STABILIZING UNSUPERVISED ENVIRONMENT DESIGN WITH A LEARNED ADVERSARY

Ishita Mediratta
Meta AI
ishitamed@meta.com

Minqi Jiang*
UCL, Meta AI

Jack Parker-Holder*
University of Oxford

Michael Dennis
University of California, Berkeley

Eugene Vinitsky
New York University

Tim Rocktaschel
UCL

ABSTRACT

A key challenge in training generally-capable agents is the design of training tasks that facilitate broad generalization and robustness to environment variations. This challenge motivates the problem setting of *Unsupervised Environment Design* (UED), whereby a student agent trains on an adaptive distribution of tasks proposed by a teacher agent. A pioneering approach for UED is PAIRED, which uses reinforcement learning (RL) to train a teacher policy to design tasks from scratch, making it possible to directly generate tasks that are adapted to the agent's current capabilities. Despite its strong theoretical backing, PAIRED suffers from a variety of challenges that hinder its practical performance. Thus, state-of-the-art methods currently rely on *curation* and *mutation* rather than *generation* of new tasks. In this work, we investigate several key shortcomings of PAIRED and propose solutions for each shortcoming. As a result, we make it possible for PAIRED to match or exceed state-of-the-art methods, producing robust agents in several established challenging procedurally-generated environments, including a partially-observed maze navigation task and a continuous-control car racing environment. We believe this work motivates a renewed emphasis on UED methods based on learned models that directly generate challenging environments, potentially unlocking more open-ended RL training and, as a result, more general agents.

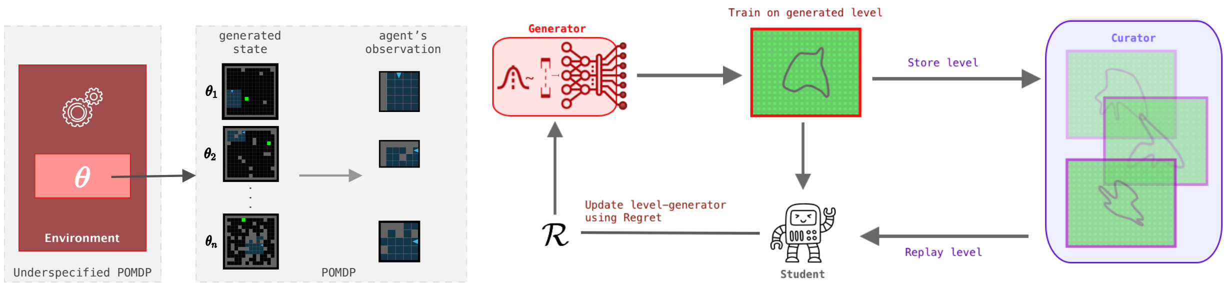
1 INTRODUCTION

Deep reinforcement learning (RL; Sutton & Barto, 1998) has been successfully applied to many challenging domains in recent years ranging from games (Silver et al., 2016; 2017; Vinyals et al., 2019; Berner et al., 2019; Hu et al., 2021) to real world problems such as controlling nuclear fusion plasma (Degrave et al., 2022). Many of these achievements are attributed to techniques like domain randomization and self-play, which provide an adaptive curriculum for training the agents. While these methods have led to impressive results, most successes are limited to single domains, necessitating re-training for each new setting. Instead, there has recently been interest in training more *generally capable* agents (Open Ended Learning Team et al., 2021; Adaptive Agent Team et al., 2023), which remains a considerable challenge (Zhang et al., 2018; Song et al., 2020).

As focus shifts from mastery to generality, emphasis shifts away from designing new agents to focus more on generating sufficiently rich environments. However, manually designing these environments is a tremendous engineering challenge, and even when possible, it is often the case that many instances of the environment are incompatible with training a robust generalist agent. Instead, it may be desirable to automatically discover useful training environments. In light of this, the *Unsupervised Environment Design* (UED; Dennis et al., 2020) paradigm has emerged, whereby a *student* agent trains on an adaptive distribution of tasks proposed by a *teacher* which ultimately aids in automatically discovering a curriculum of environments at the frontier of the agent capabilities.

A pioneering approach for UED is PAIRED, where the teacher is an RL agent trained to maximize *relative regret*, defined as the difference in performance between two student policies, called the protagonist (the student policy) and an antagonist (cooperating with the teacher agent). Using the teacher objective of maximizing regret between the antagonist and protagonist, Dennis et al. (2020) showed that the teacher proposes increasingly complex environments, which leads to a robust generalist student.

* : Equal Contribution



(a) From UPOMDP to POMDP

(b) UED Framework

Figure 1: **The Unsupervised Environment Design (UED) framework (Dennis et al., 2020)** In an Underspecified POMDP, the layout and dynamics of the environment are left underspecified as a free parameter θ. The level-generator (either learned or search-based) can then set this parameter to design POMDP levels. (b) This work focuses on the Unsupervised Environment Design (UED) paradigm, which is a method for designing environments using a curriculum, either via a learned adversary to generate new levels or via smart curation of previously-seen levels.

However, PAIRED suffers from a variety of challenges which hinder performance. The predominant issue is that optimizing the teacher with RL is incredibly challenging and suffers from the following problems:

- ***Non-Stationarity:*** Both student policies, the protagonist and the antagonist, are updating, making the problem *nonstationary*.
- ***Long-term Credit Assigment:*** There is a challenging *credit assignment problem* since the teacher must fully specify an environment before receiving a sparse reward in the form of feedback from the students.
- ***High-dimensionality:*** The *high-dimensionality of the space* can make the task difficult for the RL algorithms used by the teacher and students, leading to sub-optimal performance.

Due to the above-mentioned problems, PAIRED empirically tends to perform worse than one would expect it to based on the theoretical guarantees. In this paper, we will show and also try to alleviate the following two problems:

1. ***P1 - Entropy Collapse:*** The adversary (teacher) faces a challenging RL task due to the high-dimensionality of the environment design space, leading to difficulties in exploration and policy entropy collapse for the agents.
2. ***P2 - Fall Behind:*** In the PAIRED curriculum, protagonists can still face difficult exploration tasks, leading to sub-optimal performance. We demonstrate one such case in the Car Racing environment where the teacher produces simple levels favoring the antagonist, causing the protagonist to fall behind.

Thus, alternative UED algorithms focus on sampling tasks uniformly at random from the full distribution, relying on *curation* and *mutation* to provide an effective curriculum (see Section 2.3), as opposed to *generation* (guided by the regret metric, which aids in producing meaningful levels based on the agent's current capabilities) in PAIRED.

In this paper, we seek to investigate the major shortcomings in PAIRED, seeing if PAIRED can be made competitive with state-of-the-art methods. We propose a variety of techniques for stabilizing training by overcoming problems ***P1, P2***. In particular, we explore regularization approaches to aid the teacher in better exploration of the level-design space, alternative optimizers and finally behavior distillation across students. In each case we provide rigorous empirical analysis and demonstrate the specific mechanisms by which these approaches improve performance. As a result of these variations, we make it possible for PAIRED to match or exceed state-of-the-art methods, producing general agents in challenging procedurally generated environments. We believe this work paves the way for a renewed emphasis on methods that learn to automatically *generate* environments, potentially unlocking more open-ended RL training and as a result more robust, general agents. We have released the accompanying code as part of the existing DCD repository at https://github.com/facebookresearch/dcd.

2 Background

2.1 Reinforcement Learning

In reinforcement learning ((RL; Sutton & Barto, 1998), a Markov Decision Process (MDP) is a mathematical framework that models an agent's decision-making process. An MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where: $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{T}$ is the state transition probability function, denoted as $P(s', s|a)$, $\mathcal{R}$ is the reward function, denoted as $R(s, a)$, and γ is the discount factor, a scalar between 0 and 1, used to balance the trade-off between immediate and future rewards. The agent's objective in an MDP is to learn an optimal policy, denoted as π (which is a mapping from states to actions) that maximizes the expected cumulative discounted reward over time, defined as $\mathbb{E}[\sum_{t=0}^{\infty}\gamma^t R(s_t, \pi(s_t))]$. The value function, denoted as $V^\pi(s)$, is a measure of the long-term expected cumulative discounted reward for a given state s, when following a specific policy π. It is defined as: $V^\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty}\gamma^t R(s_t, \pi(s_t))|s_0 = s]$.

However, in many environments, the full context of a state is not known to the agent. Rather, the agent gets a partial-view of the state, also called as *observation*, which it then processes for decision-making. Such an MDP is termed as Partially-Observable Markov Decision Process (POMDP). In this case, the tuple is $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{I} : \mathcal{S} \rightarrow \Delta(\mathcal{O})$ is the set of observations that the agent receives. We refer the readers to Figure 1 (a) for a qualitative representation of POMDPs.

2.2 Unsupervised Environment Design

This work considers a more general problem setting where we seek to train an agent across an entire distribution of POMDPs. In this case there are set of design parameters θ which control the initial state, transition function, and reward function, e.g. the positions of obstacles in a maze. Given such parameters θ we can model the target domain as an Underspecified Partially-Observable Markov Decision Process (UPOMDP; Dennis et al., 2020). A UPOMDP is defined as a tuple $\mathcal{M} = (\theta, \mathcal{S}, \tilde{\mathcal{S}}_0 \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{T}, \mathcal{R}, \gamma)$, where θ as the set of free parameters of the environment, $\mathcal{S}$ is the set of states, $\tilde{\mathcal{S}}_0 : \theta \rightarrow \mathcal{S}$ is the initial distribution of states which can depend on the parameters θ, $\mathcal{A}$ as the set of actions, $\mathcal{O}$ as the set of observations, $\mathcal{I} : \mathcal{S} \rightarrow \Delta(\mathcal{O})$ is the observation function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \theta \rightarrow \Delta(S)$ is the transition function which can depend on the parameters θ, $\mathcal{R} : \mathcal{S} \times \theta \rightarrow \mathbb{R}$ is the reward function which can depend on the parameters θ, and γ is the discount factor. Here $\Delta(X)$ represents the set of distributions over the set X. Without setting the parameters θ, the environment, and thus the task, is underspecified and the UPOMDP can be thought of as a level editor. Once the parameters θ have been specified, the initial state distribution, reward function, and transition function are all fully defined, and the goal of the agent is to maximize the expected discounted sum of rewards as usual (Figure 1).

Unsupervised Environment Design (Dennis et al., 2020) considers the problem whereby a *teacher* (or adversary) must select levels from this distribution, such that the resulting *student* agent is capable of systematic generalization across all conceivable (solvable) levels. In order to do this, the teacher must maximize a utility function U. The most prominent recent approach to UED is to use a teacher that maximizes the concept of *regret*, defined as the difference between the expected return of the current policy π and the optimal policy π^*:

$$U_t^R(\pi, \theta) = \max_{\pi^* \in \Pi} \left(\textsc{Regret}^\theta(\pi^*, \pi)\right) \tag{1}$$

$$= \max_{\pi^* \in \Pi} \left(V^\theta(\pi^*) - V^\theta(\pi)\right) \tag{2}$$

Regret-based objectives are desirable as it can be argued, under certain set of assumptions (Theorem 1; Dennis et al., 2020), that they promote the creation of the simplest possible levels that the agent cannot currently solve. Moreover, if $S_t = \Pi$ is the set of strategies that the student agent can take and $S_t = \Theta$ is the set of strategies that the teacher can take, then if the learning process reaches a Nash equilibrium, the resulting agent policy π provably returns to a minimax regret policy (Theorem 2; Dennis et al., 2020), defined as:

$$\pi = \arg\min_{\pi_A \in \Pi} \max_{\theta, \pi_B \in \Theta, \Pi} \left(\textsc{Regret}^\theta(\pi_A, \pi_B)\right). \tag{3}$$

2.2.1 UED with a Learned Adversary

Protagonist Antagonist Induced Regret Environment Design (PAIRED; Dennis et al., 2020) is a method for generating an adaptive curriculum of levels by training the teacher agent π_θ to generate levels which maximize the *relative regret* between *two* student agents, referred to as the protagonist π_p and antagonist π_a. By designing levels on which the antagonist succeeds and the protagonist fails, the teacher (adversary) in PAIRED develops challenging levels while avoiding the failure case of generating overly difficult or unsolvable levels. Both the protagonist and antagonist agents are then trained on the generated levels resulting in an emergent curriculum of increasingly complex levels as the teacher adapts to find new challenges the protagonist cannot yet solve. The teacher agent is tasked with maximizing the relative regret between the two student agents using $\textsc{Regret}^{\vec{\theta}}(\pi^P, \pi^A) = V^{\vec{\theta}}(\pi^A) - V^{\vec{\theta}}(\pi^P)$. This approach allows for the endless generation of new levels, with the ultimate goal of continually challenging and improving the abilities of the two student agents.

