UNSUPERVISED LEARNING OF EFFICIENT EXPLORATION: PRE-TRAINING ADAPTIVE POLICIES VIA SELF-IMPOSED GOALS

Octavio Pappalardo *
Independent Researcher
octaviopappalardo@gmail.com

ABSTRACT

Unsupervised pre-training can equip reinforcement learning agents with prior knowledge and accelerate learning in downstream tasks. A promising direction, grounded in human development, investigates agents that learn by setting and pursuing their own goals. The core challenge lies in how to effectively generate, select, and learn from such goals. Our focus is on broad distributions of downstream tasks where solving every task zero-shot is infeasible. Such settings naturally arise when the target tasks lie outside of the pre-training distribution or when their identities are unknown to the agent. In this work, we (i) optimize for efficient multi-episode exploration and adaptation within a meta-learning framework, and (ii) guide the training curriculum with evolving estimates of the agent's post-adaptation performance. We present ULEE, an unsupervised meta-learning method that combines an in-context learner with an adversarial goal-generation strategy that maintains training at the frontier of the agent's capabilities. On XLand-MiniGrid benchmarks, ULEE pre-training yields improved exploration and adaptation abilities that generalize to novel objectives, environment dynamics, and map structures. The resulting policy attains improved zero-shot and few-shot performance, and provides a strong initialization for longer fine-tuning processes. It outperforms learning from scratch, DIAYN pre-training, and alternative curricula. Code is available at: https://github.com/Octavio-Pappalardo/ulee-jax

1 INTRODUCTION

Large-scale pre-training has underpinned many real-world successes of deep learning in computer vision (Krizhevsky et al., 2012; Chen et al., 2020; Grill et al., 2020; He et al., 2022) and language modeling (Devlin, 2018; Brown, 2020). These advancements have motivated analogous efforts in deep reinforcement learning (RL) (Liu & Abbeel, 2021; Schwarzer et al., 2021; Seo et al., 2022), where the prevailing paradigm remains to train agents from scratch for each new task. The ultimate objective is to develop foundation policies: agents equipped with transferable knowledge that address RL's fundamental challenges of sample inefficiency and lack of generalization. We investigate unsupervised RL, where agents interact freely with their environments without access to extrinsic rewards, as a means to acquire such policies. This setting raises key questions regarding what data an agent should seek and what it should learn from it. These questions are tightly coupled because the collected data is a direct consequence of the agent's incorporated behaviors. Ultimately, we seek a system that can continuously gather useful data at scale in an unsupervised, open-ended fashion and leverage it to acquire transferable capabilities.

A promising direction is to design agents that autonomously generate their own goals and learn by attempting to solve them (Oudeyer et al., 2007; Nair et al., 2018; Forestier et al., 2022). Our focus is on how such goals should be generated, selected, and exploited for pre-training. This connects to automatic curriculum learning, yet existing methods have often assumed either a fixed set of goals or a goal space that is closely aligned with the evaluation tasks. Progress toward foundation policies

*Work conducted as an independent researcher. Currently at University College London (UCL).

requires assuming little or no information about downstream objectives and instead pre-training for a broad range of human-relevant tasks. Moreover, a general base policy must have the capacity to contend not only with novel objectives but also with limited information, whether in the form of partial observability, uncertain dynamics, or unspecified goals. Thus, it must be effective across a spectrum of difficulty levels, delivering quick performance on easier tasks and sample-efficient adaptation over long timescales on complex ones.

With these desiderata in mind, we introduce a metric that guides goal generation using the agent's performance on goals after an adaptation budget. This contrasts with much prior work that evaluates goal desirability based on immediate performance without any task-specific adaptation (Sukhbaatar et al., 2017; Florensa et al., 2018). We pair this metric with an adversarially trained goal-generation system that proposes challenging yet achievable goals for the pre-trained policy to learn from. This yields an adaptive curriculum that maintains goals at intermediate difficulty and avoids uninformative extremes of too easy or unsolvable goals. Shifting toward a measure of difficulty that relies on post-adaptation performance aligns better with the intended evaluation setting, where the policy must face novel tasks that require adaptation. Moreover, it focuses training on more challenging tasks and leads to coverage of a broader distribution, with more tasks becoming easy and fewer remaining effectively unsolvable.

Anticipating the need for test-time adaptation naturally points to meta-learning (Duan et al., 2016; Finn et al., 2017), which explicitly optimizes for efficient learning. Meta-learning has primarily been studied with pre-training carried out over hand-designed task distributions, and its integration with unsupervised settings remains largely underexplored (Gupta et al., 2018). We bridge these directions by introducing an unsupervised meta-learning algorithm that meta-learns a base policy using an automatic curriculum of self-generated goals guided by our post-adaptation difficulty metric.

Our main contributions are as follows:

- We introduce a post-adaptation task-difficulty metric for guiding automatic goal generation and selection in unsupervised autotelic agents.
- We present ULEE, an unsupervised meta-learning method composed of (1) an in-context learner trained with self-imposed goals, (2) a difficulty-prediction network estimating post-adaptation performance, (3) an adversarial agent trained to find challenging candidate goals, and (4) a sampling strategy that selects goals within a desired difficulty range.
- We evaluate ULEE in complex yet computationally accessible environments across multiple timescales and types of generalization. It outperforms baselines and ablations in exploration, fast adaptation, fine-tuning to fixed environments, and fine-tuning in meta-learning settings. All code is open-sourced.

2 Related Work

Several strategies have emerged to make use of reward-free objectives in RL. These include learning a world model (Sekar et al., 2020; Seo et al., 2022; Rajeswar et al., 2023), utilizing auxiliary objectives that aid representation learning (Jaderberg et al., 2016; Oord et al., 2018; Zhang et al., 2020a; Stooke et al., 2021; Schwarzer et al., 2021), and training agents with intrinsic rewards (Chentanez et al., 2004; Oudeyer et al., 2007; Schmidhuber, 2010) that lead to useful behaviors. The latter has been a major line of research into both pre-training and online exploration, with methods in both areas holding a close relationship. Most work in this line of research can be grouped into methods that consider predictions over some aspect of the environment and are guided by error, uncertainty, or progress in these predictions (Schmidhuber, 1991; Pathak et al., 2017; Burda et al., 2018; Pathak et al., 2019); those whose rewards come from measures of diversity in collected data (Bellemare et al., 2016; Hazan et al., 2019; Lee et al., 2019; Liu & Abbeel, 2021; Yarats et al., 2021); and those who seek to maximize empowerment (Klyubin et al., 2005; Mohamed & Jimenez Rezende, 2015; Gregor et al., 2016; Eysenbach et al., 2018; Sharma et al., 2019) or are guided by self-imposed goals (Sukhbaatar et al., 2017; Nair et al., 2018; Florensa et al., 2018; Pong et al., 2019; Forestier et al., 2022). Our work focuses on the latter and incorporates ideas from automatic curriculum learning (Bengio et al., 2009; Andrychowicz et al., 2017; Florensa et al., 2017; Matiisen et al., 2019; Jiang et al., 2021), which has proved capable of significantly accelerating learning in RL.

