{\theta\sim\Lambda}\left[\, G^R_\Phi(\pi;\theta)\,\right]. \tag{11}$$

E.1 Generalizing the expected regret identity to partially observable environments

Proposition 4 (Expected regret identity for partially observable environments)**.** *Consider an UMDP $\langle \Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T}\rangle$, a goal R, a level distribution $\Lambda \in \Delta(\Theta)$, and a subset of policies $\Phi \subseteq \Pi$. Then*

$$G^R_\Phi(\pi;\Lambda) = \max_{\pi' \in \Phi} V^R(\pi';\Lambda) - V^R(\pi;\Lambda) + \min_{\pi' \in \Phi} G^R_\Phi(\pi';\Lambda).$$

Proof. Equivalently,

$$\begin{aligned}
&G^R_\Phi(\pi;\Lambda) - \min_{\pi' \in \Phi} G^R_\Phi(\pi';\Lambda)\\
&= \mathbb{E}_{\theta\sim\Lambda}\left[\, G^R_\Phi(\pi;\theta)\,\right] - \min_{\pi'\in\Phi} \mathbb{E}_{\theta\sim\Lambda}\left[\, G^R_\Phi(\pi';\theta)\,\right] && \text{(by equation 11)}\\
&= \mathbb{E}_{\theta\sim\Lambda}\left[\max_{\pi'\in\Phi} V^R(\pi';\theta)\right] - V^R(\pi;\Lambda) - \min_{\pi'\in\Phi}\left(\mathbb{E}_{\theta\sim\Lambda}\left[\max_{\pi''\in\Phi} V^R(\pi'';\theta)\right] - V^R(\pi';\Lambda)\right)\\
& && \text{(by equations 10 and 6)}\\
&= \mathbb{E}_{\theta\sim\Lambda}\left[\max_{\pi'\in\Phi} V^R(\pi';\theta)\right] - V^R(\pi;\Lambda) - \mathbb{E}_{\theta\sim\Lambda}\left[\max_{\pi''\in\Phi} V^R(\pi'';\theta)\right] + \max_{\pi'\in\Phi} V^R(\pi';\Lambda)\\
&= \max_{\pi'\in\Phi} V^R(\pi';\Lambda) - V^R(\pi;\Lambda).
\end{aligned}$$

□

Compared to equation (2) (Proposition 1), Proposition 4 includes the term $\min_{\pi\in\Phi} G^R_\Phi(\pi;\Lambda)$. This extra term represents the **irreducible (expected restricted) regret** for the level distribution Λ. Proposition 4 essentially says that the restricted expected regret for a level distribution can be decomposed into two components: (1) the shortfall in expected return compared to the optimal policy for the level distribution (as in equation 2, cf. the definition of regret for individual levels); plus (2) this irreducible regret.

Irreducible regret can arise when the level is partially observable to the policy. For example, consider a mixture of two levels with two disjoint sets of optimal policies. Suppose the level is not observed by the policy, so the policy has to choose actions without knowing whether it is in the first level or the second level. In each individual level, we define expected regret based on the performance of optimal policies for that level (these policies will naturally be the ones that *assume* they are in the appropriate level). However, since no single policy can perform optimally in both levels, the expected regret with respect to the mixture is always nonzero. This nonzero minimum expected regret is exactly the irreducible regret.

E.2 Generalizing Definition 5 and Theorem 2 to partially observable environments

We are now in position to generalize the results of Section 5.2 to partially observable environments. First, we adapt Definition 5 to expected restricted regret.

Definition 9 (Approximate MMER for partially observable environments)**.** *Consider an UMDP $\langle\Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T}\rangle$, a goal R, an approximation threshold $\varepsilon, \delta \geq 0$, and a subset of policies $\Phi \subseteq \Pi$. The* restricted approximate MMER policy set *is then*

$$\Pi^{\text{MMER}}_{\Phi,\varepsilon}(R) = \underset{\pi' \in \Phi}{\text{arg-}\varepsilon\text{-min}} \max_{\Lambda \in \Delta(\Theta)} G^R_\Phi(\pi'; \Lambda).$$

Next, we generalize Theorem 2. In doing so, we need to account for the **irreducible regret gap**

$$g(\Lambda^{\text{Deploy}}) = \min_{\pi \in \Phi} \max_{\Lambda \in \Delta(\Theta)} G^R_\Phi(\pi; \Lambda) - \min_{\pi \in \Phi} G^R_\Phi(\pi; \Lambda^{\text{Deploy}}), \tag{12}$$

the difference in irreducible regret between an MMER level distribution and Λ^{Deploy}. When the deployment distribution has lower irreducible regret than an MMER level distribution, the agent has no incentive to improve expected restricted regret on the deployment distribution once it is below the irreducible regret of the MMER distribution.

If the irreducible regret gap is large, then this undermines the robustness guarantee. This is a limitation of standard MMER. However, we note that it can be addressed by a lexicographic refinement of the decision rule along the lines of Beukman et al. (2024).

Theorem 3 (MMER is robust to goal misgeneralization in partially observable environments)**.** *Consider an UMDP $\langle\Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T}\rangle$, a pair of goals $R, \tilde{R}$, a proxy-distinguishing distribution shift $\langle\alpha, \beta, C, \Lambda^{\text{Train}}, \Lambda^{\text{Deploy}}\rangle$, an approximation threshold $\varepsilon \geq 0$, and a subset of policies $\Phi \subseteq \Pi$. Let $g(\Lambda^{\text{Deploy}}) = \min_{\pi \in \Phi} \max_{\Lambda \in \Delta(\Theta)} G^R_\Phi(\pi; \Lambda) - \min_{\pi \in \Phi} G^R_\Phi(\pi; \Lambda^{\text{Deploy}})$ be the irreducible regret gap. Then*

$$\forall \pi^{\text{MMER}} \in \Pi^{\text{MMER}}_{\Phi,\varepsilon}(R), \text{ we have } \pi^{\text{MMER}} \in \text{arg-}(\varepsilon + g(\Lambda^{\text{Deploy}}))\text{-max}_{\,\pi \in \Phi} V^R(\pi; \Lambda^{\text{Deploy}}).$$

Proof. Suppose $\pi^{\text{MMER}} \in \Pi^{\text{MMER}}_{\Phi,\varepsilon}(R)$. Then, along similar lines to the proof of Theorem 2, we have the following bound on expected restricted regret:

$$\begin{aligned} G^R_\Phi(\pi^{\text{MMER}}; \Lambda^{\text{Deploy}}) &\leq \max_{\Lambda \in \Delta(\Theta)} G^R_\Phi(\pi^{\text{MMER}}; \Lambda) && (\Lambda^{\text{Deploy}} \in \Delta(\Theta)) \\ &\leq \min_{\pi \in \Phi} \max_{\Lambda \in \Delta(\Theta)} G^R_\Phi(\pi; \Lambda) + \varepsilon. && \text{(by Definition 9)} \end{aligned}$$

Once again, we can convert this upper bound on expected restricted regret to a lower bound on expected return:

$$\begin{aligned} &V^R(\pi^{\text{MMER}}; \Lambda^{\text{Deploy}}) \\ &= \max_{\pi \in \Phi} V^R(\pi; \Lambda^{\text{Deploy}}) - G^R_\Phi(\pi^{\text{MMER}}; \Lambda^{\text{Deploy}}) + \min_{\pi \in \Phi} G^R_\Phi(\pi; \Lambda^{\text{Deploy}}) && \text{(by Proposition 4)} \\ &\geq \max_{\pi \in \Phi} V^R(\pi; \Lambda^{\text{Deploy}}) - \left(\min_{\pi \in \Phi} \max_{\Lambda \in \Delta(\Theta)} G^R_\Phi(\pi; \Lambda) + \varepsilon\right) + \min_{\pi \in \Phi} G^R_\Phi(\pi; \Lambda^{\text{Deploy}}) && \text{(by above bound)} \\ &= \max_{\pi \in \Phi} V^R(\pi; \Lambda^{\text{Deploy}}) - \varepsilon - g(\Lambda^{\text{Deploy}}). && \text{(by equation 12)} \end{aligned}$$

Thus, $\pi^{\text{MMER}} \in \text{arg-}(\varepsilon + g(\Lambda^{\text{Deploy}}))\text{-max}_{\,\pi \in \Phi} V^R(\pi; \Lambda^{\text{Deploy}})$. □

F Additional environment details

In this appendix, we provide additional details about each environment, including details about procedurally generating non-distinguishing and distinguishing levels, edit distributions, computation of maximum level value for the oracle-latest estimator, and the origin of each environment.

F.1 The CHEESE IN THE CORNER environment

Environment. In this environment, levels are parameterized by a 13×13 wall layout, a mouse spawn position within this grid, and a cheese position within the grid. We require that the cheese position and the mouse spawn position are not equal, and moreover that they are not obstructed by walls. We do not require that there is an unobstructed path between them.

In the initial state, the mouse begins in the spawn position. The actions available to the agent are to attempt to move the mouse up, left, down, or right, which succeeds if the respective position is not obstructed by a wall or the edge of the grid. If the mouse position equals the cheese position, the mouse collects the cheese. The episode terminates when the cheese has been collected or after a maximum of 128 steps.

Observations. We represent states to the agent as a $15 \times 15 \times 3$ Boolean grid. The first of the three channels encodes the maze layout, including a border of width 1. The second channel one-hot encodes the position of the mouse. The third channel one-hot encodes the position of the cheese (if it has not been collected). All of the relevant information about the level and the state is encoded into this observation, therefore this environment is fully observable.

True goal and proxy goal. The training goal is for the mouse to collect the cheese. The reward function assigns $+1$ reward to transitions in which the mouse collects the cheese.

We also consider a proxy goal of navigating to the top left corner of the maze. This could be formulated as a reward function that assigns $+1$ reward the first time the mouse steps into the top left corner (this reward can be made Markovian by augmenting the state with a flag for whether the corner has previously been reached). Note that we never train with this proxy goal as the reward function.

Level classification. Given this environment and this pair of goals, we can classify levels according to the definitions in Section 4. Note that in the following, we assume that the discount factor is strictly between 0 and 1 (so that shorter paths obtain higher return), and that the property of *reachability* accounts for episode termination conditions (including collecting the cheese).