The PAIRED algorithm (shown in Algorithm 1) is an iterative process, where in each iteration, the teacher (adversary) generates the parameters of the environment, $\vec{\theta} \sim \tilde{\Lambda}$, both student agents will play. The protagonist and antagonist agents then generate several trajectories within that same environment. The protagonist is trained to minimize regret, while the antagonist and the teacher (environment adversary) are trained to maximize regret. For more details on the level-generation process as well as the architectures of teacher and student agents, we refer readers to Section A.

Algorithm 1 PAIRED (Dennis et al., 2020)

1: **Input:** Initial policies for protagonist π^P, antagonist π^A, initial environment generator $\tilde{\Lambda}$
2: **while** not converged **do**
3: Use $\tilde{\Lambda}$ to generate environment parameters $\vec{\theta}$
4: Collect a trajectory τ^P using π^P in environment $\vec{\theta}$
5: Update π^P to minimize $\mathcal{L}_{ppo}(\pi_P)$
6: Collect a trajectory τ^A using π^A in environment $\vec{\theta}$
7: Update π^A to minimize $\mathcal{L}_{ppo}(\pi_A)$
8: Compute the regret as: $\textsc{Regret}^{\vec{\theta}}(\pi^P, \pi^A) = V^{\vec{\theta}}(\pi^A) - V^{\vec{\theta}}(\pi^P)$
9: Update $\tilde{\Lambda}$ to maximize regret
10: **end while**

2.3 UED WITH SEARCH-BASED CURATOR

Another approach to UED is Prioritized Level Replay (PLR; Jiang et al., 2021b;a). This method trains the agent on challenging levels found by curating a rolling buffer of the highest-regret levels surfaced through random search over possible level configurations. PLR approximates regret using the positive value loss, given by:

$$\frac{1}{T}\sum_{t=0}^{T}\max\left(\sum_{k=t}^{T}(\gamma\lambda)^{k-t}\delta_k, 0\right) \tag{4}$$

where λ and γ are the Generalized Advantage Estimation (GAE) and MDP discount factors respectively, and δ_t, the TD-error at timestep t. PLR has been shown to produce policies with strong generalization capabilities, but it is limited to only curating randomly sampled levels. More recently, ACCEL (Parker-Holder et al., 2022) built on PLR to include a mutation operator, making it possible to increase the regret of existing levels. This small modification allows for a rapid escalation in complexity, and results in strong zero-shot transfer generalization, including the challenging BipedalWalker environment.

Despite these recent innovations, both ACCEL and PLR rely on random samples for level-generation — and subsequently mutate them randomly. In our work we seek to improve upon PAIRED such that it is competitive with these newer approaches.

3 LIMITATIONS OF PAIRED

In theory, if the PAIRED algorithm reaches a Nash equilibrium, the protagonist policy has desirable properties like being at least as good as the antagonist policy on every level. However, in practical implementations, the teacher is trained using RL, which poses a series of challenges. Concretely, the teacher MDP consists of state s which is the current environment configuration and actions a that incrementally change it, until the end of the episode. The teacher receives a sparse reward r once both the student policies have been evaluated in the level.

This is a difficult RL problem—the teacher must grapple with challenging credit assignment from only getting reward after building a complete level, non-stationarity from updating students and finally a potentially high dimensional problem as there are often many parameters that need to be set in a given environment. For instance, the maze environment has more than 169 parameters which need to be set before the environment designer can receive any reward signal, presenting the environment designer with a high-dimensional exploration problem.

3.1 P1: ENTROPY COLLAPSE

Indeed, a typical instantiation of PPO struggles to properly explore the space, with the entropy of the policy often collapsing on an overly narrow set of possible level designs. In Section 4.1, we propose to circumvent this challenge by simply adding an entropy bonus which promotes exploration and also acts as a regularizer preventing premature convergence to suboptimal policies. Further, we also explore alternative optimizers in Section 4.2 since RL may not be best fit for this blackbox optimization problem.

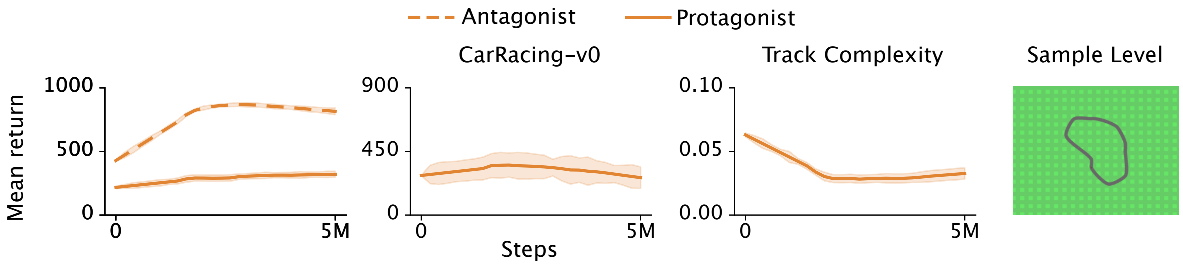


Figure 2: **PAIRED Breaks Down in CarRacing: Left:** Mean returns of protagonist and antagonist on the training levels generated by the adversary. **Center:** Performance of the protagonist in the CarRacing-v0 validation track as the training progresses. **Right:** Track complexity of the levels generated by the adversary throughout training. After a while, the training returns of the protagonist saturate, whereas the antagonist keeps on doing well even in very simple tracks (as track complexity decreases severely after 2M steps of training), thus leading to a never-ending cycle of degenerate levels.

3.2 P2: Fall Behind

As a result of this complexity, the system often converges to suboptimal solutions. For example, in the CarRacing UED environment proposed in Jiang et al. (2021a), θ corresponds to the coordinates for 12 control points defining a track. As we see in Figure 2, the adversary finds tracks that initially overly exploit the protagonist agent, which never subsequently recovers. Throughout training, the mean return of protagonist always stays lower than that of antagonist, thus leading to a vicious cycle where neither the tracks become complex nor the protagonist is able to learn any useful skills. After 5M steps of training, the protagonist still cannot solve simple round tracks, that are easy for the antagonist agent training with the same algorithm on the same data. This example also demonstrates the significance of getting the right curriculum, with agents unable to learn anything at all after seeing the wrong sequence of initial tasks.

This example failure mode of PAIRED can also offer a potential solution—in such cases the protagonist's exploration problem can be avoided by treating the antagonist, which is already solving the task by the PAIRED construction, as an *expert demonstrator*. In Section 4.3 we describe another possible improvement to PAIRED, adding a behavioral cloning loss to the protagonist so it may learn from the antagonist's solution, inspired by OpenAI et al. (2021) in the context of asymmetric self play (Sukhbaatar et al., 2017).

4 Revisiting Design Choices in PAIRED

In light of the limitations outlined in the previous section, we proposed several design modifications to improve PAIRED. These include the addition of a high entropy bonus (**HiEnt**), an investigation of the impact of using different optimizers for the level-generating network (**Evo**), and the implementation of an online behavioral cloning term (**BiBC, UniBC**). The effectiveness of these modifications will be discussed in the following sections.

Through our experiments, we show that our proposed changes produce more robust and transferable policies. In the following subsections, we will show how our method proposed modifications improves zero-shot generalization over RobustPLR in CarRacing, reaches a performance comparable to ACCEL in Minigrid, and is able to generate complex levels in BipedalWalker by changing the optimizer of the teacher.

4.1 Entropy Bonus

In the context of maze environments, empirically we observe during training that the entropy of the teacher (adversary) collapses before training is complete (see Figures 12 and 13). Therefore, to help alleviate that issue (Problem P1 in Section 1), we propose to add a high entropy bonus to the policy of the agents (and term the baseline as **PAIRED+HiEnt (HiEnt)**), which encourages both, the teacher and the students (protagonist and antagonist), to explore more robustly. Overall, adding such regularization presents a promising approach for addressing the exploration challenge for both, level-design and action-space.

4.2 Optimization Algorithm

It is common for Deep RL algorithms to utilize Proximal Policy Optimization (PPO; Schulman et al., 2017) as the objective. However, we hypothesize that the problems P1 and P2 mentioned in Section 3 are also affected by the choice of optimizer used inside the teacher agent. In Section 5, we examine the impact of using different optimizers, specifically a PPO-based teacher (learned editor from PAIRED) and a search-based editor (from ACCEL), on the performance of the level-design process. While a learned editor uses a neural network to design the best level based on the current capabilities of both the student agents, a search-based editor seeks to curate the best level from a pool of randomly generated levels based on a score metric (Jiang et al., 2021b;a; Parker-Holder et al., 2022), and hence the editor here has no learned weights. Our baseline, referred to as **PAIRED+Evo (Evo)**, is described in the accompanying algorithm 3 which is based on the algorithm from Parker-Holder et al. (2022) and we highlight our modifications in violet font. More specifically, we replace the teacher from PAIRED with a random level editor from ACCEL and use the approximate regret (calculated from the antagonist and the protagonist) as the scoring function for replaying or editing the next set of levels.

4.3 Protagonist-Antagonist Behavioral Cloning

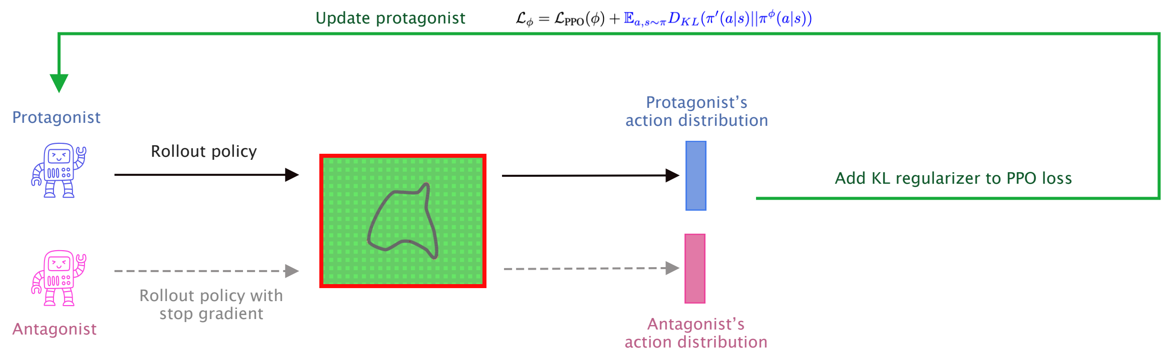


Figure 3: PAIRED+BC baseline in the context of the DCD framework (Jiang et al., 2021a). PAIRED+BC makes use of an additional regularization term which is added to the PPO loss of both (or either) of the student agents at regular intervals. For more details, refer to Section 4.3.

The population-based *Peer-to-Peer Online Distillation Strategy for RL* (P2PDRL; Zhao & Hospedales, 2021) was demonstrated to improve robustness in a randomized continuous control domain. Taking inspiration from P2PDRL, we propose **PAIRED+BC**, an online-policy distillation method that generalizes prior methods, whereby the protagonist agent optimizes the RL objective while simultaneously behaviorally cloning the antagonist agent (unidirectional behavioral cloning **UniBC**), and, if needed, the antagonist agent does the same to the protagonist (bidirectional behavioral cloning **BiBC**).

Policy distillation for policy-gradient RL commonly occurs through minimizing the KL divergence of the student's policy π from the teacher's policy π', $D_{KL}(\pi'||\pi)$. When performed online with PPO training, policy distillation for a student π with parameters ϕ thus adds an additional regularization term to the PPO objective, $J_{\text{PPO}}(\phi)$:

$$\mathcal{L}_\phi = \mathcal{L}_{\text{PPO}}(\phi) + \mathbb{E}_{a,s\sim\pi^\phi} D_{KL}(\pi'(a|s)||\pi^\phi(a|s)), \tag{5}$$

PPO performs minibatch SGD over multiple epochs of the collected transition data. To ensure each policy update is regularized by the policy distillation loss, we jointly optimize Equation 5 periodically in each minibatch update. Algorithm 2 provides a high-level overview of the PAIRED+BC algorithm.

It should be noted that our best results are achieved with our PAIRED+BC method when utilizing a high entropy bonus for both the students and teacher policies (i.e. UniBC+HiEnt or BiBC+HiEnt). While either of the two baselines, i.e. PAIRED+BC (UniBC or BiBC) without entropy or simply PAIRED+HiEnt, may yield improvements over the PAIRED method alone, they are not sufficient as standalone, optimal approaches.