Intermediate difficulty. Previous work on unsupervised RL and automatic curriculum generation has explored dictating the agent's curriculum by continually selecting tasks of intermediate difficulty. In GoalGAN (Florensa et al., 2018), a GAN (Goodfellow et al., 2014) variant is trained to output goals that are neither too easy nor too hard for the agent. The goal space is fixed beforehand and remains the same for training and evaluation. The curriculum's aim is to accelerate learning across all feasible goals. Racaniere et al. (2019) address the same multitask setting but generate goals uniformly across all difficulty levels. Their goal generator is trained with custom objectives that promote goal validity, coverage, and controllable difficulty. They introduce a "judge" network trained on past goals to estimate success probability. Conditioning the judge and generator on environment observations enables them to handle procedurally generated and partially observable environments. Zhang et al. (2020b) propose a curriculum method that does not rely on adversarial training. They treat previously visited states as candidate goals and estimate goal difficulty from the epistemic uncertainty of the value function, approximated using an ensemble of value networks. They show that goals at both extremes of difficulty correspond to low uncertainty. AMIGo (Campero et al., 2020) is a method to improve online exploration in which a generative network is trained jointly with the agent to provide it with progressively more challenging yet achievable tasks. This interaction leads to the exploration of increasingly complex behaviors that assist the agent in finding the environment's sparse extrinsic reward. Like our work, these methods ground their curricula in difficulty-based metrics, though in settings that differ fundamentally from ours. None consider unsupervised pre-training followed by transfer to previously unseen tasks, and they don't consider scenarios where goal information isn't available. Instead, they train goal-conditioned policies and design curricula either for multitask learning or to accelerate progress on a single extrinsic task available alongside the unsupervised goals. Moreover, their difficulty estimates are derived from the agent's immediate performance, without allowing for task-specific adaptation. ASP (Sukhbaatar et al., 2017) is closer to our setup in that it also employs an adversarial policy ("Alice") to propose goals, but still optimizes a different objective. ASP is evaluated on a single external task using a goal-conditioned policy ("Bob"), which alternates between self-play and training on the target task. During self-play episodes, Bob is rewarded for either imitating or undoing Alice's actions. OpenAI et al. (2021) extend ASP with an imitation learning loss and scale it to evaluate the generalization of the goal-conditioned policy to held-out tasks. Beyond goal-based curricula, other works investigate adjusting task difficulty through the automatic selection or modification of training environments (Wang et al., 2019; Akkaya et al., 2019; Mehta et al., 2020; Team et al., 2021; 2023).

Unsupervised meta-learning. The setting of unsupervised meta-learning for RL, where meta-learning is performed on automatically acquired tasks, was first explored in Gupta et al. (2018). There, each latent skill z discovered by Diversity is All You Need (DIAYN) (Eysenbach et al., 2018) is associated with a self-generated goal, and the discriminator probabilities $D(z|s)$ are used as the rewards for that goal. Jabri et al. (2019) extends this approach with a dynamic curriculum and evaluates it in partially observable visual environments. Alternatively, Mutti et al. (2022) learns a policy that maximizes the entropy of the induced state-visitation distribution, with particular emphasis on adverse environments.

Ada. Our work also bears similarities to Ada (Team et al., 2023). Ada meta-learns an in-context learning agent (Duan et al., 2016; Wang et al., 2016; Mishra et al., 2017) on a large task distribution using a curriculum and evaluates performance on novel tasks in a few-shot regime. Unlike our method, Ada trains solely on tasks produced by its environment generator and does not incorporate intrinsic goals. In addition, its policy is explicitly conditioned on goal and environment-dynamics information, and the learned policy is not evaluated as an initialization for extended fine-tuning.

3 METHOD

We introduce ULEE (Unsupervised Learning of Efficient Exploration), an unsupervised pre-training method in which an intrinsically motivated agent meta-learns an adaptive policy from a curriculum of self-imposed goals. This section outlines its main components. Pseudocode and additional details are provided in Appendices A.1 and A.2.

Setting and Notation: Both pre-training and evaluation are performed over families of partially observable Markov decision processes (POMDPs). We denote by μ^{train} and μ^{eval} the distributions of pre-training and evaluation environments, respectively. Agent interactions with each environment

M occur in discrete time steps $t \in \{0, \ldots, T-1\}$, where $s_t \in \mathcal{S}$ is the underlying state of the environment at time step t, $o_t \sim \mathcal{O}(\cdot|s_t)$ is the observation the agent gets, and $a_t \in \mathcal{A}$ is the action it takes. Environments in our work differ in their initial state distribution ρ_M, transition dynamics $P_M(s_{t+1} \mid s_t, a)$, and reward function $r_M(s_t, a_t)$. In the pre-training phase, rewards are not provided by the environments but determined by the agent's own intrinsic goals. We write μ^{unsup} for the reward-free counterpart of μ^{train} and pair each $M \sim \mu^{\text{unsup}}$ with a goal-conditioned reward function to obtain full POMDPs. Although the policy operates on observations, for simplicity we give the goal-generation system and goal-conditioned reward function privileged access to full state information during pre-training.

3.1 Pre-trained Policy

The **Pre-trained Policy** π is the component trained with self-imposed goals and the sole component evaluated at test time. Prior work on self-generated goals typically learns goal-conditioned policies. Although goal conditioning scales to broad goal distributions, it is ill-suited for evaluation settings in which (i) the goal to be achieved is unknown and must be discovered, (ii) there is no suitable way to encode the goal, or (iii) the goal representation is sufficiently out-of-distribution to be uninterpretable by the policy. These scenarios place goal-conditioned policies outside the regime where they excel – known goals with recognizable representations. Some methods address this by treating the goal-conditioned policy as a low-level controller within a hierarchy and deploying the high-level controller (Kulkarni et al., 2016; Vezhnevets et al., 2017; Sukhbaatar et al., 2018; Sharma et al., 2019). In contrast, we pre-train an *unconditioned* policy that can be directly deployed.