1. **Levels for which the cheese is in the top-left corner of the maze are non-distinguishing.** If the top-left corner is reachable from the mouse spawn position, optimally pursuing the proxy goal implies following a shortest path to the corner, which is also a shortest path to collecting the cheese. If the top-left corner is unreachable from the mouse spawn position, then all policies achieve zero return according to either goal, and are therefore equally optimal.
2. **Most levels for which the cheese is not in the top-left corner of the maze are distinguishing.** If the cheese is not in the top-left corner, but is reachable from the mouse spawn position, then either (1) the top-left corner is reachable from the mouse spawn position (other than via the cheese), or (2) it is not. If (1), then a proxy-optimal policy could visit the top-left corner and then remain there until the episode terminates. If (2), then all policies are optimal under the proxy goal, and in particular there exists a proxy-optimal policy that avoids the cheese.
3. **Some levels for which the cheese is in the top-left corner are non-distinguishing.** If the cheese is not in the top-left corner, and is not reachable from the mouse spawn position, then all policies are optimal under the true goal, and in particular all proxy-optimal policies are.

When defining the proportion of distinguishing levels in the buffer, we use the approximately correct approach of checking whether the cheese is not in the corner. We note that all UED algorithms rapidly remove levels in which the cheese is unreachable from their buffers.

Procedural level generation. We construct two procedural level generators, $\Lambda_{\neg\text{Distg.}}, \Lambda_{\text{Distg.}} \in \Delta(\Theta)$, (approximately) concentrated on non-distinguishing and distinguishing levels, respectively.

- **Non-distinguishing level distribution ($\Lambda_{\neg\text{Distg.}}$).** We procedurally generate non-distinguishing levels as follows. We position the cheese in the top-left corner. For each remaining position, we place a wall independently with probability 25%. We position the mouse spawn in some remaining position, assuming there is at least one such position.

  All of the generated levels are technically non-distinguishing, though this may include levels for which the cheese position is unreachable from the mouse spawn position. Note that these levels still do not provide any training signal in favor of the true goal over the proxy goal.

- **Distinguishing level distribution ($\Lambda_{\text{Distg.}}$).** We procedurally generate *mostly* distinguishing levels as follows. For each position, we place a wall independently with probability 25%. We position the cheese somewhere where there is not a wall. We position the mouse spawn somewhere where there is not a wall, different from the cheese position.

  The generated levels are in most cases C-distinguishing levels for some C. It may arise that the cheese is unreachable from the mouse spawn location or is in the top-left corner, making the level non-distinguishing. Note that the effect of these levels is only to reduce the training signal in favor of the true goal available to the agent, as if to reduce α.

Elementary edit distributions. ACCEL additionally requires specifying an edit distribution used to sample "similar" levels for potential entry into the level buffer. In Appendix J, we discuss how we build our edit distributions from elementary edit distributions of the following three types.

1. **Classification preserving edits.** Each classification preserving edit either removes an existing wall, positions a new wall, or moves the mouse spawn position to a random unobstructed position (other than the cheese position). These edits don't change the cheese position, though they may change whether the cheese position is reachable from the mouse spawn position. As a result, they may change the exact classification of the level (though not its approximate classification).
2. **Biased classification transforming edits.** Given a probability α, a biased classification transforming edit randomizes the cheese position with probability α or places the cheese in the top-left corner with probability $1 - \alpha$. Note that when randomizing the cheese position, it's possible that the cheese will be positioned in the top-left corner.
3. **Unrestricted classification transforming edits.** An unrestricted classification transforming edit randomizes the cheese position with probability 1. It's possible that the cheese will be positioned in the top-left corner.

Oracle maximum return. In this environment, an optimal policy follows any shortest path from the mouse spawn position to the cheese position in the graph representing the maze layout. We compute the length of a shortest path using the Floyd–Warshall algorithm. Given a level $\theta \in \Theta$. Let R be the true reward function ($+1$ for collecting the cheese). If the shortest path has length $d \in \mathbb{N} \cup \{\infty\}$, then with discount factor $\gamma \in (0, 1)$, we have

$$\max_{\pi} V^R(\pi; \theta) = \gamma^{d-1}. \tag{13}$$

This covers the case in which the cheese position is unreachable from the mouse spawn position ($d = \infty$). Shortest paths of (finite) length greater than the maximum episode length of 128 are impossible given this grid size.

Origin. Hubinger (2019) originated the idea of creating a navigation task with a location proxy as a potential means of inducing goal misgeneralization. Langosco et al. (2022) implemented the MAZE I based on this idea by modifying the MAZE environment of OpenAI ProcGen (Cobbe et al., 2020) such that the cheese position could be restricted to a region surrounding the top-right corner (cf., Appendix M). CHEESE IN THE CORNER is an interpretation of MAZE I using an original JAX implementation. We depart from MAZE I by using a fixed size maze and replacing the maze layout algorithm with a simpler algorithm based on random block placement.

F.2 The CHEESE ON A DISH environment

Environment. In this environment, levels are parameterized by a 13×13 wall layout, and positions within this grid for the mouse spawn, the dish, and the cheese. We require that the cheese position and the mouse spawn position are not equal, and that the dish position and the mouse spawn position are not equal, though the dish can be co-located with the cheese. Moreover, we require that none of the three positions are obstructed by walls. We do not require that there is an unobstructed path between the positions.

In the initial state, the mouse begins in the spawn position. The actions available to the agent are to attempt to move the mouse up, left, down, or right, which succeeds if the respective position is not obstructed by a wall or the edge of the grid. If the mouse position equals the cheese position, the mouse collects the cheese, and likewise for the dish. The episode terminates when the cheese *or* the dish has been collected or after a maximum of 128 steps.

Observations. We represent states to the agent as a $15 \times 15 \times (3 + D)$ Boolean grid where $D \in \mathbb{N}$. The first of the channels encodes the maze layout, including a border of width 1. The second channel one-hot encodes the position of the mouse. The third channel one-hot encodes the position of the cheese (if it has not been collected). The remaining D channels each one-hot encode the dish position (if the dish has not been collected).

Redundantly coding the dish encourages the agent to learn a policy that is based on the dish position, eliciting a clearer case of goal misgeneralization. In our main experiments, $D = 6$. In Appendix N, we consider alternative values of D. All of the relevant information about the level and the state is encoded into this observation, therefore this environment is fully observable.

True goal and proxy goal. The training goal is for the mouse to collect the cheese. The reward function assigns $+1$ reward to transitions in which the mouse collects the cheese. We also consider a proxy goal of collecting the dish. This reward function assigns $+1$ reward to transitions in which the mouse collects the dish. Note that we never train with this proxy goal as the reward function.

Level classification. Given this environment and this pair of goals, we can classify levels according to the definitions in Section 4. Note that in the following, we assume that the discount factor is strictly between 0 and 1 (so that shorter paths obtain higher return), and that the property of *reachability* accounts for episode termination conditions.

1. **Levels for which the cheese and the dish are co-located are non-distinguishing.** If the cheese/dish position is reachable from the mouse spawn position, optimally pursuing the proxy goal implies following a shortest path to the dish, which is also a shortest path to the cheese. If the cheese/dish position is unreachable from the mouse spawn position, then all policies achieve zero return under either goal, and are therefore equally optimal.
2. **Most levels for which the cheese is not in the same position as the dish are distinguishing.** If the cheese is not in the same position as the dish, but is reachable from the mouse spawn position, then either (1) the dish is reachable from the mouse spawn position (other than via the cheese), or (2) it is not. If (1), then a proxy-optimal policy could visit the dish, terminating the episode. If (2), then all policies are optimal under the proxy goal, and in particular there exists a proxy-optimal policy that avoids the cheese.
3. **Some levels for which the cheese is not in the same position as the dish are non-distinguishing.** If the cheese is not in the same position as the dish, and moreover is not reachable from the mouse spawn position, then all policies are optimal under the true goal, and in particular all proxy-optimal policies are.

When defining the proportion of distinguishing levels in the buffer, we use the approximately correct approach of checking whether the cheese and the dish have distinct positions. We note that all UED algorithms rapidly remove levels in which the cheese is unreachable from their buffers.

Procedural level generation. We construct two procedural level generators, $\Lambda_{\neg\text{Distg.}}, \Lambda_{\text{Distg.}} \in \Delta(\Theta)$, (approximately) concentrated on non-distinguishing and distinguishing levels, respectively.

- **Non-distinguishing level distribution** ($\Lambda_{\neg\text{Distg.}}$)**.** We procedurally generate non-distinguishing levels as follows. For each position, we place a wall independently with probability 25%. We position the mouse spawn somewhere where there is not a wall. We position the cheese and the dish somewhere where there is not a wall, other than the mouse spawn position (we assume there are at least two positions without walls).

  All of the generated levels are technically non-distinguishing, though this may include levels for which the cheese position is unreachable from the mouse spawn position. Note that these levels still do not provide any training signal in favor of the true goal over the proxy goal.

- **Distinguishing level distribution** ($\Lambda_{\text{Distg.}}$)**.** We procedurally generate *mostly* distinguishing levels as follows. For each position, we place a wall independently with probability 25%. We position the mouse spawn somewhere where there is not a wall. We position the cheese and the dish, independently, somewhere where there is not a wall, different from the mouse spawn position.

  The generated levels are in most cases C-distinguishing levels for some C. It may arise that the cheese is unreachable from the mouse spawn location or is in the top-left corner, making the level non-distinguishing. Note that the effect of these levels is only to reduce the training signal in favor of the true goal available to the agent, as if to reduce α.

Elementary edit distributions. ACCEL additionally requires specifying an edit distribution used to sample "similar" levels for potential entry into the level buffer. In Appendix J, we discuss how we build our edit distributions from elementary edit distributions of the following three types.

1. **Classification preserving edits.** Each classification preserving edit either removes an existing wall, positions a new wall, or moves the mouse spawn position to a random unobstructed position (other than the cheese position or the dish position). These edits don't change the cheese position or the dish position, though they may change whether the cheese position is reachable from the mouse spawn position. As a result, they may change the exact classification of the level (though not its approximate classification).

2. **Biased classification transforming edits.** Given a probability α, a biased classification transforming edit randomizes the dish position, and then either independently randomizes the cheese position (with probability α) or positions the cheese at the new dish position (with probability $1 - \alpha$). Note that when independently randomizing the cheese position, it's possible that the cheese will be co-located with the dish.

3. **Unrestricted classification transforming edits.** An unrestricted classification transforming edit independently randomizes the cheese position and the dish position with probability 1. It's possible that the cheese and the dish will be co-located.

Oracle maximum return. In this environment, an optimal policy follows any shortest path from the mouse spawn position to the cheese position in the graph representing the maze layout (in which the dish counts as an obstruction if it is not co-located with the cheese, since collecting the dish terminates the episode). Given this graph we compute the maximum expected return as in (13).