5 Experimental Results

In this section, we examine the empirical performance of our proposed fixes on three challenging environments, CarRacing F1 (Brockman et al., 2016), Minigrid (Chevalier-Boisvert et al., 2018) mazes with 0-60 uniformly sampled blocks (Jiang et al., 2021a; Parker-Holder et al., 2022), and the BipedalWalker environment (Wang et al., 2019). We describe the technical details for each of these environments in Appendix A. For demonstrating the effectiveness of the proposed design choices in highly challenging environments, we also report results on the Minigrid Maze environment with a strict budget of 25-blocks. In this environment, the agents are thus trained on highly sparse grids (see Figure 15) and tested on the same mazes as in Minigrid 0-60 uniform environment, therefore increasing the difficulty of zero-shot transfer evaluation. All in all, we run the following baselines and ablations in each environment:

PAIRED The original PAIRED algorithm from an open source library[1]

HiEnt Using the PAIRED baseline from above, and adding high entropy bonuses to the teacher and students

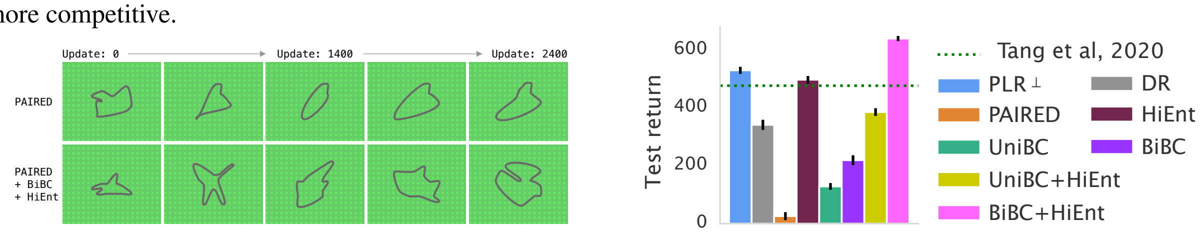
Uni/BiBC We run both Bidirection (**BiBC**) and Unidirectional (**UniBC**) to demonstrate the effectiveness of both approaches

Uni/BiBC+HiEnt To take things further, we combine the two design choices to see how much more we can benefit

PLR$^{\perp}$, ACCEL Additionally, we report the performance of existing SoTA approaches in each environment and also show how our suggested improvements help PAIRED achieve comparable performance with respect to replay-guided UED approaches.

5.1 CarRacing

First, we consider the CarRacing environment, previously described in Section 3. In Figure 4 we show the performance of all PAIRED variants as well as the DR and Robust PLR baselines in the held-out F1 benchmark. As we can see, adding an entropy bonus to the students and the teacher (adversary) is sufficient to avoid the sub-optimal local optimum we previously saw in Figure 2. Overall, the track complexity increases significantly, which can be seen in Figure 11 and Figure 4(a). Furthermore, PAIRED+BiBC+HiEnt stabilizes the open-ended learning process by ensuring that the protagonist is able to learn well from the curriculum (see Figure 11 and provides significant gains over PAIRED, even beating the current SoTA, Robust PLR (Jiang et al., 2021a) and a non-UED approach (Tang et al., 2020), by a clear margin in the CarRacing F1 benchmark. Even the BiBC, UniBC as well as UniBC+HiEnt variants are able to improve the performance of the protagonist over vanilla PAIRED, however, they are not sufficient in making the student agents more competitive.



(a) Tracks (b) Mean Performance

Figure 4: (a) F1 levels generated by the adversary as training progresses. (b) Mean aggregate results on real-world F1 tracks (see Figure 9) averaged over 10 seeds (error bars represent the standard deviation). When equipped with a Bidirectional BC term and a high entropy bonus, the protagonist trained in the CarRacing F1 environment (PAIRED+BiBC+HiEnt) is able to generalize better and the teacher agent produces more challenging tracks.

As well as improving transfer performance, we note the complexity of emergent problems generated by the algorithm drastically improved—tracks generated by the teacher become much more difficult, comprising of sharper edges and turns, compared to those under PAIRED, and start to resemble real-world tracks (see Figure 14 in Appendix B).

5.2 MiniGrid 25-Blocks

Next we consider the Minigrid environment which was first introduced in Chevalier-Boisvert et al. (2018). Readers may refer to Section A for a summary of this environment. In this experiment, we used the default hyperparameters from Jiang et al. (2021a) for PAIRED, which has an entropy coefficient of 0.0 for both, students and teacher. As shown in Figure 5, simply increasing the entropy bonus coefficient for both the students and the teacher (adversary) leads to significant improvement on the IQM metric (calculated using RLiable library (Agarwal et al., 2021)), and the teacher (adversary) continues

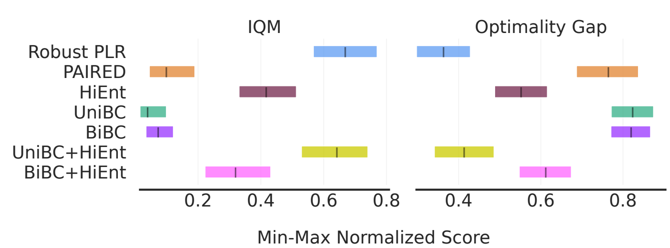


Figure 5: IQM and Optimality Gap for all baselines trained on the **Minigrid 25-Blocks** environment average across 5 seeds. PAIRED+UniBC+HiEnt is able to significantly improve the zero-shot transfer performance over PAIRED when evaluated on 12 test mazes (Table 1).

[1] https://github.com/facebookresearch/dcd

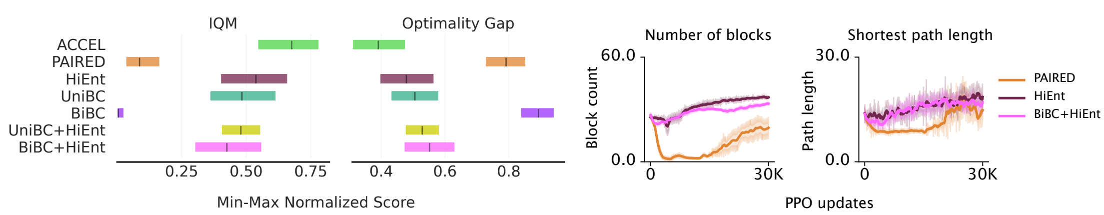
generating challenging mazes, as demonstrated in Figure 12. Since in this environment, there's a sharp drop in the PAIRED's adversary's entropy (see Figure 12), hence, similar to CarRacing, we found that increasing the entropy bonus helps in improving the performance (see UniBC+HiEnt and BiBC+HiEnt).

Table 1 shows the performance of the design choices in individual test mazes. We find that in this training environment (with a fixed budget of 25 blocks), the protagonist benefits only by unidirectional cloning (unlike bidirectional BC in the case of CarRacing). This maybe happening because there is very less difference between the training returns of the protagonist and antagonist (Figure 12) and both the agents are doing equally good in the sparse mazes, thereby eliminating the need for the antagonist to copy protagonist's actions. Overall, PAIRED+UniBC+HiEnt is able to achieve a performance on-par with RobustPLR, thus demonstrating the benefits of our proposed design modifications.

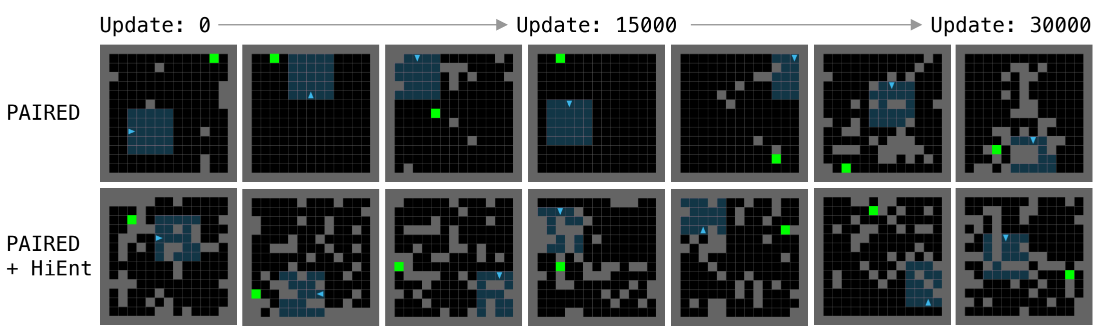
Table 1: Zero-shot test performance results for each baseline in 12 challenging minigrid mazes when trained with a budget of **25-blocks** only. Each baseline has been evaluated for 5 seeds and 100 episodes per seed. We report mean and standard deviation for Solved Rate as the metric here, highlighting in bold the best performing agents.

Environment	Robust PLR	PAIRED	HiEnt	UniBC	BiBC	UniBC+HiEnt	BiBC+HiEnt
Labyrinth	$\mathbf{0.66 \pm 0.16}$	0.27 ± 0.17	0.28 ± 0.18	0.0 ± 0.0	0.0 ± 0.0	$\mathbf{0.77 \pm 0.15}$	0.19 ± 0.13
Labyrinth2	$\mathbf{0.56 \pm 0.13}$	$\mathbf{0.32 \pm 0.2}$	0.24 ± 0.18	0.0 ± 0.0	0.0 ± 0.0	$\mathbf{0.41 \pm 0.18}$	0.21 ± 0.14
LargeCorridor	$\mathbf{0.71 \pm 0.16}$	0.26 ± 0.17	$\mathbf{0.68 \pm 0.18}$	0.38 ± 0.22	0.48 ± 0.17	$\mathbf{0.59 \pm 0.19}$	$\mathbf{0.79 \pm 0.14}$
Maze	$\mathbf{0.4 \pm 0.18}$	0.0 ± 0.0	0.11 ± 0.09	0.0 ± 0.0	0.0 ± 0.0	$\mathbf{0.26 \pm 0.19}$	0.07 ± 0.07
Maze2	$\mathbf{0.65 \pm 0.16}$	0.0 ± 0.0	0.11 ± 0.08	0.0 ± 0.0	0.0 ± 0.0	$\mathbf{0.6 \pm 0.15}$	0.33 ± 0.14
Maze3	$\mathbf{0.67 \pm 0.16}$	0.2 ± 0.2	$\mathbf{0.69 \pm 0.16}$	0.0 ± 0.0	0.03 ± 0.03	$\mathbf{0.58 \pm 0.15}$	$\mathbf{0.38 \pm 0.23}$
MiniGrid-FourRooms	$\mathbf{0.51 \pm 0.04}$	0.33 ± 0.04	$\mathbf{0.45 \pm 0.02}$	0.42 ± 0.04	0.35 ± 0.02	$\mathbf{0.53 \pm 0.06}$	0.42 ± 0.02
MiniGrid-SimpleCrossingS11N5	$\mathbf{0.91 \pm 0.04}$	0.3 ± 0.14	0.73 ± 0.04	0.36 ± 0.08	0.37 ± 0.1	0.74 ± 0.1	$\mathbf{0.76 \pm 0.12}$
PerfectMazeMedium	$\mathbf{0.44 \pm 0.12}$	0.17 ± 0.09	$\mathbf{0.31 \pm 0.04}$	0.04 ± 0.02	0.09 ± 0.03	$\mathbf{0.42 \pm 0.05}$	0.24 ± 0.06
SixteenRooms	$\mathbf{0.87 \pm 0.05}$	0.46 ± 0.19	$\mathbf{0.68 \pm 0.14}$	0.31 ± 0.18	0.19 ± 0.16	$\mathbf{0.79 \pm 0.09}$	0.14 ± 0.08
SixteenRoomsFewerDoors	0.48 ± 0.17	0.18 ± 0.16	0.4 ± 0.05	0.1 ± 0.09	0.18 ± 0.12	$\mathbf{0.75 \pm 0.09}$	0.27 ± 0.17
SmallCorridor	$\mathbf{0.78 \pm 0.09}$	0.35 ± 0.19	$\mathbf{0.67 \pm 0.17}$	0.51 ± 0.18	0.48 ± 0.17	$\mathbf{0.59 \pm 0.21}$	$\mathbf{0.86 \pm 0.08}$
Mean	$\mathbf{0.64 \pm 0.03}$	0.24 ± 0.09	0.45 ± 0.03	0.18 ± 0.03	0.18 ± 0.05	$\mathbf{0.59 \pm 0.06}$	0.39 ± 0.05

5.3 MiniGrid 0-60 Uniform-Blocks



(a) IQM (b) Maze Complexity metrics

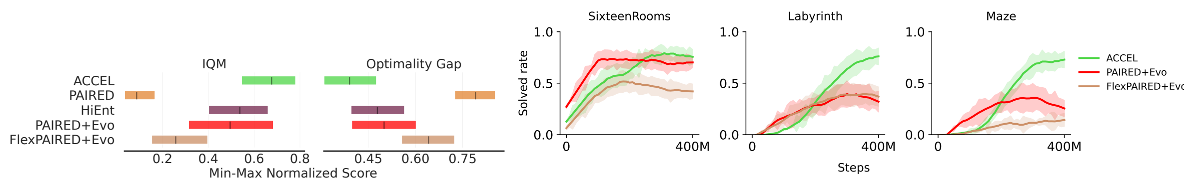


(c) Tracks

Figure 6: **Minigrid 0-60 Uniform-Blocks environment.** (a) IQM and Optimality Gap for all baselines when evaluated on test mazes (Table 4) averaged over 5 seeds. (b) Maze complexity metrics showing the emergent complexity during training in HiEnt and BiBC+HiEnt methods. (c) Sample tracks generated by the teacher agent (adversary) in PAIRED and HiEnt methods. Due to entropy collapse (Problem P1), PAIRED's adversary generates sparse mazes leading to poor zero-shot transfer at test time, and adding a high entropy bonus alleviates that issue.