We investigate the agent's ability to solve goals under various adaptation timescales, noting that longer horizons better match the demands of challenging or unfamiliar tasks. To that end, we adopt a meta-learning approach to train π, which allows explicit optimization of exploration and adaptation over multiple episodes. In our implementation, we take a black-box approach (Duan et al., 2016; Wang et al., 2016), where the agent interacts with environments over a sequence of contiguous episodes and selects actions based on its full interaction history. By conditioning on past observations, actions, and rewards, the policy learns to adapt *in-context*. We refer to the whole multi-episode interaction as a *lifetime* and train π to maximize its expected discounted lifetime return.

$$\mathcal{J}(\pi) = \mathbb{E}_{M \sim \mu^{\text{unsup}}, g \sim p(g|M)} \left[\mathbb{E}_{\rho_M,\, P_M,\, \pi} \left[\sum_{j=1}^{H} \sum_{t=0}^{T-1} \gamma^{(j-1)T+t}\; r_t^{(j)} \right] \right] \tag{1}$$

In Equation 1, $j \in \{1, \ldots, H\}$ indexes episodes within a lifetime, $r_t^{(j)}$ is the reward for the sampled goal g at time step t in episode j, $\gamma \in [0, 1)$ is the discount factor, and $p(g \mid M)$ is the evolving distribution of self-imposed goals induced by ULEE's goal-generation mechanism.

3.2 Goal Curriculum

In our work, goals are obtained as functions of states via $f : \mathcal{S} \to \mathcal{G}$. An agent accomplishes goal g at time step t if $f(s_{t+1}) = g$, meaning the goal is reached with action a_t. Our method is flexible to the choice of f. Our procedure finds and selects goals that are of intermediate difficulty according to the current capabilities of the Pre-trained Policy. This prevents it from focusing on goals that are too hard to get any feedback on or too easy to require new capabilities, and thus provides a strong learning signal (Florensa et al., 2018; Zhang et al., 2020b). We define the difficulty of a goal g for a policy π in environment M as the complement of π's expected success rate over the last K episodes of sequential interaction (Eq. 2). Thus, with H being the total number of episodes in the lifetime, the performance over the first $H - K$ episodes, which the agent leverages to explore and adapt, is ignored.

$$d(g; \pi, M) \;=\; 1 - \mathbb{E}_{\rho_M,\, P_M,\, \pi} \left[\frac{1}{K} \sum_{j=H-K+1}^{H} \mathbf{1}\Big\{ \exists\, t \in \{0, \ldots, T-1\} :\; f\big(s_{t+1}^{(j)}\big) = g \Big\} \right] \tag{2}$$

For each sampled environment $M \sim \mu^{\text{unsup}}$ and goal $g \sim p(g \mid M)$, running π for one lifetime yields a single-sample empirical estimate of that task's difficulty $\tilde{d}(g; \pi, M)$. We implement the intermediate-difficulty curriculum with three subsystems that dynamically adjust $p(g \mid M)$.

3.2.1 GOAL PROPOSAL

A **Goal-search Policy** π_{gs} is trained adversarially to reach hard goals. More specifically, it is trained to maximize $\mathcal{J}_{gs}(\pi_{gs}) = \mathbb{E}_{M,\pi_{gs}}\left[\sum_{t=0}^{T-1} \gamma^t r_t^{gs}\right]$, where γ is a discount factor and the goal-search rewards r^{gs} are the difficulties of the candidate goals encountered (Eq. 3).

$$r_t^{gs} = r^{gs}(s_t; \pi, M) = d(f(s_t); \pi, M) \tag{3}$$

In each training environment M, the Goal-search Policy runs for k episodes prior to the Pre-trained Policy. Let $s_0, s_1, \dots$ denote the sequence of states visited across these episodes. Then, the multiset of candidate goals collected by π_{gs} for M is defined as $GC_M = \{f(s_t) : t \in \{0, n, 2n, \dots\}\}$, where $n \in \mathbb{N}$ controls the temporal spacing between considered goals.

3.2.2 GOAL SELECTION

To decide with which goal to train the Pre-trained Policy among the candidate goals GC_M, we perform a random sample among goals at a desired level of difficulty. More specifically, we define a lower bound LB and an upper bound UB and sample uniformly among goals whose difficulty lies within these bounds (Eq. 4).

$$g_M \sim \text{Unif}(S), \quad S = \{g \in GC_M : LB \leq d(g; \pi, M) \leq UB\} \tag{4}$$

Here, g_M is the goal that is sampled for environment M, and π is the current Pre-trained Policy. If $S = \emptyset$, we instead sample uniformly from GC_M.

3.2.3 DIFFICULTY PREDICTION

In practice, there is no access to the goal difficulties needed to compute the goal-search rewards or to uncover which goals are in S. Even computing a single-sample empirical estimate requires running π for multiple episodes, which becomes prohibitively expensive and sample-inefficient. Consequently, we introduce a **Difficulty Predictor** network that can estimate the difficulty of goals the agent has not faced. By replacing the true goal difficulties with the Difficulty Predictor estimates $\hat{d}$, we can approximate the behavior of Equations 3 and 4 without additional environment interactions.

As the difficulty of goals depends on the agent's evolving capabilities, we keep a buffer B_g of triplets $(g, \xi_M, \tilde{d}(g))$ that holds only the most recent goals on which pre-training was performed. In each iteration, we train the Difficulty Predictor with a supervised L2 regression loss computed from goals in this buffer (Eq. 5).

$$\mathcal{L}_{\text{DP}}(\phi) \;=\; \frac{1}{|B_g|} \sum_{(g,\xi,\tilde{d}) \in B_g} \big(\, \hat{d}_\phi(g, \xi) \;-\; \tilde{d}(g) \,\big)^2. \tag{5}$$

The target $\tilde{d}(g)$ is a one-sample empirical estimate of goal difficulty obtained from the Pre-trained Policy's training lifetime on g. ξ_M denotes a generic object (e.g., $s_0 \sim \rho_M$) that carries information regarding the environment M on which the empirical estimate was obtained.