Origin. Langosco et al. (2022) originally implemented the MAZE II environment by modifying the MAZE environment from OpenAI ProcGen (Cobbe et al., 2020) such that the cheese is replaced by a yellow diamond during training, and subsequently by a pair of objects (a red diamond and a yellow diagonal line) during evaluation. In their setup, the yellow diamond is rewarding because of its shape (diamond), rather than its color (yellow), so the intended extrapolation of the goal is for the policy to pursue the red diamond. However, Langosco et al. (2022) found that learned policies tend to pursue the yellow line instead. CHEESE ON A DISH is an interpretation of MAZE II using an original JAX implementation. In addition to the differences discussed for CHEESE IN THE CORNER, we break the symmetry between the dish and the cheese by redundantly coding the dish (since we use Boolean grid observations rather than colored images).

F.3 The KEYS AND CHESTS environment

Environment. In this environment, levels are parameterized by a 13×13 wall layout, and positions within this grid for the mouse spawn, $k \leq 10$ keys, and $c \leq 10$ chests. We require that the mouse spawn position and the key and chest positions are distinct and are not obstructed by walls. We do not require that there are unobstructed paths between the positions.

In the initial state, the mouse begins in the spawn position. The actions available to the agent are to attempt to move the mouse up, left, down, or right, which succeeds if the respective position is not obstructed by a wall or the edge of the grid. If the mouse position equals a key position, it collects the key, removing it from the maze and adding it to the mouse's inventory. If the mouse position equals a chest position, assuming it has at least one key in its inventory, it collects the chest, removing it from the maze and removing the key from its inventory. The mouse can occupy the same position as a chest if it has an empty inventory. In a level with k keys and c chests, the episode terminates when $\min(k, c)$ chests have been collected, or after a maximum of 128 steps.

Observations. We represent states to the agent as a $15 \times 15 \times 5$ Boolean grid. The first channel encodes the maze layout, including a border of width 1. The second one-hot encodes the position of the mouse. The third encodes the positions of all the as-yet-uncollected chests. The fourth encodes the positions of all as-yet-uncollected keys. The fifth encodes the mouse's inventory, with one cell along the top row of the channel for each key. All of the relevant information about the level and the state is encoded into this observation, therefore this environment is fully observable.

True goal and proxy goal. The training goal is to collect chests. The reward function assigns $+1$ reward to transitions in which the mouse collects a chest. This reward function is not normalized like those we consider in the theory sections, but it is bounded and could easily be normalized. Under this goal, collecting keys has no intrinsic value, but since collecting a chest requires collecting a key, keys have instrumental value. We also consider a proxy goal that assigns reward for collecting keys as well as chests. This goal could be modeled as a reward function that assigns, for example, $1 - \eta$ reward for collecting each key and η reward for collecting each chest, where $\eta \in (0, 1)$. Note that we never train with such a proxy goal as the reward function.

Level classification. Compared to the other environments, describing this environment in terms of the definitions in Section 4 is not as straight forward. Optimal behavior involves collecting keys and chests in an order that depends on subtle tradeoffs driven by the exponential discounting. For the true goal, the optimal behavior is to collect as many chests as fast as possible. However, it may make sense to make a brief detour to collect multiple keys if it slightly delays the collection of the next chest, as long as this sufficiently accelerates the collection of subsequent chests. The proxy goal rewards key collection for its own sake, so as to increase the incentive for the policy to take larger and larger detours to collect keys and even collect more keys than there are reachable chests. In our experiments, we mainly consider the following kinds of levels.

1. **Most levels with 3 keys and 10 chests are approximately non-distinguishing.** Consider a level in which there are $k \approx 3$ reachable keys, and $c \approx 10$ reachable chests. These levels are approximately non-distinguishing, in the sense that while the proxy goal incentivizes collecting keys earlier than optimal, it still incentivizes eventually collecting chests. For many key/chest layouts, pursuing the proxy goal entails similar behavior to pursing the true goal.
2. **Most levels with 10 keys and 3 chests are approximately distinguishing.** Consider a level in which there are $k \approx 10$ reachable keys, and $c \approx 3$ reachable chests. These levels are mostly distinguishing because optimizing the proxy goal will usually involve long detours to collect all reachable keys, delaying the collection of chests compared to optimizing the true goal.

The exact classifications depend on the key/chest layout. Keys and chests that are unreachable from the mouse spawn position have no influence on optimal behavior. For the purpose of measuring the proportion of distinguishing levels, we approximately classify the level by the total number of keys and chests in the level without accounting for reachability or the exact key/chest layout (non-distinguishing levels have 3 keys and 10 chests, distinguishing levels have 10 keys and 3 chests).

Procedural level generation. We construct two procedural level generators, $\Lambda_{\neg\text{Distg.}}, \Lambda_{\text{Distg.}} \in \Delta(\Theta)$, (approximately) concentrated on non-distinguishing and distinguishing levels, respectively.

- **Non-distinguishing level distribution** $(\Lambda_{\neg\text{Distg.}})$**.** We procedurally generate *mostly* non-distinguishing levels as follows. For each position, we place a wall independently with probability 25%. We then position the mouse spawn, 3 keys, and 10 chests in distinct, unobstructed positions (assuming there are enough positions). The generated levels are usually approximately non-distinguishing in the sense of the classification system described above. It is possible that fewer than 3 keys and 10 chests will be reachable from the mouse spawn position, and it is even possible that the exact layout will lead to a significant disadvantage for a policy that over-prioritizes key collection.
- **Distinguishing level distribution** $(\Lambda_{\text{Distg.}})$**.** We procedurally generate *mostly* distinguishing levels in the same fashion, but with 10 keys and 3 chests instead. The generated levels are usually distinguishing, along the lines of the classification system described above. It is possible that fewer than 10 keys and 3 chests will be reachable from the mouse spawn position, and it is possible that the exact layout will not substantially disincentivize key collection (for example, all keys may be positioned on a shortest path through the set of chests).

Elementary edit distributions. ACCEL additionally requires specifying an edit distribution used to sample "similar" levels for potential entry into the level buffer. In Appendix J, we discuss how we build our edit distributions from elementary edit distributions of the following three types.

1. **Classification preserving edits.** Each classification preserving edit either removes an existing wall, positions a new wall, or moves the mouse spawn position or the position of one key or chest to a random unobstructed, unoccupied position. These edits don't change the number of keys or chests, though they may change whether these positions are reachable from the mouse spawn position. As a result, they may change the exact classification of the level (though not its approximate classification).
2. **Biased classification transforming edits.** Given a probability α, a biased classification transforming edit sets the number of keys and chests to that of the distinguishing level generator (10 and 3) with probability α, or that of the non-distinguishing level generator (3 and 10) with probability $1 - \alpha$.
3. **Unrestricted classification transforming edits.** An unrestricted classification transforming edit is a biased classification transforming edit with α set to 50%.

Oracle maximum return. As described above, an optimal policy in this environment collects keys and chests in the some order so as so collect as many chests as fast as possible. In particular, noting that at most $m = \min(k, c)$ chests can be collected in a given level (due to the requirement that a key must be expended to collect a chest), and supposing that they are collected after steps $s_1, s_2, \ldots, s_m \in \mathbb{N} \cup \{\infty\}$, the return is given by

$$R(s_1, s_2, \ldots, s_m) = \sum_{i=1}^{m} \gamma^{s_i - 1} \tag{14}$$

where $\gamma \in (0, 1)$ is the discount factor.

Our approach to computing the optimal value is a to enumerate a subset of paths through the network of key/chest/mouse spawn positions that must contain an optimal path, to brute-force evaluate a lower bound on the return of these paths, and to identify the greatest return lower bound as the optimal return for the level. We explain the procedure in detail as follows.

1. We begin by enumerating a set of so-called *viable* collection sequences. Each collection sequences comprising an m-permutation of the k keys, an m-permutation of the c chests, and a Dyck path of order m (a permutation of m keys and m chests such that each chest is preceded by a corresponding key, cf. balanced parentheses). These three combinatorial objects jointly identify a sequence in which particular keys and chests could be collected.

For example, suppose $k = 3$ and $c = 10$ and number the keys $\mathrm{k}_1, \ldots, \mathrm{k}_3$ and the chests $\mathrm{c}_1, \ldots, \mathrm{c}_{10}$. Suppose the key 3-permutation is $(3\ 1\ 2)$, the chest 3-permutation is $(1\ 6\ 4)$, and the Dyck path is $(\mathrm{k\ c\ k\ k\ c\ c})$. Then the corresponding collection sequence is $\mathrm{k}_3, \mathrm{c}_1, \mathrm{k}_1, \mathrm{k}_2, \mathrm{c}_6, \mathrm{c}_4$.

The total number of viable collection sequences is

$$\underbrace{\binom{k}{m} \cdot m!}_{\text{key } m\text{-permutations}} \times \underbrace{\binom{c}{m} \cdot m!}_{\text{chest } m\text{-permutations}} \times \underbrace{\frac{1}{m+1}\binom{2m}{m}}_{\text{order } m \text{ Dyck paths}}. \tag{15}$$

Since, in our setup, either $k = 3$ and $c = 10$ or vice versa, $m = 3$ and (15) evaluates to 21,600.

2. We evaluate a lower bound on the return of each viable collection sequence as follows. First we compute all-pairs shortest path distances for the mouse spawn position, each key position, and each chest position, using the Floyd–Warshall algorithm. We represent unreachable pairs with a distance of ∞. We then simulate each sequence, computing the step counts required for each collection as the cumulative sum of pairwise shortest path distances for each transition along the sequences, starting at the mouse spawn position. We round any distances above 128 up to ∞. We then use the step counts of each of the m chest collections in the expression (14).

   For example, consider the collection sequence described earlier, $\mathrm{k}_3, \mathrm{c}_1, \mathrm{k}_1, \mathrm{k}_2, \mathrm{c}_6, \mathrm{c}_4$. Let $D(p, q)$ represent the shortest path distance between the positions of objects p and q. Then we set $s_{\mathrm{c}_1} = D(\mathrm{spawn}, \mathrm{c}_1)$, $s_{\mathrm{k}_3} = s_{\mathrm{c}_1} + D(\mathrm{c}_1, \mathrm{k}_3)$, and so on until $s_{\mathrm{c}_4} = s_{\mathrm{c}_6} + D(\mathrm{c}_6, \mathrm{c}_4)$. (If any of these step counts pass the timeout of 128, we round them up to ∞ to account for termination.) Finally, we compute the lower bound on the return of this sequence as $\gamma^{s_{\mathrm{c}_1}-1} + \gamma^{s_{\mathrm{c}_6}-1} + \gamma^{s_{\mathrm{c}_4}-1}$.