Figure 6 (a) shows the performance of the HiEnt baseline versus the original PAIRED method on the Minigrid 0-60 Uniform Blocks benchmark. The HiEnt baseline consists of 0.05 entropy coefficient for the level-generating teacher (adversary), and a 0.005 bonus for the students. Moreover, Figure 6 (b) shows the number of blocks per level and the minimum path length to reach the goal (with a length of 0 assigned to unsolvable levels). These figures illustrate that the grids generated by PAIRED's adversary kept on becoming sparse as training progressed, thus leading to a degraded performance. Both HiEnt and BiBC+HiEnt baselines improve on the complexity front and we found that the performance of the HiEnt baseline can significantly reduce the gap with ACCEL, often matching performance on highly challenging test mazes (see Table 4). Readers may note that we train all PAIRED baselines for 30k gradient updates (250M environment steps), whereas we train ACCEL for 20k gradient updates only (which corresponds to 400M environment steps). However, in Minigrid 0-60 (unlike Minigrid 25-Blocks benchmark), the level-generating process doesn't benefit the most from behavioral cloning, as the HiEnt baseline achieves the highest gains instead of BiBC+HiEnt or UniBC+HiEnt. For a comparison of all these methods on individual test environments, readers may refer to Table 4.

5.4 Changing the optimizer in BipedalWalker and Minigrid



(a) Test IQM

(b) Validation Solved Rates

Figure 7: IQM, Optimality Gap and Solved Rates averaged across 5 seeds for PAIRED+Evo and FlexPAIRED+Evo baselines on Minigrid 0-60 Uniform-blocks environment.

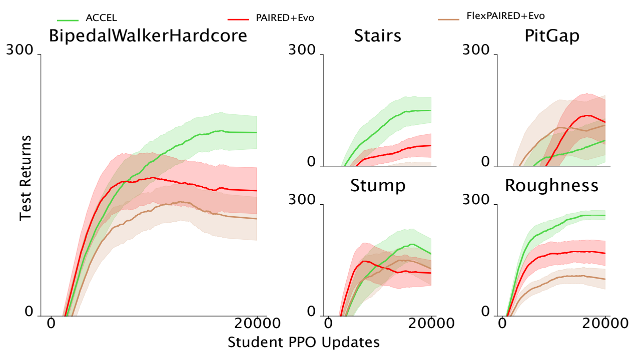


Figure 8: Zero-shot transfer performance of PAIRED+Evo on 5 test environments as training progresses in the BipedalWalker environment.

In this section, we study the effect of changing the optimizer from a learned teacher to a random search-based level-designer, naming the baseline to be PAIRED+Evo. Additionally, we also report the performance of this baseline when using a *flexible* regret objective (Dennis et al., 2020; Gur et al., 2021) where there is no fixed antagonist or protagonist. We call this baseline Flex-PAIRED+Evo.

As depicted in Figure 7(a), both PAIRED+Evo and FlexPAIRED+Evo underperform compared to PAIRED+HiEnt in the Minigrid 0-60 Uniform Blocks environment, suggesting that a high entropy bonus has a greater impact than the choice of optimizer. Moreover, in Figure 7(b), it can be seen that the ACCEL tends to do a lot better on the three validation mazes, indicating that the positive value loss leads to better curation of training levels compared to the approximate regret calculated by a pair of student agents in PAIRED(or Flexible PAIRED)+Evo. Similar to Minigrid, in the challenging BipedalWalker (Wang et al., 2019) environment, PAIRED+Evo performs worse than ACCEL, but better than PAIRED (Figure 8), indicating that the choice of optimizer is a significant design factor and warrants further investigation, especially in more challenging environments where the level-design space is even bigger.

It would be interesting to see how PAIRED performs when combined with all three design choices, i.e. PAIRED+Evo+BC+HiEnt, which we leave for future work.

6 Related Work

Our work extends a popular line of research in regret-based Unsupervised Environment Design (UED), which aims to address problems of generalization and robustness in RL.

Unsupervised Environment Design Unsupervised Environment Design (UED), which automatically designs training environments to maximize learning potential is a quickly growing field (Wang et al., 2019; Dharna et al., 2020; Wang et al., 2020; Dennis et al., 2020; Open Ended Learning Team et al., 2021; Jiang et al., 2021a; Parker-Holder et al., 2022; Jiang et al., 2022; Dharna et al., 2022; Team et al., 2023). A significant line of this work is targeted at designing high-regret environments (Dennis et al., 2020; Gur et al., 2021; Jiang et al., 2021a; Parker-Holder et al., 2022; Adaptive Agent Team et al., 2023). Many of the most recent successful techniques have used curation or evolution to build individual levels (Jiang et al., 2021a; Parker-Holder et al., 2022; Team et al., 2023). While these approaches are currently state of the art they build each level one by one. Thus a promising direction UED is for neural models to generate environment parameters, thus allowing for combinatorial generalization across levels.

PAIRED (Dennis et al., 2020) and the algorithms which are built on it takes this neural generative approach (Gur et al., 2021; Du et al., 2022; Jiang et al., 2021a; Wang et al., 2022). Thus our work gives guidance to many approaches on how they may be tuned to be competitive with the state of the art. Unsupervised environment design is also very closely related to work on AI for procedural environment generation. AI for procedural environment generation is a long-standing (Togelius et al., 2011; Browne & Maire, 2010; Togelius & Schmidhuber, 2008) and active field (Earle et al., 2021; Khalifa et al., 2022; Bhatt et al., 2022). Though our work focuses on the case of using the resulting environments to train an agent and thus does not directly address these approaches, cross-pollinating the ideas of these fields is an important line of future work.

Generalization in RL In addition to providing a curriculum, regret-based UED approaches improve robustness and generalization in RL (Kirk et al., 2021). Deep RL systems, like the deep neural networks on which they are based Szegedy et al. (2013), have been shown to exhibit robustness failures such as failing under adversarial attacks Kos & Song (2017); Lin et al. (2017); Gleave et al. (2019) or overfitting to the environment configuration Witty et al. (2021); Di Langosco et al. (2022). Many approaches have aimed to remedy these failures by changing the objective of RL systems, aiming to solve something closer to a robust MDP (Bagnell et al., 2001; Iyengar, 2005; Nilim & El Ghaoui, 2005). There have been a wide variety of such approaches (Garcıa & Fernández, 2015), including those which aim to solve regret-based MDP formulations (Ghavamzadeh et al., 2016; Regan & Boutilier, 2011; 2012).

A powerful and scalable approach for solving robust MDPs is adversarial training, for which there exist many specialized methods (Pinto et al., 2017b;a; Morimoto & Doya, 2005). The regret-based UED algorithms we study in this work can be seen as a high dimensional adversarial training approach to a particular sort of solve regret-based MDP.

7 Conclusion

This work considers problems arising when using a learned adversary (or teacher) for unsupervised environment design (UED). Focusing on the popular PAIRED algorithm, we discussed potential instabilities during training and explored several approaches for mitigating them. Our results indicate that properly tuning the entropy parameter is crucial for achieving good performance—by doing so, we find that this method is able to achieve results that are competitive with other state-of-the-art UED methods. We also consider the use of behavioral cloning, which helps the student agent in learning usefull skills in the simplest of levels. However, in some environments where this is not an issue, behavioral cloning can lead to reduced out-of-distribution (OOD) robustness in the student model. Moreover, we find that, when combined with the proper entropy tuning, the choice of optimizer only makes a marginal difference in the performance of the PAIRED method. In general, our results make it possible to achieve state-of-the-art results with a learned adversary, propelling this class of methods back to the forefront of UED research. We believe this should result in a variety of future innovation, such as combining our proposed solutions with population-based training.

Acknowledgments

We would like to thank Roberta Raileanu, Mikayel Samvelyan, Eric Hambro, and Jakob Foerster for providing valuable feedback and engaging in insightful discussions about our work. We are also grateful to the anonymous reviewers for their constructive suggestions. This work was funded by Meta AI.

References

Adaptive Agent Team, Jakob Bauer, Kate Baumli, Satinder Baveja, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, Vibhavari Dasagi, Lucy Gonzalez, Karol Gregor, Edward Hughes, Sheleem Kashem, Maria Loks-Thompson, Hannah Openshaw, Jack Parker-Holder, Shreya Pathak, Nicolas Perez-Nieves, Nemanja Rakicevic, Tim Rocktäschel, Yannick Schroecker, Jakub Sygnowski, Karl Tuyls, Sarah York, Alexander Zacherl, and Lei Zhang. Human-timescale adaptation in an open-ended task space, 2023. URL https://arxiv.org/abs/2301.07608.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*. 2021.

J Andrew Bagnell, Andrew Y Ng, and Jeff G Schneider. Solving uncertain markov decision processes. 2001.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019.

Varun Bhatt, Bryon Tjanaka, Matthew C Fontaine, and Stefanos Nikolaidis. Deep surrogate assisted generation of environments. *arXiv preprint arXiv:2206.04199*, 2022.

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

Cameron Browne and Frederic Maire. Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):1–16, 2010.

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

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

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

Aaron Dharna, Julian Togelius, and L. B. Soros. Co-generation of game levels and game-playing agents. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 16(1):203–209, Oct. 2020.

Aaron Dharna, Amy K Hoover, Julian Togelius, and Lisa Soros. Transfer dynamics in emergent evolutionary curricula. *IEEE Transactions on Games*, 2022.

Lauro Langosco Di Langosco, Jack Koch, Lee D Sharkey, Jacob Pfau, and David Krueger. Goal misgeneralization in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 12004–12019. PMLR, 2022.

Yuqing Du, Pieter Abbeel, and Aditya Grover. It takes four to tango: Multiagent selfplay for automatic curriculum generation. *arXiv preprint arXiv:2202.10608*, 2022.

Sam Earle, Justin Snider, Matthew C. Fontaine, Stefanos Nikolaidis, and Julian Togelius. Illuminating diverse neural cellular automata for level generation, 2021.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Mohammad Ghavamzadeh, Marek Petrik, and Yinlam Chow. Safe policy improvement by minimizing robust baseline regret. *Advances in Neural Information Processing Systems*, 29, 2016.

Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

Izzeddin Gur, Natasha Jaques, Yingjie Miao, Jongwook Choi, Manoj Tiwari, Honglak Lee, and Aleksandra Faust. Environment generation for zero-shot compositional reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Hengyuan Hu, Adam Lerer, Brandon Cui, Luis Pineda, Noam Brown, and Jakob Foerster. Off-belief learning. In *International Conference on Machine Learning*, pp. 4369–4379. PMLR, 2021.

Garud N Iyengar. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280, 2005.

Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. In *Advances in Neural Information Processing Systems*. 2021a.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *The International Conference on Machine Learning*. 2021b.

Minqi Jiang, Michael Dennis, Jack Parker-Holder, Andrei Lupu, Heinrich Küttler, Edward Grefenstette, Tim Rocktäschel, and Jakob Foerster. Grounding aleatoric uncertainty in unsupervised environment design. *arXiv preprint arXiv:2207.05219*, 2022.

Ahmed Khalifa, Julian Togelius, and Michael Cerny Green. Mutation models: Learning to generate levels by imitating evolution. In *Proceedings of the 17th International Conference on the Foundations of Digital Games*, pp. 1–9, 2022.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of generalisation in deep reinforcement learning. *CoRR*, abs/2111.09794, 2021.

Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017.

Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*, 2017.

Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural computation*, 17(2):335–359, 2005.

Arnab Nilim and Laurent El Ghaoui. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michaël Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki. Open-ended learning leads to generally capable agents. *CoRR*, abs/2107.12808, 2021.

OpenAI OpenAI, Matthias Plappert, Raul Sampedro, Tao Xu, Ilge Akkaya, Vineet Kosaraju, Peter Welinder, Ruben D'Sa, Arthur Petron, Henrique Ponde de Oliveira Pinto, Alex Paino, Hyeonwoo Noh, Lilian Weng, Qiming Yuan, Casey Chu, and Wojciech Zaremba. Asymmetric self-play for automatic goal discovery in robotic manipulation, 2021.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. *arXiv preprint arXiv:2203.01302*, 2022.

Lerrel Pinto, James Davidson, and Abhinav Gupta. Supervision via competition: Robot adversaries for learning tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1601–1608, 2017a. doi: 10.1109/ICRA.2017.7989190.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pp. 2817–2826. PMLR, 2017b.

Kevin Regan and Craig Boutilier. Robust online optimization of reward-uncertain mdps. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, pp. 2165, 2011.

Kevin Regan and Craig Boutilier. Regret-based reward elicitation for markov decision processes. *arXiv preprint arXiv:1205.2619*, 2012.

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

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529: 484–489, 2016.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. Observational overfitting in reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Yujin Tang, Duong Nguyen, and David Ha. Neuroevolution of self-interpretable agents. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pp. 414–424, 2020.

Adaptive Agent Team, Jakob Bauer, Kate Baumli, Satinder Baveja, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, et al. Human-timescale adaptation in an open-ended task space. *arXiv preprint arXiv:2301.07608*, 2023.

Julian Togelius and Jurgen Schmidhuber. An experiment in automatic game design. In *2008 IEEE Symposium On Computational Intelligence and Games*, pp. 111–118, 2008. doi: 10.1109/CIG.2008.5035629.

Julian Togelius, Georgios N Yannakakis, Kenneth O Stanley, and Cameron Browne. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3): 172–186, 2011.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Çaglar Gülçehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nat.*, 575(7782):350–354, 2019. doi: 10.1038/s41586-019-1724-z.

Rose E Wang, Jesse Mu, Dilip Arumugam, Natasha Jaques, and Noah Goodman. In the zone: Measuring difficulty and progression in curriculum generation. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. Paired open-ended trailblazer (POET): endlessly generating increasingly complex and diverse learning environments and their solutions. *CoRR*, abs/1901.01753, 2019.

Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeffrey Clune, and Kenneth Stanley. Enhanced POET: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9940–9951. PMLR, 13–18 Jul 2020.

Sam Witty, Jun K Lee, Emma Tosch, Akanksha Atrey, Kaleigh Clary, Michael L Littman, and David Jensen. Measuring and characterizing generalization in deep reinforcement learning. *Applied AI Letters*, 2(4):e45, 2021.

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

Chenyang Zhao and Timothy Hospedales. Robust domain randomised reinforcement learning through peer-to-peer distillation. In *Asian Conference on Machine Learning*, pp. 1237–1252. PMLR, 2021.

A EXPERIMENTAL DETAILS

In this section, we summarize the technical details of each of the environment used and list out the hyperparameters used for each of the baselines in Table 2.

A.1 F1 CARRACING

Environment We use the CarRacing F1 UED benchmark introduced in Jiang et al. (2021a). The environment used is a reparameterized version of the CarRacing game Brockman et al. (2016), where the tracks consist of closed loops made of Bézier curves created from 12 randomly sampled control points. The student agent is rewarded for driving over each unvisited polygon and penalized for each time step. The student agent's observation space consists of a 96x96x3 pixel RGB image with a bird's eye view of the vehicle, which allows the agent to make decisions based on the current state of the environment and the action space is a 3-dimensional continuous action, corresponding to control values for steer, gas and brake. The adversary (teacher agent) generates a sequence of 12 control points, one per time step, within a fixed radius of the center of the playfield, using a 10x10 grid to encode the control points, which is embedded using 2D convolutions and fully connected layers. The protagonists and antagonists train via PPO using 2D CNN operations. We refer the reader to Appendix of Jiang et al. (2021a) for more details about the architecture.

Hyperparameters Since our proposed methodology builds off PAIRED, we inherited most of the parameters from Jiang et al. (2021a). For HiEnt baseline, we swept over adversary and student agent entropies in the range $\{0.0, 0.005, 0.05, 0.1\}$ for the students and $\{0.0, 0.005, 0.05, 0.1, 0.2\}$ for the adversary. For the BC baselines (both UniBC and BiBC), we swept over the following hyperparameters: $\{0.005, 0.01, 0.05, 0.1, 0.5\}$ for the KL Loss coefficient, $\{10, 25\}$ gradient updates for the KL Loss interval, $\{0.0, 0.005, 0.05, 0.1\}$ for the student entropy and $\{0.0, 0.005, 0.05, 0.1, 0.2\}$ for the adversary entropy. Similar to Jiang et al. (2021a), we used Test Returns from CarRacingF1-Germany, CarRacingF1-Italy and CarRacingF1-Singapore for validation performance averaged across 3 seeds. Figure 9 shows all the tracks which were used to assess zero-shot transfer performance at test time for 10 seeds.

A.2 MINIGRID

Environment Using the same setup as in (Dennis et al., 2020; Jiang et al., 2021a), a maze-based game named MiniGrid Chevalier-Boisvert et al. (2018) is used as the environment. The agent's objective is to navigate the maze and reach the end while avoiding walls. The agent receives a reward of $1 - \frac{T}{T_{max}}$ if it reaches the end, where T is the episode length and T_{max} is the maximum episode length (set to 250). If the agent fails to reach the end, a reward of 0 is given. The agent's perception of the environment includes its orientation and a 7x7 grid that encompasses the agent and the area immediately in front of it. The agent can take 7 actions, but only 3 of them are used in the game: turn left, turn right, and move forward. Both student agents, the protagonist and the antagonist, follow this. The mazes are generated by an adversary (the teacher agent) which is given N steps to place walls in a 13x13 grid, and then chooses the location of the end and the agent's starting position. The value of N is a constant 25 in the case of 25-blocks fixed budget experiment, and in the case of 0-60 Uniform-blocks, the value of N is uniformly sampled between [0,60]. The generator architecture uses a convolution layer to process the full grid observation, followed by an LSTM with a hidden dimension of 256 and two fully connected layers to produce action logits over the possible 169 cells. The student architecture is similar to the generator architecture, but it uses a convolution with 16 filters to process the partial observation and doesn't use random noise.

Hyperparameters Here, for the HiEnt baseline, we swept over adversary and student agent entropies in the range $\{0.0, 0.005, 0.05, 0.1\}$ for the students and $\{0.0, 0.005, 0.05, 0.1\}$ for the adversary. For the BC baselines (both UniBC and BiBC), we swept over the following the hyperparameters: $\{0.005, 0.01, 0.05, 0.1, 0.5\}$ for the KL Loss coefficient, $\{5, 10, 25\}$ gradient updates for the KL Loss interval, $\{0.0, 0.005, 0.05, 0.1\}$ for the student entropy and $\{0.0, 0.005, 0.05, 0.1\}$ for the adversary entropy. For the Evo experiments, we use all the hyperparameters same as ACCEL, except the level replay coefficient ρ which we sweep over $\{0.5, 0.9\}$. We use SixteenRooms, Labyrinth and Maze for assessing validation performance by running 3 seeds per experiment, and report the final results on challenging mazes (see Figure 10) by averaging over 5 seeds.

A.3 BIPEDALWALKER

Environment We use the environment introduced in Parker-Holder et al. (2022) and refer the readers to Appendix C (Parker-Holder et al., 2022) for implementation details of the environment as well as the ACCEL baseline.

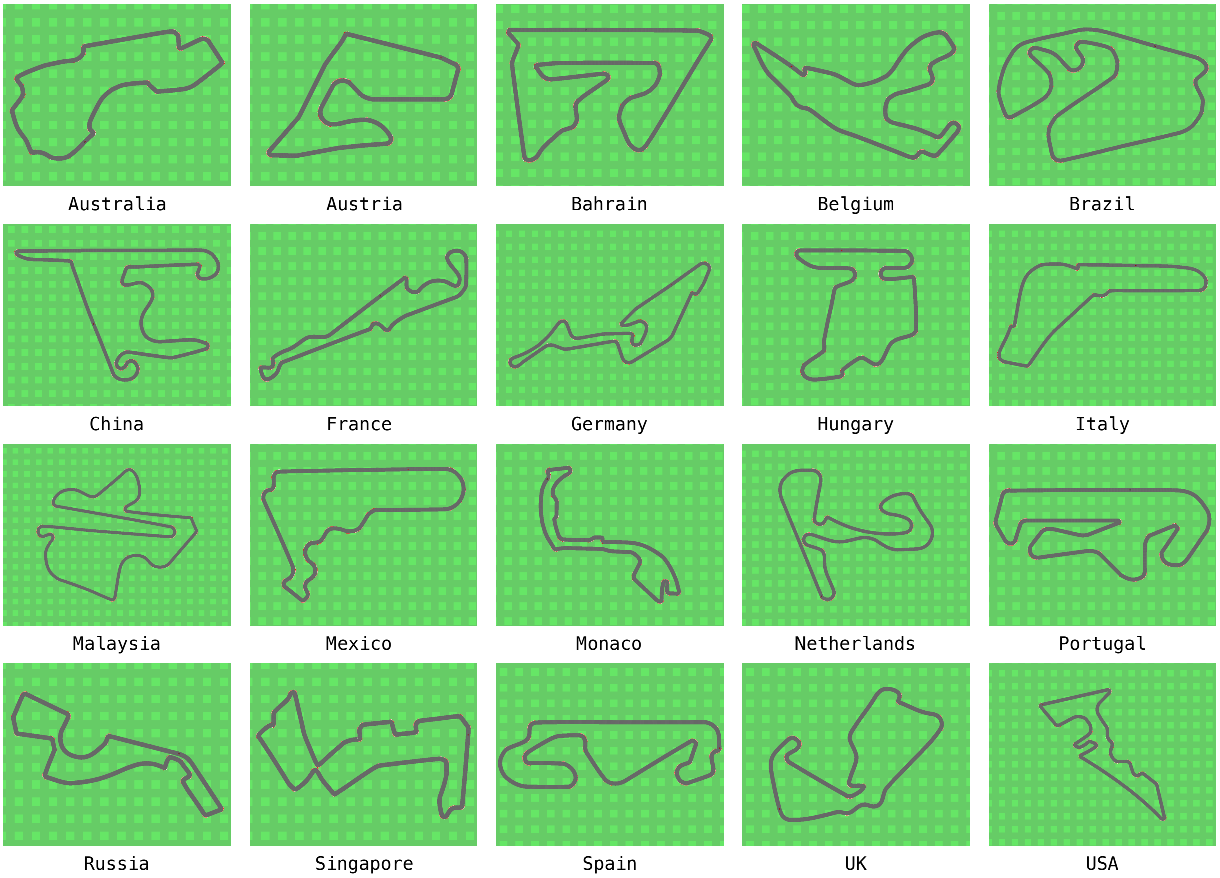


Figure 9: Test tracks used for assessing zero-shot performance in CarRacing F1 environment.

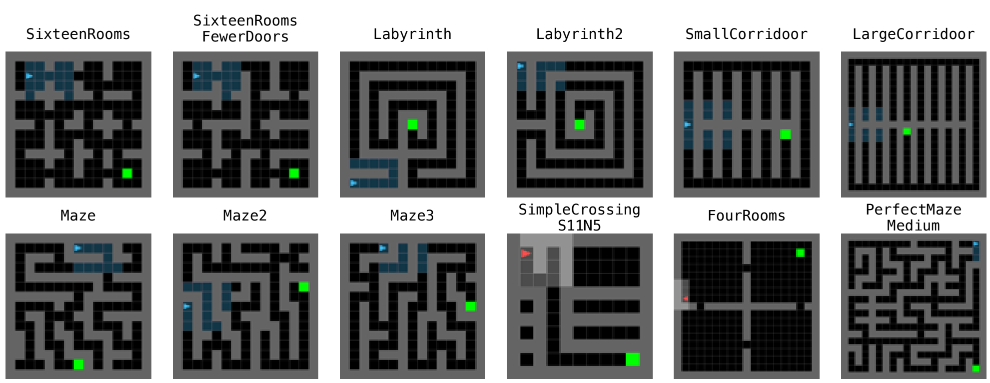


Figure 10: Test mazes used for assessing zero-shot performance in the Minigrid environment.