4 EXPERIMENTS

Unsupervised reinforcement learning methods can differ in the form of knowledge they aim to acquire, leading to various criteria for their evaluation. In this work, we assess the downstream utility of ULEE's Pre-trained Policy on novel tasks. Specifically, we ask:

Q1 What exploration capabilities does the Pre-trained Policy exhibit? (Sec. 4.3.1)

Q2 How well does the policy adapt with limited experience? (Sec. 4.3.2)

Q3 Does the Pre-trained Policy provide a strong initialization for fine-tuning under larger adaptation budgets? (Sec. 4.3.3)

Q4 Is the Pre-trained Policy an effective initialization for downstream meta-learning on curated task distributions? (Sec. 4.3.4)

Finally, Section 4.3.5 revisits Q1 and Q2 to test along an additional axis of generalization.

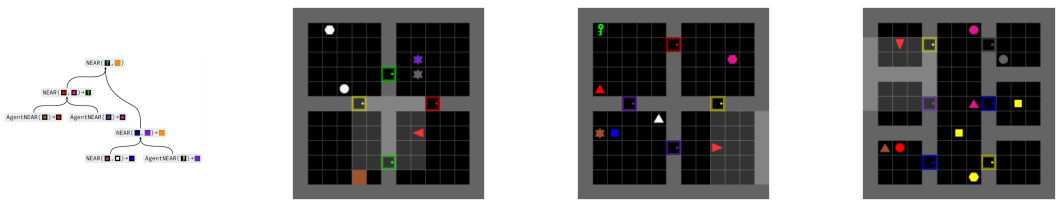
4.1 Benchmarks and Tasks

We evaluate on distributions of pre-sampled, procedurally generated, partially observable grid environments from the JAX-based XLand-MiniGrid (Nikulin et al., 2024), which retains the diversity of XLand (Team et al., 2021) while enabling orders of magnitude faster experimentation. Environments contain a goal and a set of rules governing object-object and agent-object interactions (Fig. 1a). Various types of goals, rules, and objects exist. We use two distributions of 1 million pre-sampled combinations of {goal, rules, initial objects} adhering to the following specifications:

- **trivial:** no prerequisite rules must be triggered before achieving the goal; three distractor objects not involved in achieving the goal are present.
- **small:** 0-2 rules that the agent must trigger before the goal becomes achievable; 2 distractor objects and 0-2 distractor rules that lead to dead ends are present.

We allocate 100 000 combinations to μ^{eval} and the remaining 90% to $\mu^{\text{train}}/\mu^{\text{unsup}}$. All environments use a 13x13 grid, symbolic 5x5 observations, and an episode horizon of 256 steps with early termination upon success. Each distribution is instantiated with either four- or six-room layouts, yielding our three benchmarks: 4Rooms-Trivial, 4Rooms-Small, and 6Rooms-Small (Fig. 1).

The goal mapping $f : \mathcal{S} \to \mathcal{G}$ used during unsupervised pre-training encodes an inductive bias into which behaviors are worth mastering and thus affects the difficulty of generalizing to XLand-MiniGrid extrinsic goals. We consider a mapping f_{counts} to the grid's per-object counts, where the agent succeeds when its interactions lead to the same count vector, and a mapping f_{grid} that defines goals by the agent's position and the grid's exact configuration, excluding the agent's orientation and pocket; success requires reproducing that configuration. These mappings serve as reasonable task-agnostic proxies. We utilize f_{counts} as the default mapping and evaluate f_{grid} exclusively on 4Rooms-Small, making the distinction explicit. See Appendix A.2.1 for further details on the environments.



(a) (b) (c) (d)

Figure 1: Panel (a), reproduced from Nikulin et al. (2024) (CC BY 4.0), shows a goal and the rules that must be triggered to achieve it, represented as a tree of depth 2. Analogously, tasks from the trivial and small benchmarks correspond to depth-0 and depth-1 trees. Panels (b)-(d) show example environments from the three benchmarks: 4Rooms-Trivial, 4Rooms-Small, and 6Rooms-Small.

4.2 Baselines and Ablations

Ablation Variants: We study ablations of ULEE's goal-generation mechanism. A first set of ablations denotes each variant as *ULEE (goal_search + sampling)*. The goal_search component is *adversarial* when the Goal-search Policy is trained to reach difficult goals, and *random* when replaced by a random policy. The sampling component is *bounded* when goals are drawn uniformly from those of intermediate difficulty and *uniform* when drawn from all candidates. Our main method

(Sec. 3) corresponds to ULEE (adversarial + bounded), which we refer to simply as ULEE. In addition, we introduce ULEE (SED), which estimates difficulty from the Pre-trained Policy's immediate rather than post-adaptation performance. After meta-learning with tasks over multi-episode interactions, we compute their *single-episode difficulty* using K additional episodes run as if they were first episodes (resetting memory before each). These extra episodes are excluded from the ablation step counts reported in figures and text.

Baselines: We compare our method against DIAYN (Eysenbach et al., 2018), an empowerment-based unsupervised RL method used as a pre-training baseline (Q1–Q3); PPO (Schulman et al., 2017) trained from scratch on fixed tasks as a standard model-free baseline (Q3); RND (Burda et al., 2018) trained from scratch on fixed tasks as a popular baseline that uses intrinsic rewards to improve online exploration (Q3); and RL^2 (Duan et al., 2016; Wang et al., 2016) trained from scratch on a curated task distribution as a standard meta-learning baseline (Q4).

Implementations: All policies are optimized with PPO and use the same Transformer-XL backbone (Dai et al., 2019; Parisotto et al., 2020), with separate MLP heads for the Actor and Critic. Variants with bounded sampling use $LB = 0.1$ and $UB = 0.9$. For DIAYN, the skill discriminator has access to the full state and conditions on $f(s)$. Further information is provided in Appendix A.2.

4.3 Experimental Results

4.3.1 Exploration

To assess exploration capabilities, we measure the percentage of goals from μ^{eval} reached under budgets of 1-20 episodes. ULEE's Pre-trained Policy substantially outperforms random behavior (the common default when learning from scratch) and DIAYN pre-training, reaching more than twice as many goals at the 20-episode mark across all benchmarks (Fig. 2). Ablations highlight the need for mechanisms to avoid pre-training on trivial goals and show that guiding the curriculum by post-adaptation performance, rather than immediate success, becomes increasingly beneficial as benchmark difficulty grows. Variants using an adversarial Goal-search Policy achieve the best results, and bounded goal sampling proves effective when goal search is random.