   We note that if a viable collection sequence ever involves collecting a key or chest that is unreachable from the mouse spawn position, then the step count for this collection and all subsequent collections will be infinite and thus this neither this collection nor subsequent collections will contribute to the return lower bound.

3. The greatest return lower bound across all viable collection sequences equals the maximum return achievable, as follows. Following a shortest path between a given pair of positions may involve crossing over other keys and chests. This can have any of several effects that invalidate the collection sequence, including (1) collecting keys or chests that appear later in the sequence earlier than planned, (2) expending keys before they are intended to be spent to collect a chest later in the sequence, or (3) terminating the episode early due to collecting the maximum available number of chests before finishing the sequence. However, such disruptions only ever *increase* the return. Moreover, for each disrupted sequence, there is a viable collection sequence that represents the actual order of keys and chests, except for accounting for the case where more than 3 keys are collected (which has no affect on the return as long as they are on the shortest paths to the necessary chests). It follows that for the viable collection sequence with the highest return, there are no such disruptions, and the return lower bound is tight.

We note that the set of viable collection sequences could be further filtered by eliminating (or never enumerating) sequences involving unreachable keys or chests. However, we use the above approach for simplicity and uniformity. In particular, since the above approach yields a fixed computational graph given values of k, c, we can enumerate the 21,600 viable sequences once at compile time and accelerate the brute-force evaluation step for a particular level (or batch of levels) using JAX. To handle a mixture of levels with $(k, c) = (3, 10)$ and levels with $(k, c) = (10, 3)$, we simply evaluate both ways and then dynamically keep the appropriate result for each level.

Origin. Barnett (2019) originated the idea of creating a distribution shift from a navigation environment in which keys are scarce to one in which keys are not scarce. Langosco et al. (2022) originally implemented a KEYS AND CHESTS environment by modifying the HEIST environment from OpenAI ProcGen (Cobbe et al., 2020). Our KEYS AND CHESTS environment is an interpretation of the environment from Langosco et al. (2022), implemented in JAX, with several changes similar to those for the other environments.

G Additional training details

G.1 Hyperparameters

Table G.1: Hyperparameters used for all methods and environments.

Parameter	Value	Notes/exceptions
Rollouts		
# parallel environments	256	
Rollout length	128	
# environment steps per cycle	32,768	(# parallel environments × rollout length)
Discount factor, γ	0.999	
GAE		
λ_{GAE}	0.95	
PPO loss function		
Clip range	0.1	
Value clipping	yes	
Critic coefficient	0.5	
Entropy coefficient	1e-3	Or, 1e-2 for KEYS AND CHESTS
PPO updates		
Epochs per cycle	5	
Minibatches per epoch	4	
Max gradient norm	0.5	
Adam learning rate	5e-5	
Learning rate schedule	constant	
UED configuration		N/A for DR
Replay rate		
PLR$^{\perp}$	0.33	On average, 1 replay cycle per 2 generate cycles
ACCEL	0.5	On average, 1 replay cycle per 1 generate cycle (1 edit cycle immediately follows each replay cycle)
Buffer size	4096	
Prioritization method	rank	
Temperature	0.1	
Staleness coefficient	0.1	
# elementary edits per level, n	12	N/A for PLR$^{\perp}$

G.2 Compute

We perform each training run on a single NVIDIA A100 80GB GPU. For CHEESE IN THE CORNER and CHEESE ON A DISH, each training run takes around 40 minutes (DR) or 80 minutes (UED methods). For KEYS AND CHESTS, each training run takes around 60 minutes (DR) or 110 minutes (UED methods). UED methods take longer than DR because UED methods require sampling additional rollouts for refining the buffer (the number of environment steps used for PPO updates is held constant across all methods). KEYS AND CHESTS runs are longer than others because we trained each method for 400 million steps instead of 200 million steps in this environment.

The experiments discussed in Section 7 took a total of 1.2k GPU hours. The additional experiments discussed in Appendices J, K, L, M, and N took, respectively, totals of 1.2k GPU hours; 500 GPU hours; 400 GPU hours (not counting the first 200 million environment steps used for training, which we counted with Section 7); 210 GPU hours; and 350 GPU hours.

H Additional evaluation results (non-distinguishing levels and proxy goal)

In Section 7, we investigate which training distributions and methods led to good performance on distinguishing levels. We claim that when the performance is low, this is an instance of goal misgeneralization, and therefore when the performance is high, goal misgeneralization has been prevented.

In order to justify this claim, we also check that the poor performance is explained primarily by the policy pursuing a proxy goal on distinguishing levels, rather than the policy behaving incapably in non-distinguishing or distinguishing levels. Figure H.1 shows (1) return on non-distinguishing levels and (2) proxy return on distinguishing levels for each training configuration.

Figure H.1(top row) shows that the learned policies perform capably in non-distinguishing levels. For CHEESE IN THE CORNER and CHEESE ON A DISH, all training methods achieve high return on non-distinguishing levels for all training distributions. For KEYS AND CHESTS, this is the case for all training distributions except the $\alpha = 1$ baseline (in this edge case, non-distinguishing levels, with few keys and many chests, are never seen during training).

Figure H.1(bottom row) shows the proxy return achieved by learned policies on distinguishing levels. Note that we never use the proxy goal for training. Here we simply evaluate the policies according to each environment's respective proxy reward function. In particular, for CHEESE IN THE CORNER, we use a reward function that assigns $+1$ reward the first time the mouse reaches the corner. For CHEESE ON A DISH, we use a reward function that assigns $+1$ reward if the mouse collects the dish. For KEYS AND CHESTS, it's difficult to define a proxy goal (see Appendix F.3), here we report the average number of keys in the mouse's inventory throughout the rollouts (note that distinguishing levels have at most 3 reachable chests, and so carrying more than three keys suggests the agent is over-prioritizing key collection). The trends in proxy return mirror the trends in true return displayed in Figure 4, suggesting that our learned policies retain enough capabilities in distinguishing levels to pursue the proxy goal—a case of goal misgeneralization.

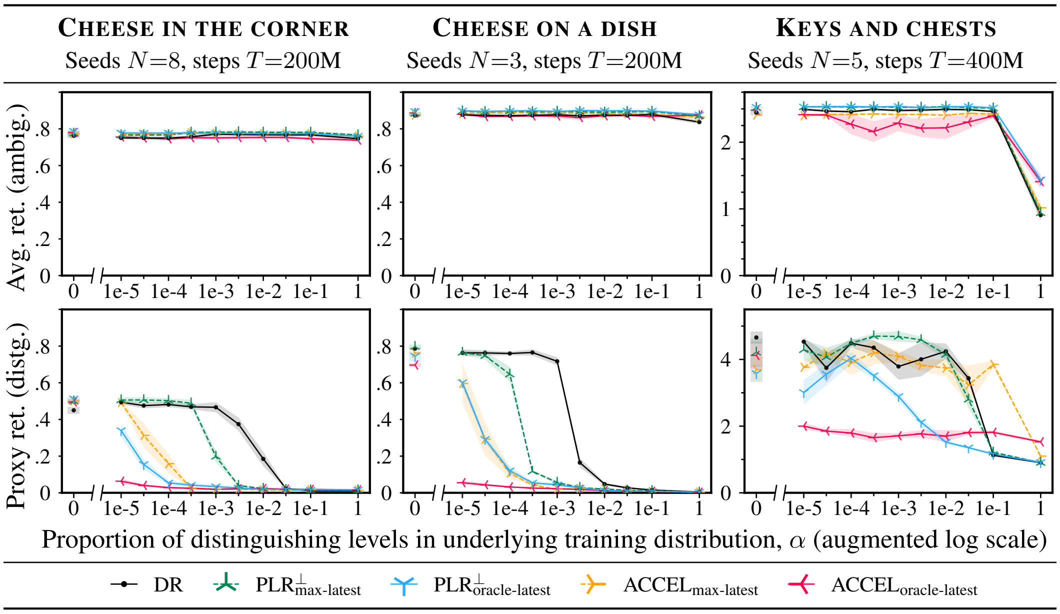

Figure H.1: **Performance on non-distinguishing levels and with respect to the proxy goal.** Each policy is trained on T environment steps using the indicated training method with underlying training distribution $\Lambda_{\alpha}^{\text{Train}}$. *($1^{st}$ row):* Average return over 512 steps for an evaluation batch of 256 non-distinguishing levels. *(2^{nd} row):* Average *proxy* return over 512 steps for an evaluation batch of 256 distinguishing levels. Note that the proxy goal is never used for training. *(Both):* Mean over N seeds, shaded region is one standard error. Note the split axes used to show zero on the log scale.

I Visualizing performance on levels with different cheese positions

For each training configuration (training distribution, training method) studied in the main text for CHEESE IN THE CORNER, we save the policy from the end of training for the first of 8 training seeds. In order to visualize the robustness of these policies to varying changes in the cheese position, we create a batch of 122 levels with a shared wall layout and a fixed mouse position, but where each level in the batch has a different cheese position. For each level, we sample 512 environment steps from each policy and compute the average per-episode return as a measure of the policy's performance in that level. This gives us a vector of 122 average return values for each policy (one for each level), which we visualize in a policy-specific heatmap such that the average return of the policy in a level with the cheese in position (i, j) is indicated by the color of the cell (i, j) in the heatmap. For context we overlay the wall layout and the mouse spawn position (note that we do not consider levels with cheese positions that would coincide with a wall or with the mouse spawn position).

Heatmaps for each method and training distribution follow in Figures I.1 and I.2. We see a rough progression whereby for more advanced algorithms or higher α, the agent is robust to a greater proportion of cheese positions. There are also instances of "blind spots" indicating cheese positions for which certain methods are not robust, indicating that these training methods do not produce perfectly robust policies.