Hyperparameters Our PAIRED+Evo baseline is based off ACCEL, with the addition of one extra agent. Hence, all our hyperparameters are same as that of ACCEL, with the modifications highlighted in Algorithm 3.

Table 2: Table summarizing the hyperparameters for each of the methods.

PARAMETER	CARRACING	MINIGRID 0-60 UNIFORM	MINIGRID 25	BIPEDALWALKER
PPO				
γ	0.99	0.995	0.995	0.99
λ_{GAE}	0.9	0.95	0.95	0.9
PPO ROLLOUT LENGTH	125	256	256	2000
PPO EPOCHS	8	5	5	5
PPO MINIBATCHES PER EPOCH	4	1	1	32
PPO CLIP RANGE	0.2	0.2	0.2	0.2
PPO NUMBER OF WORKERS	16	32	32	16
ADAM LEARNING RATE	3E-4	1E-4	1E-4	3E-4
ADAM ϵ	1E-5	1E-5	1E-5	1E-5
PPO MAX GRADIENT NORM	0.5	0.5	0.5	0.5
PPO VALUE CLIPPING	NO	YES	YES	NO
RETURN NORMALIZATION	YES	NO	NO	YES
VALUE LOSS COEFFICIENT	0.5	0.5	0.5	0.5
STUDENT ENTROPY COEFFICIENT	0.0	0.0	0.0	1E-3
PAIRED				
GENERATOR ENTROPY COEFFICIENT	0.0	0.0	0.0	0.0
HiEnt				
STUDENT ENTROPY COEFFICIENT	0.05	0.005	0.005	-
GENERATOR ENTROPY COEFFICIENT	0.1	0.05	0.01	-
BiBC				
KL LOSS COEFFICIENT	0.5	0.01	0.1	-
KL LOSS INTERVAL	10	5	25	-
KL LOSS DIRECTION	BIDIRECTIONAL	BIDIRECTIONAL	BIDIRECTIONAL	-
STUDENT ENTROPY COEFFICIENT	0.005	0.005	0.005	-
GENERATOR ENTROPY COEFFICIENT	0.05	0.005	0.05	-
UniBC				
KL LOSS COEFFICIENT	0.5	0.01	0.1	-
KL LOSS INTERVAL	10	5	25	-
KL LOSS DIRECTION	UNIDIRECTIONAL	UNIDIRECTIONAL	UNIDIRECTIONAL	-
STUDENT ENTROPY COEFFICIENT	0.005	0.005	0.005	-
GENERATOR ENTROPY COEFFICIENT	0.05	0.005	0.05	-
ACCEL				
EDIT RATE, q	-	1.0	-	1.0
REPLAY RATE, p	-	0.8	-	0.9
BUFFER SIZE, K	-	4000	-	1000
SCORING FUNCTION	-	POSITIVE VALUE LOSS	-	POSITIVE VALUE LOSS
EDIT METHOD	-	RANDOM	-	RANDOM
LEVELS EDITED	-	EASY	-	EASY
PRIORITIZATION	-	RANK	-	RANK
TEMPERATURE, β	-	0.3	-	0.1
STALENESS COEFFICIENT, ρ	-	0.5	-	0.5
PLR				
SCORING FUNCTION	POSITIVE VALUE LOSS	POSITIVE VALUE LOSS	POSITIVE VALUE LOSS	POSITIVE VALUE LOSS
REPLAY RATE, p	0.5	0.5	0.5	0.5
BUFFER SIZE, K	10000	4000	4000	1000
Evo				
SCORING FUNCTION	-	POSITIVE VALUE LOSS	-	POSITIVE VALUE LOSS
REPLAY RATE (PAIRED+EVO), p	-	0.5	-	0.5
REPLAY RATE (FLEXPAIRED+EVO), p	-	0.9	-	0.5
BUFFER SIZE, K	-	4000	-	1000

B ADDITIONAL RESULTS

Here we report the performance of each design choice on individual test environments. Table 3 shows the mean and standard error on each F1 track (Figure 9) when averaged over 10 seeds. Each seed is run for 10 episodes. Similarly, Table 4 and 5 report the performance on challenging mazes (Figure 10) when trained in the Minigrid environment with 0-60 uniform blocks. It can be clearly seen that our proposed design choices lead to strong gains in performance over PAIRED, with a high entropy bonus having an especially big impact on the zero-shot generalization capability.

Figure 11 (top) shows the mean return achieved by the protagonist, antagonist and the adversary throughout training in the generated tracks. We also report the performance of protagonist in the vanilla CarRacing track used for validation purposes. The bottom row shows the entropy loss of each agent as well as the complexity of the tracks generated by the adversary.

Similarly, Figure 12 (a) and 13 show the mean returns achieved by the three agents in the generated mazes (top row) and the entropy loss of each agent in those mazes (bottom row). Figure 12 (b) shows the complexity of the mazes generated by the baselines when trained with a fixed budget of 25 blocks. PAIRED experiences a sharp decline in the

Environment	Robust PLR	PAIRED	HiEnt	UniBC	BiBC	UniBC+Ent	BiBC+HiEnt
CarRacingF1-Australia	692.3 ± 14.96	100.28 ± 21.67	651.72 ± 13.19	167.02 ± 18.07	273.58 ± 27.92	517.53 ± 24.04	$\mathbf{779.11 \pm 8.69}$
CarRacingF1-Austria	615.14 ± 12.93	92.15 ± 24.0	597.53 ± 6.88	226.64 ± 17.61	286.47 ± 26.47	487.17 ± 17.67	$\mathbf{758.52 \pm 10.83}$
CarRacingF1-Bahrain	589.83 ± 15.05	-34.96 ± 18.8	556.74 ± 16.58	42.46 ± 19.5	192.03 ± 27.19	429.02 ± 22.03	$\mathbf{631.47 \pm 14.71}$
CarRacingF1-Belgium	473.52 ± 12.22	72.19 ± 20.14	452.7 ± 6.6	143.56 ± 16.76	229.74 ± 22.13	301.16 ± 16.98	$\mathbf{587.27 \pm 15.13}$
CarRacingF1-Brazil	454.92 ± 13.35	75.73 ± 18.16	435.26 ± 13.2	150.35 ± 18.02	188.18 ± 22.24	368.09 ± 14.52	$\mathbf{557.48 \pm 19.77}$
CarRacingF1-China	227.71 ± 24.37	-100.5 ± 9.15	180.44 ± 30.45	-67.78 ± 6.95	117.33 ± 26.1	113.43 ± 23.97	$\mathbf{511.67 \pm 23.78}$
CarRacingF1-France	478.31 ± 22.35	-80.76 ± 12.73	482.9 ± 28.46	61.25 ± 20.53	187.0 ± 29.93	299.8 ± 22.46	$\mathbf{603.53 \pm 15.62}$
CarRacingF1-Germany	498.59 ± 17.9	-32.51 ± 16.05	471.62 ± 13.22	115.9 ± 22.96	195.58 ± 21.53	311.54 ± 19.91	$\mathbf{532.02 \pm 14.86}$
CarRacingF1-Hungary	707.75 ± 17.5	97.6 ± 28.53	696.65 ± 17.79	137.16 ± 20.82	229.81 ± 28.15	517.05 ± 26.73	$\mathbf{739.13 \pm 9.53}$
CarRacingF1-Italy	624.96 ± 11.94	131.56 ± 23.71	568.26 ± 6.21	233.89 ± 24.51	300.09 ± 25.83	502.34 ± 19.91	$\mathbf{780.99 \pm 8.88}$
CarRacingF1-Malaysia	399.86 ± 17.51	-26.18 ± 16.61	386.58 ± 11.78	37.31 ± 14.04	154.91 ± 23.64	306.64 ± 14.98	$\mathbf{519.64 \pm 17.22}$
CarRacingF1-Mexico	$\mathbf{712.1 \pm 12.48}$	66.53 ± 31.4	$\mathbf{711.34 \pm 10.05}$	183.82 ± 26.69	237.1 ± 30.77	543.13 ± 23.33	$\mathbf{697.4 \pm 12.06}$
CarRacingF1-Monaco	485.64 ± 19.46	-28.3 ± 18.18	356.1 ± 26.13	98.85 ± 19.73	181.65 ± 26.54	309.81 ± 23.53	$\mathbf{605.03 \pm 17.37}$
CarRacingF1-Netherlands	419.22 ± 25.24	70.41 ± 20.35	464.78 ± 17.1	135.91 ± 18.48	231.52 ± 23.17	312.16 ± 23.39	$\mathbf{612.2 \pm 17.67}$
CarRacingF1-Portugal	483.44 ± 12.7	-48.96 ± 13.0	424.54 ± 14.99	92.87 ± 16.31	177.13 ± 24.81	370.01 ± 19.6	$\mathbf{621.8 \pm 16.41}$
CarRacingF1-Russia	$\mathbf{649.34 \pm 13.55}$	51.26 ± 20.52	603.0 ± 22.04	202.54 ± 22.4	248.56 ± 26.74	452.11 ± 24.86	$\mathbf{668.23 \pm 12.66}$
CarRacingF1-Singapore	566.37 ± 15.11	-34.96 ± 13.5	435.81 ± 19.94	71.41 ± 14.78	187.53 ± 23.83	343.82 ± 27.13	$\mathbf{644.05 \pm 13.25}$
CarRacingF1-Spain	621.64 ± 13.7	134.11 ± 23.79	609.63 ± 12.38	260.1 ± 20.17	298.67 ± 25.3	504.32 ± 16.79	$\mathbf{681.99 \pm 13.13}$
CarRacingF1-UK	537.58 ± 16.81	137.63 ± 24.67	558.9 ± 12.65	257.21 ± 18.42	295.4 ± 25.75	445.71 ± 16.96	$\mathbf{620.75 \pm 13.1}$
CarRacingF1-USA	380.65 ± 33.22	-119.17 ± 10.99	324.71 ± 27.69	29.12 ± 16.07	185.31 ± 28.58	294.41 ± 31.42	$\mathbf{667.87 \pm 12.87}$
Mean	530.94 ± 6.67	26.16 ± 15.22	498.46 ± 6.3	128.98 ± 12.13	219.88 ± 23.08	386.46 ± 12.29	$\mathbf{641.01 \pm 7.21}$

Table 3: Mean and Standard Error averaged over 10 seeds for test returns of each agent on individual F1 track from the **CarRacing F1 environment**.

Environment	ACCEL	PAIRED	HiEnt	UniBC	BiBC	UniBC+HiEnt	BiBC+HiEnt
Labyrinth	$\mathbf{0.73 \pm 0.18}$	$\mathbf{0.46 \pm 0.23}$	$\mathbf{0.7 \pm 0.2}$	0.36 ± 0.18	0.02 ± 0.02	0.33 ± 0.1	$\mathbf{0.48 \pm 0.2}$
Labyrinth2	$\mathbf{0.62 \pm 0.2}$	0.15 ± 0.14	$\mathbf{0.67 \pm 0.18}$	$\mathbf{0.46 \pm 0.22}$	0.0 ± 0.0	0.26 ± 0.15	$\mathbf{0.46 \pm 0.19}$
LargeCorridor	$\mathbf{0.64 \pm 0.2}$	0.16 ± 0.08	$\mathbf{0.63 \pm 0.19}$	$\mathbf{0.62 \pm 0.16}$	0.16 ± 0.16	$\mathbf{0.57 \pm 0.11}$	$\mathbf{0.47 \pm 0.2}$
Maze	$\mathbf{0.79 \pm 0.2}$	0.01 ± 0.01	0.19 ± 0.11	0.06 ± 0.05	0.0 ± 0.0	0.03 ± 0.02	0.14 ± 0.08
Maze2	$\mathbf{0.33 \pm 0.18}$	0.02 ± 0.02	$\mathbf{0.54 \pm 0.16}$	$\mathbf{0.36 \pm 0.2}$	0.0 ± 0.0	$\mathbf{0.38 \pm 0.15}$	$\mathbf{0.24 \pm 0.19}$
Maze3	$\mathbf{0.53 \pm 0.2}$	0.15 ± 0.15	0.39 ± 0.19	$\mathbf{0.8 \pm 0.2}$	0.19 ± 0.19	$\mathbf{0.5 \pm 0.13}$	$\mathbf{0.5 \pm 0.14}$
MiniGrid-FourRooms	$\mathbf{0.51 \pm 0.04}$	0.38 ± 0.06	0.42 ± 0.02	$\mathbf{0.49 \pm 0.02}$	0.25 ± 0.06	$\mathbf{0.55 \pm 0.05}$	$\mathbf{0.51 \pm 0.04}$
MiniGrid-SimpleCrossingS11N5	$\mathbf{0.82 \pm 0.02}$	0.48 ± 0.16	0.62 ± 0.04	0.65 ± 0.03	0.27 ± 0.14	0.68 ± 0.08	$\mathbf{0.86 \pm 0.04}$
PerfectMazeMedium	$\mathbf{0.5 \pm 0.05}$	0.17 ± 0.07	$\mathbf{0.47 \pm 0.15}$	0.24 ± 0.07	0.12 ± 0.11	0.25 ± 0.04	0.31 ± 0.14
SixteenRooms	0.64 ± 0.18	0.16 ± 0.13	0.42 ± 0.16	$\mathbf{0.91 \pm 0.07}$	0.2 ± 0.18	$\mathbf{0.83 \pm 0.07}$	0.59 ± 0.18
SixteenRoomsFewerDoors	$\mathbf{0.55 \pm 0.19}$	0.1 ± 0.1	$\mathbf{0.57 \pm 0.23}$	0.25 ± 0.19	0.03 ± 0.03	$\mathbf{0.64 \pm 0.17}$	$\mathbf{0.41 \pm 0.17}$
SmallCorridor	$\mathbf{0.67 \pm 0.16}$	0.25 ± 0.17	$\mathbf{0.63 \pm 0.22}$	$\mathbf{0.74 \pm 0.17}$	0.04 ± 0.04	$\mathbf{0.63 \pm 0.07}$	$\mathbf{0.4 \pm 0.19}$
Mean	$\mathbf{0.61 \pm 0.03}$	0.21 ± 0.07	$\mathbf{0.52 \pm 0.13}$	0.49 ± 0.05	0.11 ± 0.05	0.47 ± 0.05	0.45 ± 0.11