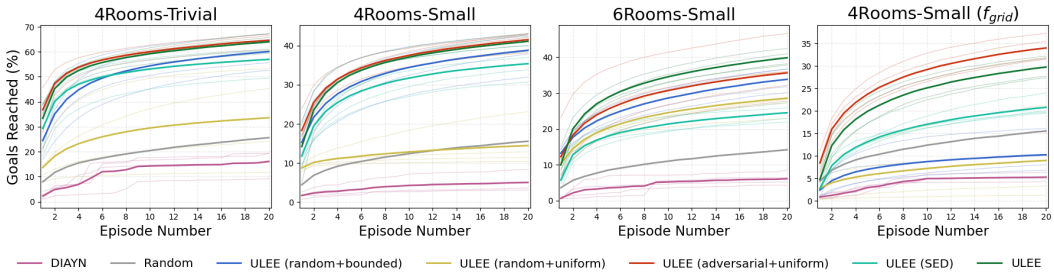


Figure 2: Percentage of μ^{eval} goals reached under different exploration budgets. A goal is considered reached at episode j if it was achieved in any episode $\leq j$. Results are averaged across 4 seeds, with individual seeds overlaid as faint thin lines.

4.3.2 Fast Adaptation

A policy must not only discover goal solutions but also know how to reliably achieve them once it does. Figure 3a evaluates gradient-free, few-shot adaptation over 30-episode lifetimes. ULEE's Pre-trained Policy leverages its interaction history to improve steadily, reaching up to a $3\times$ increase in mean return by the 30th episode. Variants trained with a random Goal-search Policy manage some adaptation early on but soon stagnate. For DIAYN, adaptation occurs at a single, discrete point by selecting the best-performing skill after all have been evaluated. Figure 3b reports the agent's post-adaptation return (measured as the mean over the last 10 episodes) by task percentile. ULEE outperforms all baselines and ablations. However, the hardest out-of-distribution tasks remain challenging. In two of the three benchmarks, no return was achieved on 60% of tasks. Figure 3c shows that as the pre-training budget is increased (up to 5 billion steps), ULEE's post-adaptation performance continues to improve. DIAYN exhibits an initial gain followed by stagnation or even

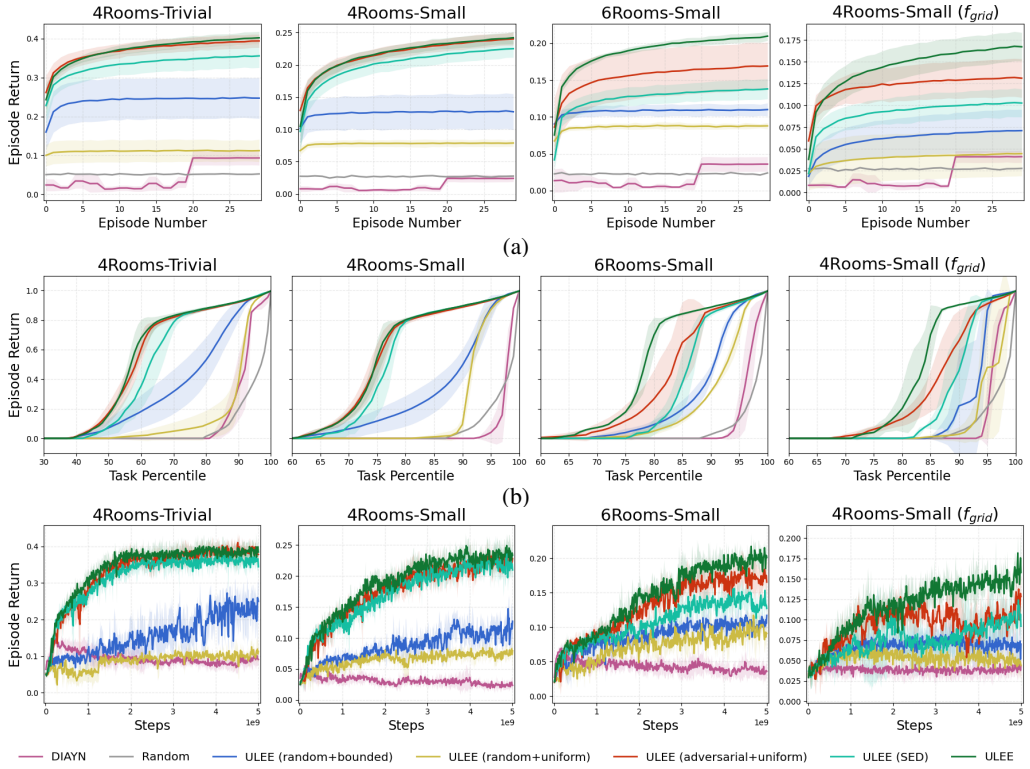


Figure 3: Evaluations on μ^{eval} tasks: (a) performance across episodes during task-specific adaptation, (b) few-shot performance by task percentile, (c) few-shot performance as pre-training on μ^{unsup} progresses. ULEE pre-training improves over baselines and ablations across all views. The legend in (c) applies to all panels. Reported steps for ULEE variants in (c) omit those from the Goal-search Policy, which adds 25%. Results are averaged over 4 seeds, with shaded regions indicating standard deviation.

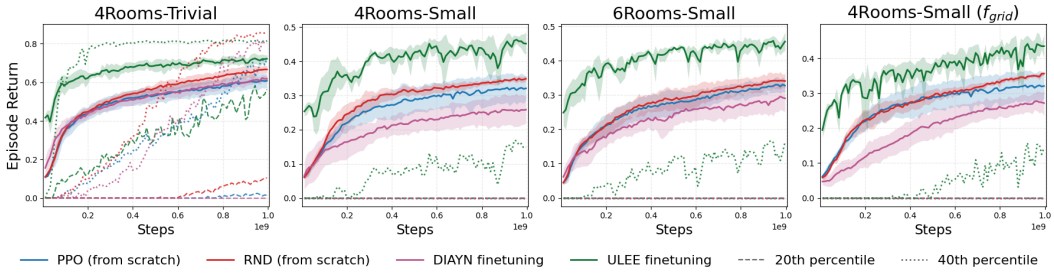


Figure 4: Mean, 40th, and 20th percentile returns on a fixed set of μ^{eval} tasks as learning on them progresses. Results are averaged over 4 seeds, with shaded regions indicating standard deviation.

decline as training continues. Lastly, across all views, using f_{counts} over f_{grid} during pre-training yields better results for ULEE, highlighting the value of f as a source of inductive bias.