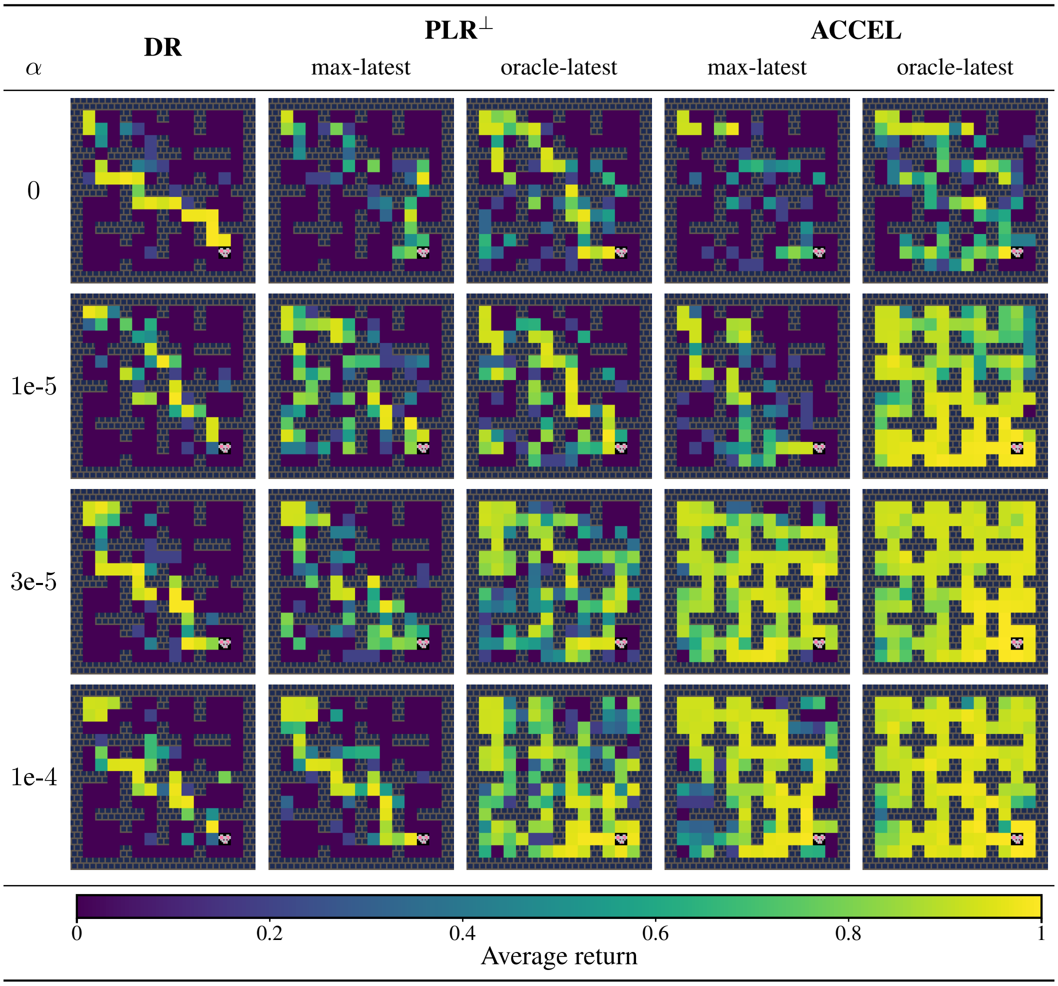

Figure I.1: Heatmap visualizations (part 1 of 2). See Figure 6 and Appendix I for details.

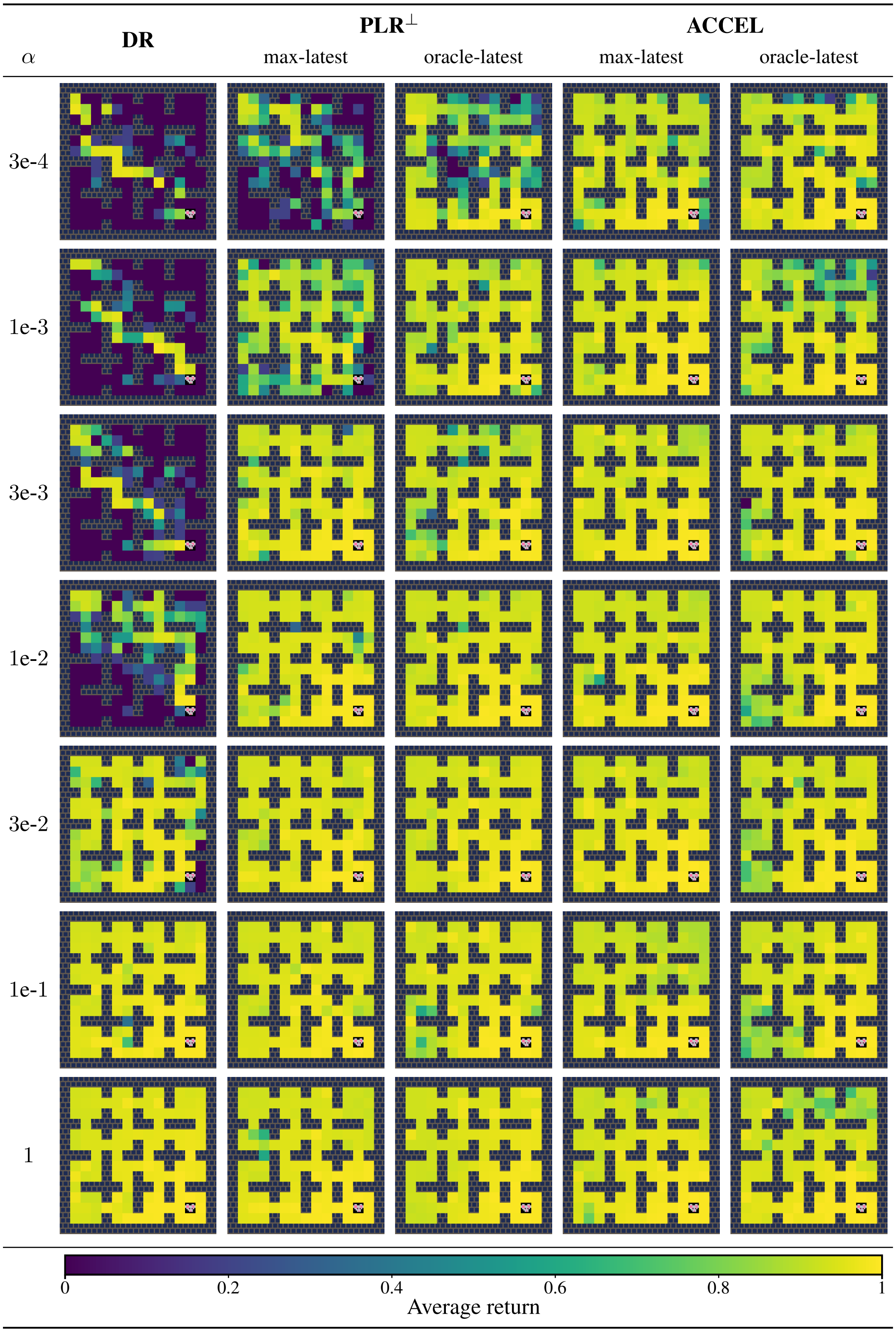

Figure I.2: Heatmap visualizations (part 2 of 2). See Figure 6 and Appendix I for details.

J Experiments with different edit distributions

As discussed in Section 6.2, ACCEL requires additionally specifying an **edit distribution** for mutating levels in the buffer. In this appendix, we explore the effect of different edit ditributions on the ability for ACCEL to mitigate goal misgeneralization.

J.1 Elementary edits

We assemble our edit distributions by sampling a sequence of **elementary edits** of the following three kinds.

- **Classification preserving edits.** These edits change the level without changing whether the level is non-distinguishing or distinguishing. For example, in CHEESE IN THE CORNER, such an edit may randomly toggle a wall or move the mouse's starting position, but would not change the location of the cheese.
- **Biased classification transforming edits.** These edits transform the level into a distinguishing level with probability α or an non-distinguishing level with probability $1 - \alpha$, where α is the proportion of distinguishing levels in the underlying training distribution. For example, in CHEESE IN THE CORNER, a biased classification transforming edit may randomize the cheese position with probability α or move it to the corner with probability $1 - \alpha$.
- **Unrestricted classification transforming edits.** These edits transformers the position of the cheese or the number of keys and chests uniformly at random given the level parameterization. For CHEESE IN THE CORNER and CHEESE ON A DISH, the cheese moves to a random position in the maze, which usually results in a distinguishing level. For KEYS AND CHESTS, we flip a coin to make either keys or chests sparse, and the other type of object dense.

We document the elementary edit distributions for each environment in full detail in Appendix F.

J.2 ACCEL variants

In these terms, we list the ACCEL variant considered in the main text along with three additional variants of ACCEL with different edit distributions. We fix a hyperparameter n, the number of elementary edits to apply to each level (we use $n = 12$).

1. **Identity ACCEL** (ACCEL$^{\text{identity}}$, simply "ACCEL" in main text)**.** We make a sequence of n random edits, all of which are classification preserving, resulting in the sequence of edits itself being classification preserving.
2. **Constant ACCEL** (ACCEL$^{\text{constant}}$)**.** We make $n - 1$ random classification preserving edits, followed by one biased classification transforming edit. Applying this operation to any distribution of levels results in a distribution with the same proportion of non-distinguishing and distinguishing levels as the underlying training distribution.
3. **Binomial ACCEL** (ACCEL$^{\text{binomial}}$)**.** We make a sequence of n random edits, each independently chosen to be either classification preserving (with probability $1 - 1/n$) or else biased classification transforming. If the sequence has only classification preserving edits (probability $(1 - 1/n)^n$) then it is classification preserving (like ACCEL$^{\text{identity}}$), otherwise the output is non-distinguishing with the same probability as a level sampled from the underlying training distribution (like ACCEL$^{\text{constant}}$).
4. **Unrestricted ACCEL** (ACCEL$^{\text{unrestricted}}$)**.** We make a sequence of $n - 1$ random classification preserving edits, followed by one unrestricted classification transforming edit. Applying this operation to any distribution of levels results in a distribution with a proportion of distinguishing levels that is independent of the parameter α that restricts access to distinguishing levels in the underlying training distribution.

ACCEL$^{\text{constant}}$ simulates restricted access to distinguishing levels. This variant is able to introduce new distinguishing levels through edit operations, however, its ability to replicate existing distinguishing levels is limited, since every time it edits a level, the mutated level reverts to an non-distinguishing level with probability $1 - \alpha$.

ACCEL$^{\text{identity}}$ (the variant studied in the main text) simulates a different kind of restriction, whereby we don't allow edits to turn non-distinguishing levels into distinguishing levels. However, we do allow edits to create new distinguishing levels by making similar copies of existing distinguishing levels in the buffer. Through this mechanism, ACCEL$^{\text{identity}}$ can rapidly populate the buffer with distinguishing levels, providing this variant with additional capacity to amplify the small training signal from distinguishing levels (beyond simply curating levels, like in PLR$^{\perp}$ or ACCEL$^{\text{constant}}$).