Table 4: Mean and Standard Error for Solved Rates averaged over 5 seeds of each agent on individual mazes when trained in the **Minigrid 0-60 Uniform-Blocks environment**.

Environment	ACCEL	PAIRED+Evo	FlexPAIRED+Evo
Labyrinth	$\mathbf{0.73 \pm 0.18}$	$\mathbf{0.51 \pm 0.28}$	$\mathbf{0.48 \pm 0.2}$
Labyrinth2	$\mathbf{0.62 \pm 0.2}$	0.2 ± 0.14	$\mathbf{0.4 \pm 0.24}$
LargeCorridor	$\mathbf{0.64 \pm 0.2}$	$\mathbf{0.53 \pm 0.27}$	0.23 ± 0.19
Maze	$\mathbf{0.79 \pm 0.2}$	$\mathbf{0.5 \pm 0.27}$	0.06 ± 0.05
Maze2	$\mathbf{0.33 \pm 0.18}$	$\mathbf{0.28 \pm 0.24}$	$\mathbf{0.28 \pm 0.18}$
Maze3	$\mathbf{0.53 \pm 0.2}$	$\mathbf{0.44 \pm 0.25}$	$\mathbf{0.33 \pm 0.18}$
MiniGrid-FourRooms	$\mathbf{0.51 \pm 0.04}$	$\mathbf{0.42 \pm 0.09}$	0.38 ± 0.05
MiniGrid-SimpleCrossingS11N5	$\mathbf{0.82 \pm 0.02}$	$\mathbf{0.86 \pm 0.06}$	0.69 ± 0.06
PerfectMazeMedium	$\mathbf{0.5 \pm 0.05}$	$\mathbf{0.42 \pm 0.14}$	$\mathbf{0.37 \pm 0.11}$
SixteenRooms	$\mathbf{0.64 \pm 0.18}$	$\mathbf{0.78 \pm 0.13}$	0.26 ± 0.16
SixteenRoomsFewerDoors	$\mathbf{0.55 \pm 0.19}$	$\mathbf{0.48 \pm 0.23}$	$\mathbf{0.21 \pm 0.16}$
SmallCorridor	$\mathbf{0.67 \pm 0.16}$	$\mathbf{0.58 \pm 0.25}$	$\mathbf{0.62 \pm 0.24}$
Mean	$\mathbf{0.61 \pm 0.03}$	$\mathbf{0.5 \pm 0.14}$	0.36 ± 0.08

Table 5: Zero-shot test performance results for PAIRED+Evo and FlexPAIRED+Evo in the test Minigrid mazes when trained in the **Minigrid 0-60 Uniform-Blocks environment** averaged over 5 seeds and 100 episodes per seed. We report mean and standard error for Solved Rate as the metric here, highlighting in bold the best performing agents.

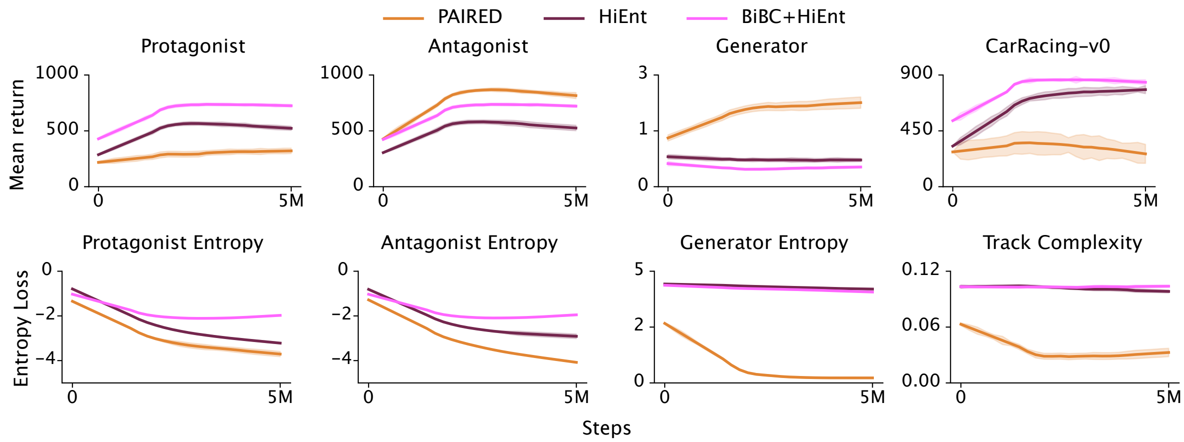
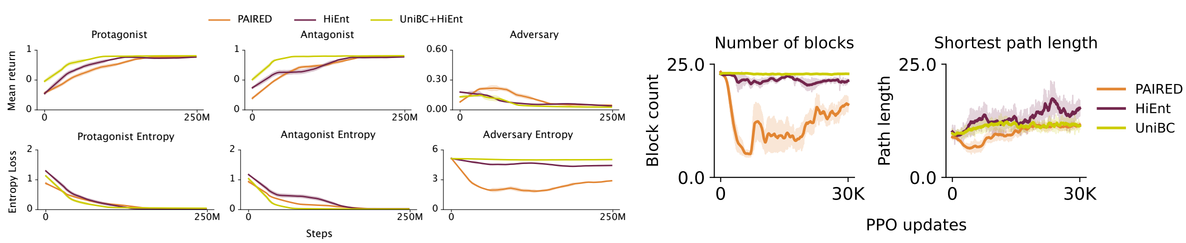


Figure 11: Mean agent returns throughout training for all agents in the **CarRacing F1 environment**.

number of blocks used and hence generates extremely sparse mazes (see Figure 15), whereas the other two baselines help the adversary in overcoming this problem.



(a) Training returns and Entropy loss (b) Grid Complexity

Figure 12: Mean agent returns throughout training for all agents in the **Minigrid 25-blocks environment**.

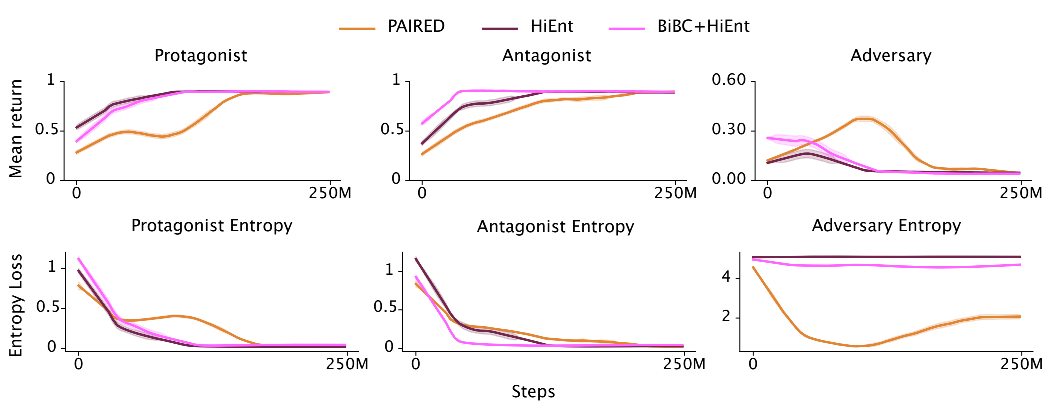


Figure 13: Mean agent returns throughout training for all agents in the **Minigrid [0-60] Uniform-blocks environment**.

C Psuedocodes

Here we describe the algorithm of BC framework. Our algorithm is the same as that of PAIRED's (Dennis et al., 2020), with an additional KL-divergence term highlighted in blue font. This KL-divergence term can be applied either to both the student agents (BiBC) or to just the protagonist (UniBC), after every N gradient update steps, where $N \in \mathbb{N}$.

Algorithm 2 PAIRED+BC

Input: Initial policies for protagonist π^P, antagonist π^A, initial environment generator $\tilde{\Lambda}$
while not converged **do**
  Use $\tilde{\Lambda}$ to generate environment parameters $\vec{\theta}$
  Collect a trajectory τ^P using π^P in environment $\vec{\theta}$
  Update π^P to minimize regret using $\mathcal{L}_{ppo}(\pi_P) + D_{KL}^{\tau^P}(\pi_A||\pi_P)$
  Collect a trajectory τ^A using π^A in environment $\vec{\theta}$
  Update π^A to maximize regret using $\mathcal{L}_{ppo}(\pi_A) + D_{KL}^{\tau^A}(\pi_P||\pi_A)$ # if bidirectional BC
  Compute the regret as:
  $\textsc{Regret}^{\vec{\theta}}(\pi^P, \pi^A) = V^{\vec{\theta}}(\pi^A) - V^{\vec{\theta}}(\pi^P)$
  Update $\tilde{\Lambda}$ to maximize regret
end while