4.3.3 Fine-tuning on Fixed Tasks

Beyond few-shot performance, we evaluate the utility of pre-training when longer adaptation budgets are available. We sample 2 048 environments from μ^{eval} and, on this fixed set, compare training from scratch to fine-tuning policies pre-trained with ULEE or DIAYN. For budgets up to 1 billion steps, ULEE consistently outperforms the other methods in mean, 40th percentile, and 20th per-

centile returns (Fig. 4). While DIAYN provides a slight initial advantage over training from scratch in some benchmarks, this benefit is short-lived. The results in Figure 4 underscore the difficulty of the 4Rooms-Small and 6Rooms-Small benchmarks. Even when targeting a fixed, small subset of tasks with extrinsic rewards, no method manages to solve 80% of tasks, and without ULEE, at least 40% remain unsolved.

4.3.4 Fine-tuning with Supervised Meta-Learning

We next investigate whether ULEE provides a strong initialization for supervised meta-reinforcement learning. Figure 5 compares meta-learning from scratch to starting from ULEE's Pre-trained Policy. For budgets of up to 5 billion steps of meta-learning on μ^{train}, ULEE initialization yields higher mean, 40th percentile, and 20th percentile returns on μ^{eval} tasks. In the 4Rooms-Small benchmark, using f_{counts} during the unsupervised phase leads to better results early in fine-tuning, but by 5 billion steps, the difference from f_{grid} is negligible.

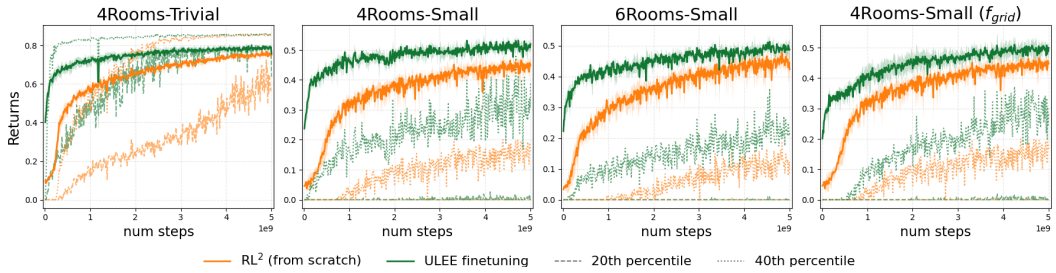


Figure 5: Mean, 40th, and 20th percentile returns on μ^{eval} tasks as meta-learning on μ^{train} progresses. Results are averaged over 4 seeds, with shaded regions indicating standard deviation.

4.3.5 Generalization to New Environment Structures

Previous sections evaluated generalization to novel goals and dynamics, but the environments' layout (grid size and number of rooms) remained unchanged between pre-training and evaluation. To assess generalization along this additional axis, we test the performance of methods on a variety of classical MiniGrid tasks after pre-training on 4Rooms-Small (Table 1). In few-shot evaluations, ULEE (f_{counts}) attains the best results, achieving a non-zero return on all environments. ULEE (f_{grid}) and DIAYN (f_{grid}) also improve substantially over a random policy.

Table 1: Mean return on MiniGrid tasks. ULEE and DIAYN methods were pre-trained on 4Rooms-Small and their performance is evaluated after 20 adaptation episodes. Results are averaged over 4 seeds, with standard deviations reported. The best-performing method is highlighted in bold.

task	Random	DIAYN	DIAYN (f_{grid})	ULEE	ULEE (f_{grid})
BlockedUnlockPickUp	0.02 ± 0.00	0.02 ± 0.01	0.03 ± 0.00	$\mathbf{0.43 \pm 0.19}$	0.03 ± 0.00
DoorKey-5x5	$\mathbf{0.05 \pm 0.00}$	0.00 ± 0.00	$\mathbf{0.08 \pm 0.14}$	$\mathbf{0.27 \pm 0.35}$	$\mathbf{0.08 \pm 0.02}$
DoorKey-8x8	$\mathbf{0.01 \pm 0.00}$	0.00 ± 0.00	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.24 \pm 0.24}$	$\mathbf{0.02 \pm 0.01}$
DoorKey-16x16	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.11 \pm 0.12}$	$\mathbf{0.00 \pm 0.01}$
Empty-8x8	0.11 ± 0.00	0.18 ± 0.19	$\mathbf{0.72 \pm 0.42}$	$\mathbf{0.64 \pm 0.22}$	$\mathbf{0.70 \pm 0.25}$
Empty-16x16	0.05 ± 0.00	$\mathbf{0.39 \pm 0.40}$	$\mathbf{0.73 \pm 0.42}$	$\mathbf{0.23 \pm 0.15}$	$\mathbf{0.53 \pm 0.36}$
EmptyRandom-8x8	0.24 ± 0.00	0.18 ± 0.23	$\mathbf{0.77 \pm 0.34}$	$\mathbf{0.49 \pm 0.30}$	$\mathbf{0.61 \pm 0.33}$
EmptyRandom-16x16	0.15 ± 0.00	$\mathbf{0.29 \pm 0.34}$	$\mathbf{0.59 \pm 0.32}$	$\mathbf{0.17 \pm 0.10}$	$\mathbf{0.47 \pm 0.31}$
FourRooms	0.02 ± 0.00	0.07 ± 0.01	$\mathbf{0.13 \pm 0.02}$	$\mathbf{0.14 \pm 0.03}$	$\mathbf{0.16 \pm 0.04}$
LockedRoom	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.01 \pm 0.00}$	0.00 ± 0.00
MemoryS8	0.18 ± 0.00	0.09 ± 0.05	0.41 ± 0.05	$\mathbf{0.47 \pm 0.10}$	$\mathbf{0.52 \pm 0.01}$
MemoryS16	0.22 ± 0.00	$\mathbf{0.31 \pm 0.20}$	$\mathbf{0.42 \pm 0.18}$	$\mathbf{0.51 \pm 0.04}$	$\mathbf{0.51 \pm 0.06}$
Unlock	0.03 ± 0.00	0.01 ± 0.01	0.10 ± 0.05	$\mathbf{0.75 \pm 0.16}$	0.02 ± 0.01
UnlockPickUp	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.68 \pm 0.15}$	0.00 ± 0.00