ACCEL$^{\text{binomial}}$ samples elementary edits in a different way. The number of biased classification transforming edits included in the sequence follows a binomial distribution. This means that with around 35% probability, no biased classification transforming edits will be applied, and the edit will resemble an edit from ACCEL$^{\text{identity}}$. Otherwise, with around 65% probability, at least one biased classification transforming edit will be applied, and the overall edit will resemble one from ACCEL$^{\text{constant}}$. We thus expect the performance of this variant to be somewhere between that of ACCEL$^{\text{constant}}$ and ACCEL$^{\text{identity}}$.

ACCEL$^{\text{unrestricted}}$ is a baseline that simulates a situation where the edit distribution can be designed to explore the space of levels completely independently of the training distribution. We expect this variant to be able to obtain much stronger performance comparable to using $\alpha = 1$ in the training distribution, even when training with $\alpha = 0$.

J.3 Experimental results

We train with the three new variants and compare performance to DR and ACCEL$^{\text{identity}}$ (from the main text). We report the results in Figure J.1 (oracle-latest regret estimator) and Figure J.2 (max-latest regret estimator). Note that we did not run ACCEL$^{\text{constant}}$ with the oracle-latest regret estimator for CHEESE ON A DISH, or ACCEL$^{\text{constant}}$ with the max-latest regret estimator for KEYS AND CHESTS.

Our results are in line with our central claim, that more advanced UED methods are more capable of mitigating goal misgeneralization.

- As predicted, ACCEL$^{\text{constant}}$ achieves lower performance than ACCEL$^{\text{identity}}$. This could be explained by the greater flexibility with which ACCEL$^{\text{identity}}$ can amplify distinguishing levels through edits.
- Moreover, ACCEL$^{\text{binomial}}$ achieves performance somewhere between that of ACCEL$^{\text{constant}}$ and ACCEL$^{\text{identity}}$.
- As predicted, ACCEL$^{\text{unrestricted}}$ is able to populate the buffer with distinguishing levels regardless of the training distribution, even when $\alpha = 0$, both with the oracle-latest regret estimator and with the max-latest regret estimator.

ACCEL$^{\text{binomial}}$ with max-latest estimation achieves the same low performance as ACCEL$^{\text{identity}}$ in KEYS AND CHESTS (as observed for ACCEL$^{\text{identity}}$ in Section 7).

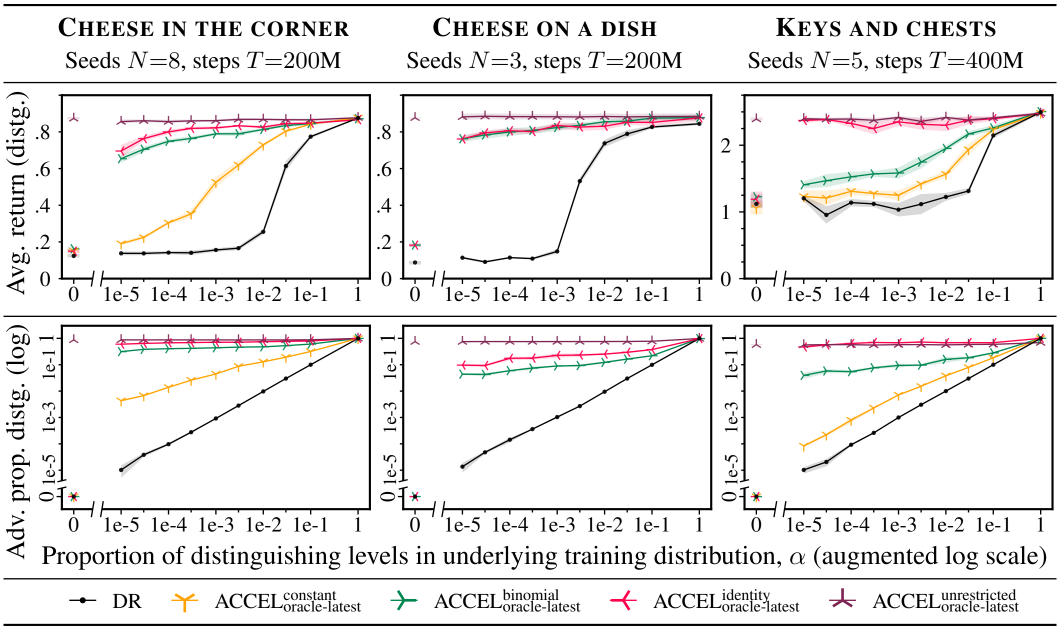

Figure J.1: ACCEL variants with oracle-latest estimator. Each policy is trained on T environment steps, using the indicated algorithm, with underlying training distribution $\Lambda^{\text{Train}}_{\alpha}$. *(1st row):* Average return over 512 steps for an evaluation batch of 256 distinguishing levels (cf. Figure 4). *(2nd row):* The proportion of distinguishing levels sampled from the adversary's buffer across training (cf. Figure 5). *(Both):* Mean over N seeds, shaded region is one standard error. Note the split axes used to show zero on the log scale.

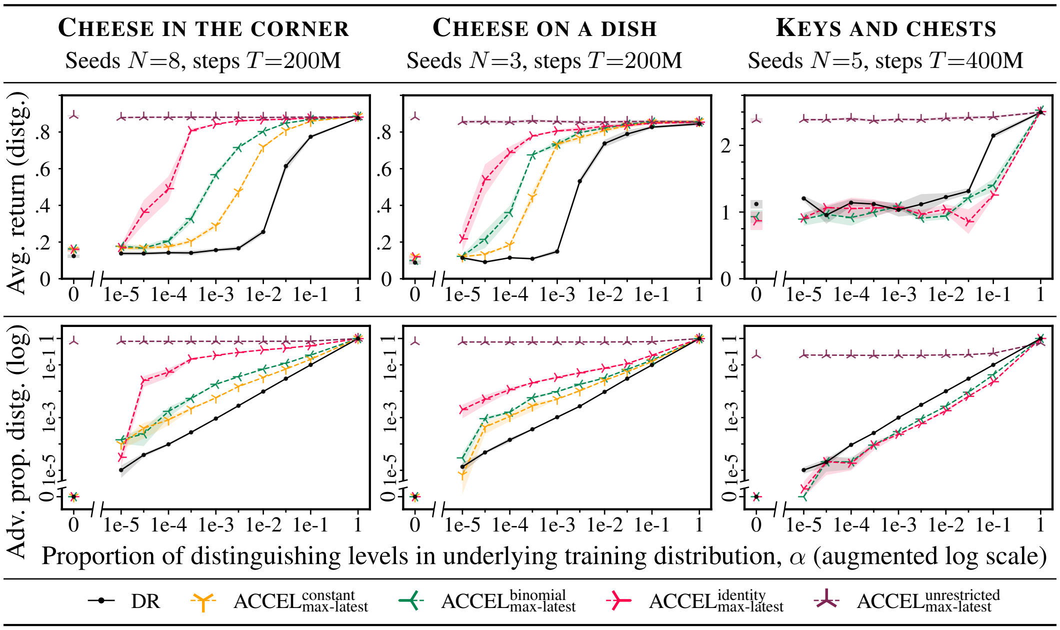

Figure J.2: ACCEL variants with max-latest estimator. Each policy is trained on T environment steps, using the indicated algorithm, with underlying training distribution $\Lambda^{\text{Train}}_{\alpha}$. *($1^{st}$ row):* Average return over 512 steps for an evaluation batch of 256 distinguishing levels (cf. Figure 4). *(2^{nd} row):* The proportion of distinguishing levels sampled from the adversary's buffer across training (cf. Figure 5). *(Both):* Mean over N seeds, shaded region is one standard error. Note the split axes used to show zero on the log scale.

K The maximin expected value objective is susceptible to goal misgeneralization

In this section, we discuss an alternative strategy to minimax expected regret for selecting robust policies. Namely, we consider the **maximin expected value** (MMEV) training objective, whereby one seeks a policy that achieves the highest possible expected value (return) on a worst-case level distribution $\Lambda^{\text{MMEV}} \in \arg\min_{\Lambda' \in \Delta(\Theta)} V^R(\pi; \Lambda')$.

Dennis et al. (2020) argues that the MMEV objective fails to induce robustness in an environment where the optimal value of each level varies within the level space. This is because the agent has no incentive to improve performance in any level above the maximum performance in worst-case levels. This same obstacle prevents MMEV from inducing robustness to goal misgeneralization, even though a policy that pursues a proxy goal in distinguishing levels will achieve low return in these levels.

In this appendix, we show theoretically that MMEV-based methods allow for goal misgeneralization in environments with levels with low maximum value. Moreover, we show empirically that MMEV-based training methods fail to mitigate goal misgeneralization in our environments. Indeed, they fail to produce policies that perform capably in any levels.

K.1 Theoretical results

We show our results for a perfect MMEV agent. It is easy to extend the result in the case where the agent is only ε-optimal—the performance on deployment levels could be suboptimal by a factor ε when such is allowed by the levels seen in deployment. The same argument holds for the adversary.

Definition 10 (MMEV policy)**.** *Consider an UMDP $\langle \Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$, a goal R. The* MMEV policy set *is then*

$$\Pi^{\text{MMEV}}(R) = \arg\max_{\pi \in \Pi} \min_{\Lambda \in \Delta(\Theta)} V^R(\pi; \Lambda).$$

Definition 11 (MMEV adversary)**.** *Consider an UMDP $\langle \Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$, a goal R, an optimal policy π^* in every level $\theta \in \Theta$. The* MMEV adversary strategy *is then*

$$\Lambda^{\text{MMEV}} \in \arg\min_{\Lambda' \in \Delta(\Theta)} V^R(\pi^\star; \Lambda')$$

Crucially, the MMEV adversary only cares about minimizing the return of the agent. Given this, the adversary will play levels in which *any* agent would achieve the minimum return possible. These are, for example, impossible levels if such exists in the level space.

Let's now define some additional machinery we will need for the proof:

Definition 12 (α-minimum achievable return)**.** *Given a level $\theta \in \Theta$ and a threshold α, define*

$$c(\theta, \alpha) = \min V_\alpha^R(\theta)$$

where $V_\alpha^R(\theta) = \{x \in \mathbb{R} \mid \exists \pi \in \Pi,$ such that $V^R(\pi; \theta) \geq \alpha$ and $V^R(\pi; \theta) = x\}$.