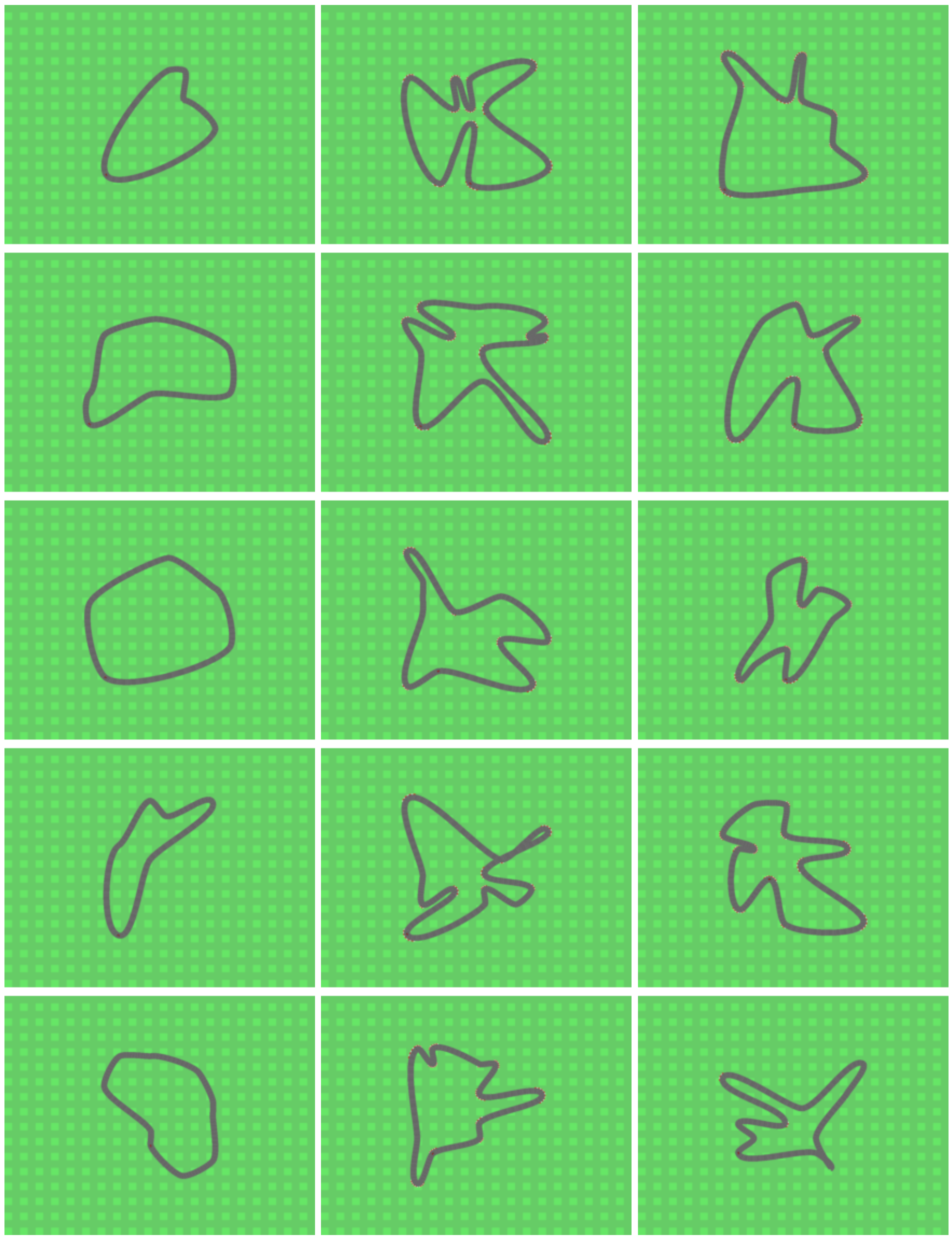
For PAIRED+Evo, we modify ACCEL (Parker-Holder et al., 2022) (highlighted in viole~~n~~t font) to utilize an additional agent which acts as the antagonist, and modify the replay buffer to store or replay levels based on the calculated regret.

Algorithm 3 PAIRED+Evo

Input: Initial policies for protagonist π^P, antagonist π^A, initial environment generator $\tilde{\pi}$, level buffer $\mathbf{\Lambda}$
while not converged **do**
  Sample replay decision $d \sim P_D(d)$
  if d=0 **then**
    # Evaluate on levels but do not update
    Use $\tilde{\pi}$ to generate environment parameters $\vec{\theta}$
    Collect a trajectory τ^P using π^P in environment $\vec{\theta}$ with **stop gradient**
    Collect a trajectory τ^A using π^A in environment $\vec{\theta}$ with **stop gradient**
    Compute the regret as:
    $\textsc{Regret}^{\vec{\theta}}(\pi^P, \pi^A) = U^{\vec{\theta}}(\pi^A) - U^{\vec{\theta}}(\pi^P)$
    Add $\vec{\theta}$ to $\mathbf{\Lambda}$ if score S meets threshold
  else
    # Train on curated high regret levels
    Sample replay level, $\vec{\theta} \sim \mathbf{\Lambda}$
    Collect a trajectory τ^P using π^P in environment $\vec{\theta}$
    Update π^P
    Collect a trajectory τ^A using π^A in environment $\vec{\theta}$
    Update π^A
    # Edit previously high regret levels and evaluate
    Edit $\vec{\theta}$ to produce $\vec{\theta}'$
    Collect a trajectory τ^P using π^P in environment $\vec{\theta}$ with **stop gradient**
    Collect a trajectory τ^A using π^A in environment $\vec{\theta}$ with **stop gradient**
    Compute the regret as:
    $\textsc{Regret}^{\vec{\theta}}(\pi^P, \pi^A) = U^{\vec{\theta}}(\pi^A) - U^{\vec{\theta}}(\pi^P)$
    Add $\vec{\theta}$ to $\mathbf{\Lambda}$ if score S meets threshold
  end if
end while

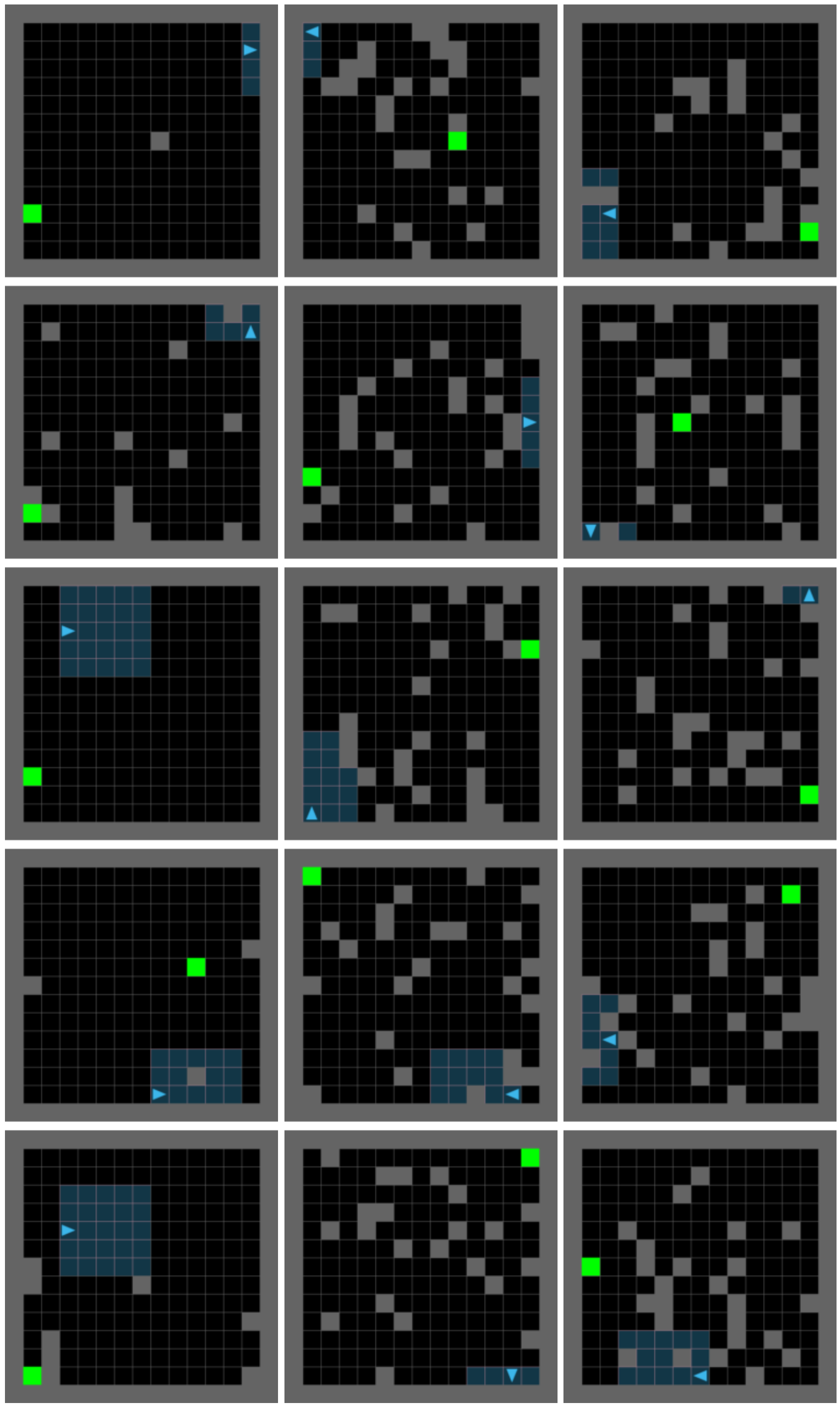
D SAMPLE TRACKS

Figures 14, 15 and 16 show some sample levels generated by the adversary from each baseline. In CarRacing, PAIRED's adversary keeps on generating simple round tracks which do not transfer to real-world F1 tracks at test time. Similarly, in Minigrid, PAIRED's adversary frequently generates empty mazes which prevent the protagonist from learning navigation skills in these mazes. On the other hand, the adversary in HiEnt and BiBC+HiEnt generates highly complex levels that significantly improve zero-shot generalization.



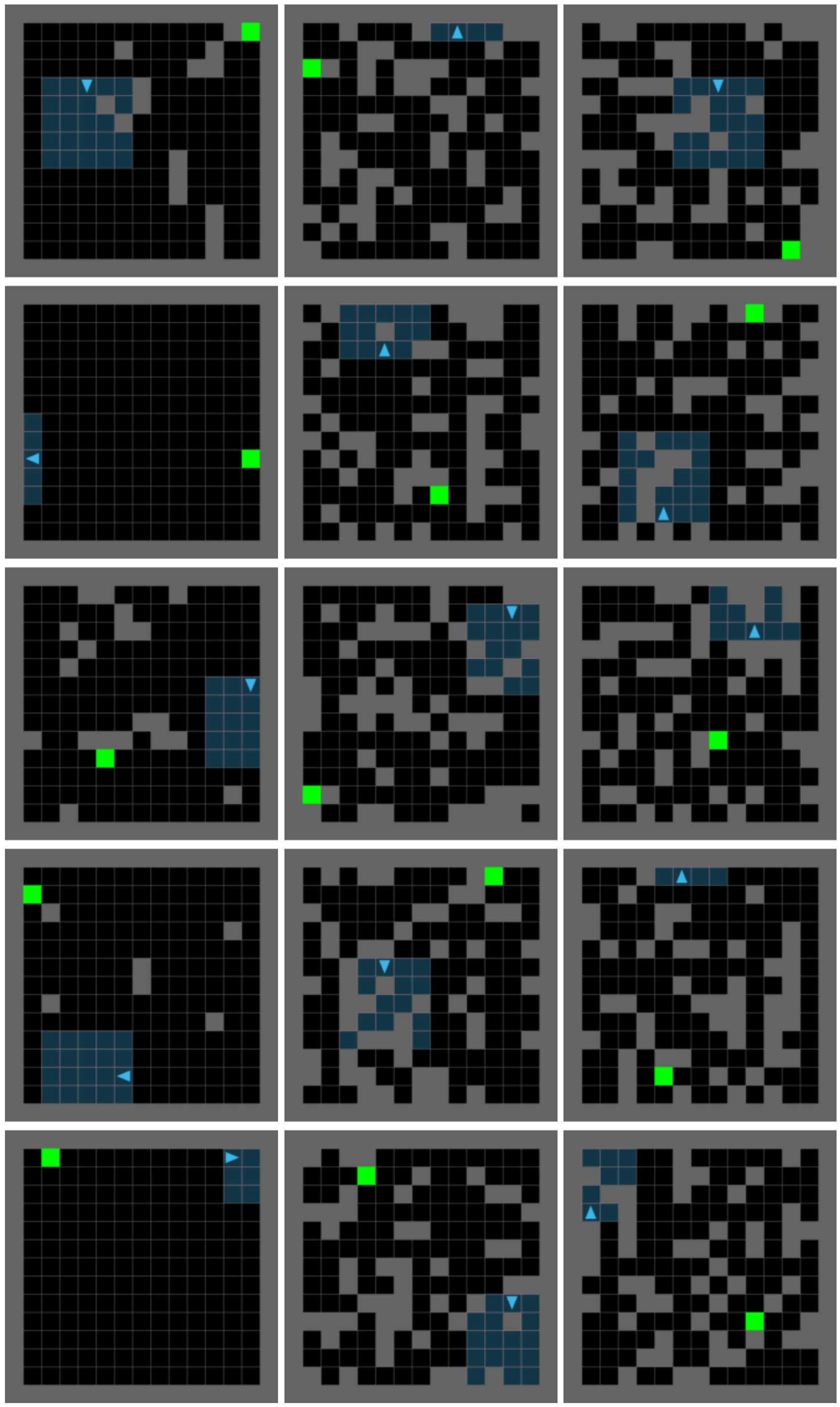
(a) PAIRED (b) PAIRED+HiEnt (c) PAIRED+BiBC+HiEnt

Figure 14: Sample tracks generated by the adversary in CarRacing Environment.



(a) PAIRED (b) PAIRED+HiEnt (c) PAIRED+UniBC+HiEnt

Figure 15: Sample tracks generated by the adversary in MiniGrid Environment with 25-blocks budget.



(a) PAIRED (b) PAIRED+HiEnt (c) PAIRED+UniBC+HiEnt

Figure 16: Sample tracks generated by the adversary in MiniGrid Environment with [0-60] Uniform-blocks budget.