5 CONCLUSION

We present ULEE, an unsupervised meta-learning approach to induce a pre-trained policy with transferable capabilities. ULEE leverages an adversarial curriculum of self-imposed goals to train on tasks that are challenging but solvable for the policy under a given adaptation budget. After pre-training on reward-free grid worlds, we evaluate on novel, partially observable, procedurally generated environments. ULEE outperforms baselines and ablations in zero-shot and few-shot settings and provides a strong initialization for fine-tuning on both fixed tasks and curated meta-learning distributions. It demonstrates generalization to new goals, transition dynamics, and grid structures, and scales favorably with the size of the pre-training and adaptation budgets. Future work could refine or extend ULEE's components, for instance by introducing hierarchical structure into the meta-learned policy to address longer-horizon tasks. A complementary direction is to integrate vision-language models (VLMs) into the goal-proposal and reward-specification mechanisms to align pre-training with human-relevant tasks and potentially enhance applicability to real-world settings.

REFERENCES

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Andres Campero, Roberta Raileanu, Heinrich Küttler, Joshua B Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. Learning with amigo: Adversarially motivated intrinsic goals. *arXiv preprint arXiv:2006.12122*, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Nuttapong Chentanez, Andrew Barto, and Satinder Singh. Intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 17, 2004.

Cédric Colas, Pierre Fournier, Mohamed Chetouani, Olivier Sigaud, and Pierre-Yves Oudeyer. Curious: intrinsically motivated modular multi-goal reinforcement learning. In *International conference on machine learning*, pp. 1331–1340. PMLR, 2019.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pp. 482–495. PMLR, 2017.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pp. 1515–1528. PMLR, 2018.

Sébastien Forestier, Rémy Portelas, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *Journal of Machine Learning Research*, 23(152):1–41, 2022.

Jean-Baptiste Gaya, Laure Soulier, and Ludovic Denoyer. Learning a subspace of policies for online adaptation in reinforcement learning. *arXiv preprint arXiv:2110.05169*, 2021.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Unsupervised meta-learning for reinforcement learning. *arXiv preprint arXiv:1806.04640*, 2018.

Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pp. 2681–2691. PMLR, 2019.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Allan Jabri, Kyle Hsu, Abhishek Gupta, Ben Eysenbach, Sergey Levine, and Chelsea Finn. Unsupervised curricula for visual meta-reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pp. 4940–4950. PMLR, 2021.

Aya Kayal, Eduardo Pignatelli, and Laura Toni. The impact of intrinsic rewards on exploration in reinforcement learning. *Neural Computing and Applications*, pp. 1–35, 2025.

A.S. Klyubin, D. Polani, and C.L. Nehaniv. Empowerment: a universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pp. 128–135 Vol.1, 2005. doi: 10.1109/CEC.2005.1554676.

Grgur Kovač, Adrien Laversanne-Finot, and Pierre-Yves Oudeyer. Grimgep: learning progress for robust goal sampling in visual deep reinforcement learning. *IEEE Transactions on Cognitive and Developmental Systems*, 15(3):1396–1407, 2022.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 34:18459–18473, 2021.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pp. 1162–1176. PMLR, 2020.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 28, 2015.

Mirco Mutti, Mattia Mancassola, and Marcello Restelli. Unsupervised reinforcement learning in multiple environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7850–7858, 2022.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Advances in neural information processing systems*, 31, 2018.

Alexander Nikulin, Vladislav Kurenkov, Ilya Zisman, Artem Agarkov, Viacheslav Sinii, and Sergey Kolesnikov. Xland-minigrid: Scalable meta-reinforcement learning environments in jax. *Advances in Neural Information Processing Systems*, 37:43809–43835, 2024.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

OpenAI OpenAI, Matthias Plappert, Raul Sampedro, Tao Xu, Ilge Akkaya, Vineet Kosaraju, Peter Welinder, Ruben D'Sa, Arthur Petron, Henrique P d O Pinto, et al. Asymmetric self-play for automatic goal discovery in robotic manipulation. *arXiv preprint arXiv:2101.04882*, 2021.

Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pp. 7487–7498. PMLR, 2020.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International conference on machine learning*, pp. 5062–5071. PMLR, 2019.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

Sebastien Racaniere, Andrew K Lampinen, Adam Santoro, David P Reichert, Vlad Firoiu, and Timothy P Lillicrap. Automated curricula through setter-solver interactions. *arXiv preprint arXiv:1909.12892*, 2019.

Sai Rajeswar, Pietro Mazzaglia, Tim Verbelen, Alexandre Piché, Bart Dhoedt, Aaron Courville, and Alexandre Lacoste. Mastering the unsupervised reinforcement learning benchmark from pixels. In *International Conference on Machine Learning*, pp. 28598–28617. PMLR, 2023.

Jürgen Schmidhuber. Curious model-building control systems. In *Proc. international joint conference on neural networks*, pp. 1458–1463, 1991.

Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE transactions on autonomous mental development*, 2(3):230–247, 2010.

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

Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12686–12699, 2021.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International conference on machine learning*, pp. 8583–8592. PMLR, 2020.

Younggyo Seo, Kimin Lee, Stephen L James, and Pieter Abbeel. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, pp. 19561–19579. PMLR, 2022.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International conference on machine learning*, pp. 9870–9879. PMLR, 2021.

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.

Sainbayar Sukhbaatar, Emily Denton, Arthur Szlam, and Rob Fergus. Learning goal embeddings via self-play for hierarchical reinforcement learning. *arXiv preprint arXiv:1811.09083*, 2018.

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.

Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, et al. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*, 2021.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International conference on machine learning*, pp. 3540–3549. PMLR, 2017.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

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. *arXiv preprint arXiv:1901.01753*, 2019.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pp. 11920–11931. PMLR, 2021.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020a.

Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33:7648–7659, 2020b.