As mentioned before, the intuition is that an MMEV adversary will play those levels where the minimum return is achieved by any policy. Thus, any agent trained on those levels will mostly be able to reach that performance on any level in deployment. Thus, this results in suboptimal performance. This is clear to see in the case where impossible levels exist in the level space of the adversary. An agent trained on those would constitute an MMEV policy, although it basically consists of a policy not able to pursue the correct goal.

We note that c as just introduced, can be thought as a *return floor*. People familiar with the decision theory and UED literature can think of this as an (inverse) analog of the *regret floor* or *regret* *stagnation* Beukman et al. (2024). While the latter quantifies the minimum amount of regret any policy will suffer against a level distribution, we quantify the minimum amount of return a policy will incur.

Theorem 4 (MMEV is susceptible to goal misgeneralization)**.** *Consider an UMDP $\langle \Theta, \mathcal{A}, \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$, a pair of goals $R, \tilde{R}$, a proxy-distinguishing distribution shift $\langle \alpha, \beta, C, \Lambda^{\text{Train}}, \Lambda^{\text{Deploy}} \rangle$, and approximation threshold $\varepsilon \geq 0$. Then,*

$$\exists \pi^{\text{MMEV}} \in \Pi^{\text{MMEV}}(R) \textit{ such that } V^R(\pi^{\text{MMEV}}; \Lambda^{\text{Deploy}}) = c(\Lambda^{\text{Deploy}}, \alpha)$$

where $\alpha = V^R(\pi^*; \Lambda^{\text{MMEV}})$ *and* $c(\Lambda^{\text{Deploy}}, \alpha) = \mathbb{E}_{\theta \sim \Lambda^{\text{Deploy}}}[\, c(\theta, \alpha) \,]$

Proof. Consider $\pi^\star$ as the optimal policy across all levels $\theta \in \Theta$. The adversary will play a strategy

$$\Lambda^{\text{MMEV}} \in \underset{\Lambda' \in \Delta(\Theta)}{\arg\min}\; V^R(\pi^\star; \Lambda')$$

In words, the adversary will play levels in which the lowest possible score is achieved by the optimal policy.

We now need to construct our policy π^{MMEV}. Take the policy such that

$$V^R(\pi^{\text{MMEV}}; \theta) = \begin{cases} V^R(\pi^*; \theta), & \text{if } \theta \in \operatorname{supp} \Lambda^{\text{MMEV}}, \\ c(\theta, \alpha), & \text{if } \theta \notin \operatorname{supp} \Lambda^{\text{MMEV}}. \end{cases}$$

We note that we always have $c(\theta, \alpha) \geq V^R(\pi^*; \theta')$, where $\theta \in \Theta, \theta' \in \operatorname{supp} \Lambda^{\text{MMEV}}$ by our definitions. In words, take the MMEV policy such that it achieves the maximum return possible on levels played by the adversary, and the smallest available return on all other levels. Note that the smallest available return on levels not played by the adversary is necessarily greater than the highest one possibly achievable on levels played by the adversary. A policy that achieves this return is clearly in an MMEV policy. So, the following holds

$$V^R(\pi^{\text{MMEV}}; \Lambda') = c(\Lambda', \alpha).$$

Taking $\Lambda' = \Lambda^{\text{Deploy}}$ we get

$$V^R(\pi^{\text{MMEV}}; \Lambda^{\text{Deploy}}) = c(\Lambda^{\text{Deploy}}, \alpha).$$

as desired. □

In our theorem, we proved that an MMEV agent will possibly achieve returns that are at most the maximum ones achievable in levels played by the adversary. If very low return levels exists, this policy would then possibly goal misgeneralize at test time if evaluated on levels where a high return is possible.

K.2 Training methods and experimental results

In this section, we outline our empirical evaluation of MMEV-based training methods for mitigating goal misgeneralization. We adapt three UED methods: PLR$^\perp$ along with two ACCEL variants (ACCEL$^{\text{identity}}$ and ACCEL$^{\text{binomial}}$, see Appendix J). These UED methods were originally designed for MMER training, but we can convert their regret-maximizing adversaries into return-minimizing adversaries simply by replacing the regret estimator used to refine the buffer with a negative expected return estimator,

$$\hat{M}^R_{\text{latest}}(\pi; \theta) = -\hat{V}^R_{\text{latest}}(\pi; \theta), \tag{16}$$

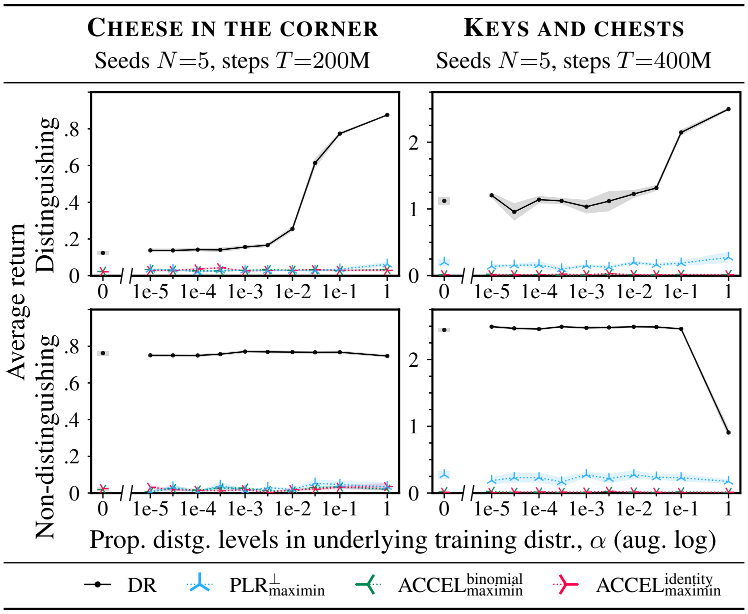

Figure K.1: Maximin training methods. Each policy is trained on T environment steps, using the indicated algorithm, with underlying training distribution $\Lambda^{\text{Train}}_{\alpha}$. Average return over 512 steps for an evaluation batch of 256 levels (top row: distinguishing levels, bottom row: non-distinguishing levels). Mean over N seeds, shaded region is one standard error. Note the split axes used to show zero on the log scale. Note the drop in performance for DR in non-distinguishing KEYS AND CHESTS levels at $\alpha = 1$ can be explained by noting that non-distinguishing levels are now out of distribution given this training distribution.

where $\hat{V}^{R}_{\text{latest}}(\pi;\theta)$ is the empirical average return achieved on θ in the latest batch of rollouts with the current policy π.

Figure K.1 shows the performance of trained policies in CHEESE IN THE CORNER and KEYS AND CHESTS for non-distinguishing and distinguishing levels. As expected, these training methods fail to mitigate goal misgeneralization, and even fail to induce robust performance in non-distinguishing levels.

We observe that these adversaries rapidly fill their buffers with levels with zero estimated value, indicating that the adversaries are working as expected. We hypothesize that the training failure is primarily due to the adversary finding levels in which the policy not only receives low expected value, but never receives any nonzero reward and thus obtains no training signal.

In the extreme case, the adversary could populate the buffer with levels that have zero *maximum* value, preventing any policy from obtaining nonzero reward. In CHEESE IN THE CORNER, levels in which the cheese position is unreachable from the mouse spawn position have zero maximum value. In Figure K.2, we plot the average proportion of such “unsolvable” levels in the adversary’s buffer over training for CHEESE IN THE CORNER. We find that this proportion is around 80% for most training distributions, which is much higher than the baseline value of around 18% of levels sampled from $\Lambda_{\neg\text{Distg.}}$ that are unsolvable. As the proportion of distinguishing levels increases, the average proportion of unsolvable levels decreases slightly, to around 50% for $\alpha = 0$. Note that the baseline value for $\Lambda_{\text{Distg.}}$ is also lower (around 7%) since it’s easier for walls to obstruct the cheese when it’s in the corner than when it is in an arbitrary position in the interior of the grid. Therefore we hypothesize that the adversaries have a slightly harder time finding unsolvable levels as α increases. However, unsolvable levels still occupy a majority of the buffer.

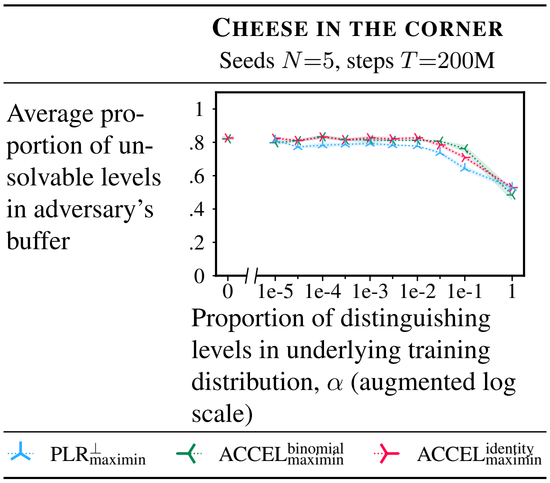

Figure K.2: Maximin training methods. Each policy is trained on T environment steps, using the indicated algorithm, with underlying training distribution $\Lambda^{\text{Train}}_{\alpha}$. Average proportion of unsolvable levels in the adversary's buffer over training. Mean over N seeds, shaded region is one standard error. Note the split axes used to show zero on the log scale.

L Experiments with increased training length

In most of our experiments, we compare the performance of algorithms after a fixed amount of training time, and find that MMER-based training methods typically outperform MEV-based training methods for a fixed training budget. We have shown that this is because a regret-maximizing adversary can amplify the proportion of distinguishing levels compared to sampling from a fixed underlying training distribution.

Another method of increasing the agent's experience in distinguishing levels is to train for longer. In this appendix, we extend training times in the CHEESE ON A DISH training environment (where DR was most robust to goal misgeneralization).

Figure L.1 shows the results. We find that training for more than 200 million environment steps with a fixed training method slightly mitigates goal misgeneralization. In particular, for $\alpha = $ 1e-3, DR gradually stops misgeneralizing after 200 million steps. The result is qualitatively consistent with Theorem 1, since increasing training time should have the effect of decreasing the optimization threshold.

However, for lower α values, further training shows diminishing returns, and even 1,500 million steps of DR training is insufficient to mitigate goal misgeneralization to the extent achieved by most UED methods within 200 million steps. This suggests that current UED methods are both more efficient and also qualitatively more effective at mitigating goal misgeneralization in this environment.

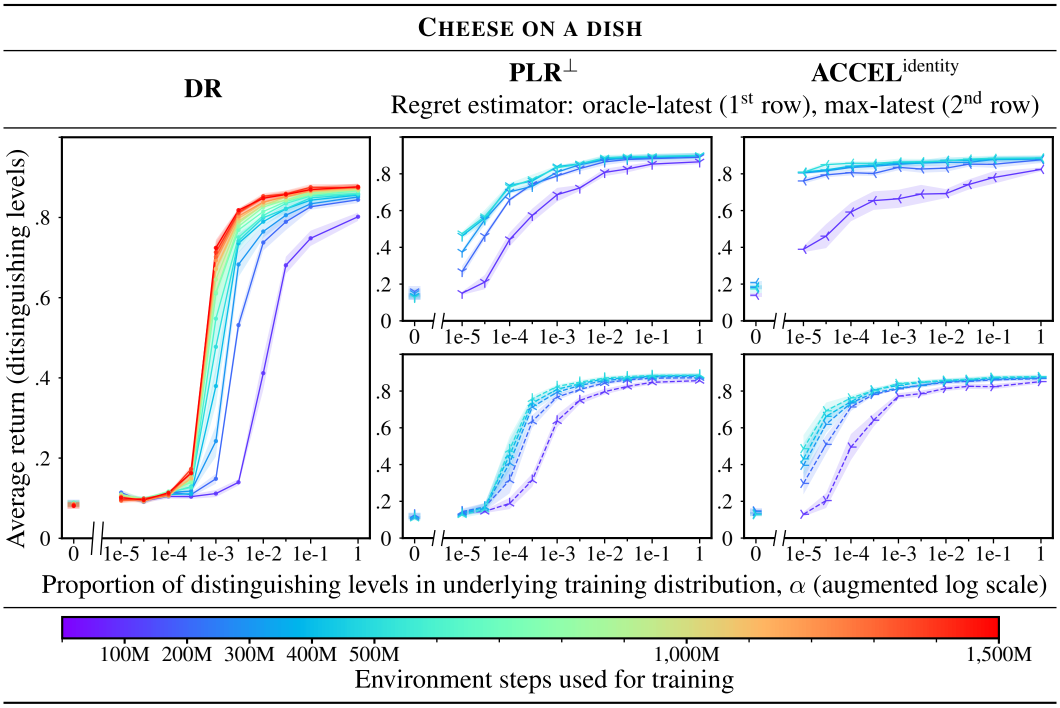

Figure L.1: Training for more environment steps in CHEESE ON A DISH. Each policy is trained for 500M environment steps (1,500M for DR), using the indicated algorithm, with underlying training distribution $\Lambda_\alpha^{\text{Train}}$. Every 100M steps, we evaluate the average return over 512 steps for an evaluation batch of 256 distinguishing levels (cf. Figure 4). Mean over 3 seeds, shaded region is one standard error. Note the split axes used to show zero on the log scale.

M Experiments with a different distinguishing level generator

In this section, we consider an alternative procedural level generator for distinguishing levels in the CHEESE IN THE CORNER environment. Our main experiments use a distinguishing level generator that places the cheese anywhere in the maze.

We consider a restricted distinguishing level generator that positions the cheese only within a region of size $c \times c$ surrounding the top-left corner, for varying c (the non-distinguishing generator would be recovered with $c = 1$, and the original, unrestricted distinguishing level generator would be recovered with $c = 13$).

Figure M.1 shows the return is evaluated on unrestricted distinguishing levels, where the cheese is positioned anywhere in the maze. We find that the UED methods are able to amplify the proportion of restricted distinguishing levels and in some cases mitigate goal misgeneralization. DR achieves low return across all corner sizes c.

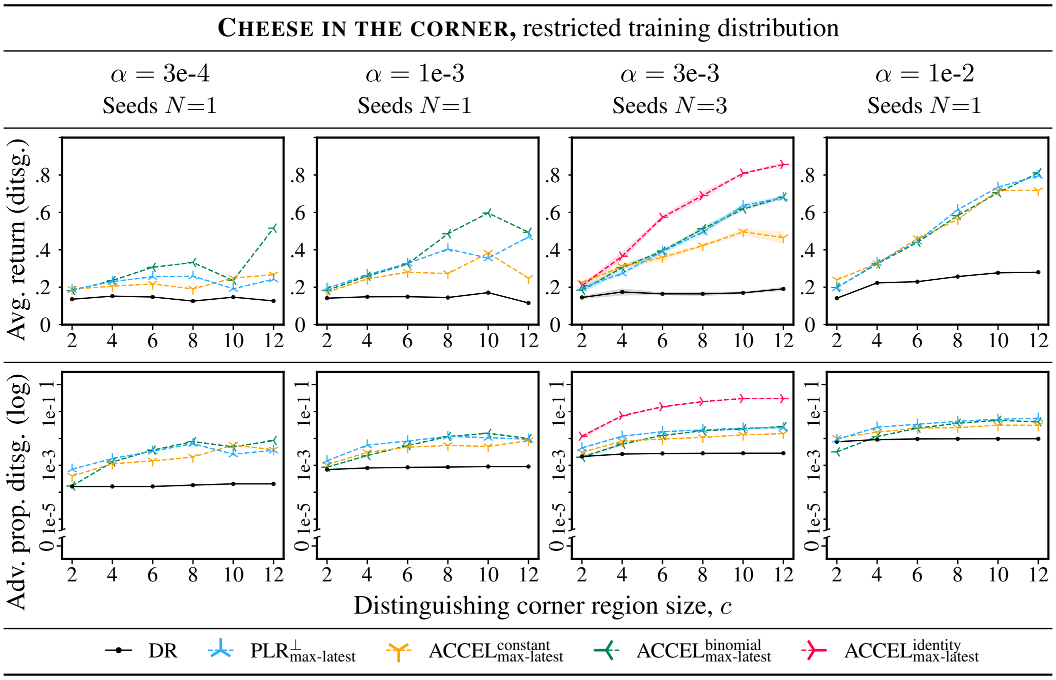

Figure M.1: **Training with varying distinguishing level generators.** Each policy is trained on 200 million environment steps, using the indicated algorithm, with underlying training distribution $\Lambda^{\text{Train}}_{\alpha,c} = (1-\alpha)\Lambda_{\neg\text{Distg.}} + \alpha\Lambda^{c}_{\text{Distg.}}$, where $\Lambda^{c}_{\text{Distg.}}$ is a procedural level generator that positions the cheese in the top-left $c \times c$ region of the maze. *(1st row):* Average return over 512 steps for an evaluation batch of 256 distinguishing levels (cf. Figure 4). *(2nd row):* The proportion of distinguishing levels sampled from the adversary's buffer across training (cf. Figure 5). *(Both):* Mean over N seeds, shaded region is one standard error. Note the split vertical axes used to show zero on the log scale.

N Experiments with different observations

In this section, we vary the way we encode observations for the policy in CHEESE ON A DISH and explore its effect on goal misgeneralization.

As discussed in Appendix F.2, the proxy goal and the true goal are symmetric in this environment, other than the fact that the position of the dish is redundantly represented across multiple channels in the Boolean grid observation. In the main text, we use $D = 6$ channels to encode the dish position (compared to 1 channel for the cheese position). The additional channels break a symmetry and create a slight inductive bias in favor of a policy that pursues the proxy goal.

We conduct an experiment where we vary the number of channels and see what affect it has on goal misgeneralization. Figure N.1 shows the results. We see that with one channel coding the dish position, all methods (including DR) are somewhat robust to goal misgeneralization, even at small α values. Additional channels significantly increase DR's susceptibility to goal misgeneralization. On the other hand, UED methods remain able to identify and amplify the training signal from rare distinguishing levels, while UED methods retain comparably similar performance.

The extent of amplification remains essentially constant with D. We hypothesize that the number of channels does not stop the adversary from noticing high-regret distinguishing levels, though it may affect how the policy responds.

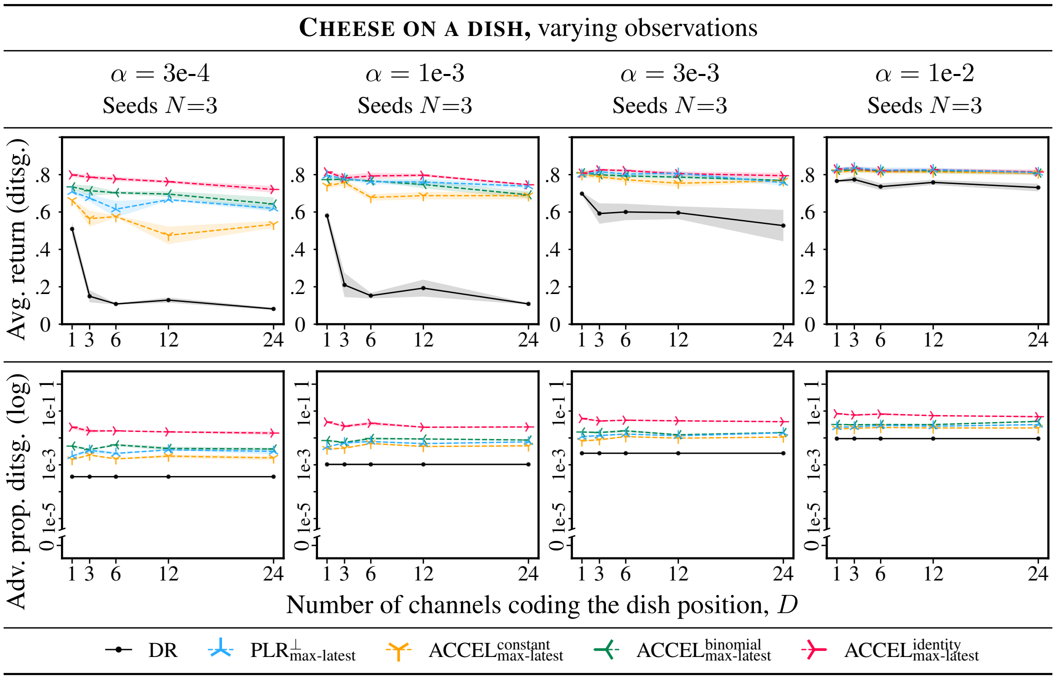

Figure N.1: **Training with observations with varying emphasis on the dish.** Each policy is trained on 200 million environment steps, using the indicated algorithm, with underlying training distribution $\Lambda_\alpha^{\text{Train}}$. We vary the number of channels (features) encoding the dish position, D. *(1st row):* Average return over 512 steps for an evaluation batch of 256 distinguishing levels (cf. Figure 4). *(2nd row):* The proportion of distinguishing levels sampled from the adversary's buffer across training (cf. Figure 5). *(Both):* Mean over N seeds, shaded region is one standard error. Note the split vertical axes used to show zero on the log scale.