A APPENDIX

A.1 PSEUDOCODE

Algorithm 1: ULEE. Unsupervised Learning of Efficient Exploration

Initialize: Pre-trained Policy π, Goal-search Policy π_{gs}, Difficulty Predictor $\hat{d}_\phi$, and goal buffer B_g
while *training* **do**
  Sample a batch of environments $\{M_i\}_{i=1}^N \sim \mu^{\text{unsup}}$
  ▷ Search for candidate goals
  foreach M_i **do**
    Run π_{gs}; obtain candidate goals GC_{M_i} (Sec. 3.2.1) and environment information ξ_{M_i}
  end
  ▷ Sample goals of intermediate difficulty
  foreach M_i **do**
    Compute predicted difficulties $\hat{d}_\phi(g, \xi_{M_i})$ for all $g \in GC_{M_i}$
    Filter to target difficulty range: $S \leftarrow \{\, g \in GC_{M_i} : LB \le \hat{d}_\phi(g, \xi_{M_i}) \le UB \,\}$ (Eq. 4)
    if $S \neq \emptyset$ **then**
      Sample $g_{M_i} \sim \text{Unif}(S)$
    else
      Sample $g_{M_i} \sim \text{Unif}(GC_{M_i})$
    end
  end
  ▷ Train Pre-trained Policy
  $D_{\text{goals\_successes}} \leftarrow \emptyset$
  while *lifetimes not finished* **do**
    $D_{\text{update}} \leftarrow \emptyset$
    foreach M_i **do**
      advance π's lifetime on M_i under goal g_{M_i}; append transitions to D_{update}
      check for success $\{\exists\, t \in \{0, \dots, T-1\} :\ f(s_{t+1}) = g\}$ in completed episodes and update $D_{\text{goals\_successes}}$
    end
    update π on D_{update} to maximize meta-RL objective $\mathcal{J}(\pi)$ (Eq. 1)
  end
  Compute empirical difficulty estimates $\tilde{d}(g_{M_i})$ from $D_{\text{goals\_successes}}$ according to Eq. 2
  Push $\{(g_{M_i}, \xi_{M_i}, \tilde{d}(g_{M_i}))\}_{i=1}^N$ into buffer B_g (replace the N oldest tuples if full)
  ▷ Train Difficulty Predictor
  Update $\hat{d}_\phi$ on data from B_g to minimize $\mathcal{L}_{\text{DP}}(\phi)$ (Eq. 5)
  ▷ Train Goal-search Policy
  for *num_gs_updates* **do**
    $D_{\text{update}} \leftarrow \emptyset$
    foreach M_i **do**
      run π_{gs}; collect trajectories into D_{update}, setting $r_t^{gs} \leftarrow \hat{d}_\phi\big(f(s_t), \xi_{M_i}\big)$ (Eq. 3)
    end
    update π_{gs} on D_{update} to maximize $\mathcal{J}_{gs}(\pi_{gs}) = \mathbb{E}_{M, \pi_{gs}}\left[\sum_{t=0}^{T-1} \gamma^t r_t^{gs}\right]$
  end
end

A.2 EXPERIMENTAL DETAILS

A.2.1 ENVIRONMENTS AND GOAL MAPPING

Our benchmarks are derived from XLand-MiniGrid. This section extends the information given in Section 4.1; for full details, see Nikulin et al. (2024). Each environment offers six discrete primitive actions. There are 70 unique objects that can appear in the environments. Objects are identified by type (e.g., *key*, *ball*, *pyramid*) and color. The (extrinsic) task space spans 13 goal types (e.g., `AgentHoldGoal(a)` – whether agent holds a; `TileOnPositionGoal(a, x, y)` – whether a is on (x, y) position; `TileNearLeftGoal(a, b)` – whether b is one tile to the left

of a). The rule space spans 10 rule types (e.g., `AgentHoldRule(a)` $\rightarrow$ `c` – If agent holds a replaces it with c; `TileNearRightRule(a, b)` $\rightarrow$ `c` – If b is one tile to the right of a, replaces one with c and removes the other). Each grid cell is associated with a corresponding ID that identifies its kind (e.g., wall, red pyramid, empty space). f_{counts} maps states to a vector that records the number of occurrences of each ID. Thus, an intrinsic goal under f_{counts} specifies how many instances of each unique object should appear on the grid. For both intrinsic and extrinsic tasks, a reward is given only upon success. If the agent succeeds in time step t, it receives a reward of $1.0 - 0.9 \cdot (t/\text{max\_steps})$, where max_steps is the maximum number of steps the agent can take before the episode is considered a failure. We use early termination upon success. The agent's 5×5 observations are not occluded by walls.

Our configuration for benchmarks differs from XLand-MiniGrid's default behavior in two ways: (i) for 13x13 grids, we set the episode horizon (max_steps) to 256 rather than the default 509; (ii) we sample the initial state $s_0 \sim \rho_M$ once at the beginning of each lifetime and keep it fixed across all episodes within that lifetime. The default configuration resamples s_0 at the start of every episode.

Our evaluations on MiniGrid tasks in Section 4.3.5 use the default configurations of XLand-MiniGrid's pre-built MiniGrid environments.

A.2.2 Hardware and Seeds

Each algorithm is trained with 4 random seeds. For every training seed, we draw two additional independent seeds: one to construct the benchmark split between $\mu^{\text{train}}/\mu^{\text{unsup}}$ and μ^{eval}, and one to run the fine-tuning experiments (Sections 4.3.3 and 4.3.4). Hyperparameters for each method are held fixed across benchmarks and seeds. All experiments are executed on a single NVIDIA RTX 4090 GPU.

A.2.3 Algorithm Implementations and Hyperparameters

Architectures

Policies: Policies use a Gated Transformer-XL architecture with 2 blocks, hidden size 192, and 4 attention heads (head dimension 48). Gates use a bias of 2. The network attends to cached activations of up to 128 steps into the past, yielding an effective memory horizon of 256 (memory_len $\times$ num_blocks). During training, gradients are propagated for up to 64 steps. Two separate MLP heads, each with a size of $(256, 256)$, are used as the Actor and Critic. Each full policy network has $\approx 1.7M$ parameters. ReLU activations are used throughout. For meta-learned policies (ULEE's Pre-trained Policy and the RL^2 baseline), the per-step input is $\{o_t, d_t, a_{t-1}, r_{t-1}\}$, where $d_t \in \{0, 1\}$ indicates the beginning of a new episode (dummy a_{t-1} and r_{t-1} are used at the start of each lifetime). The meta-learner state is reset only upon encountering a new environment (at the start of a lifetime).

Difficulty Predictor and discriminator: ULEE's Difficulty Predictor and DIAYN's discriminator are MLPs of size $(256, 256)$ that operate on learned encodings of their inputs (see *Input encodings* below). Both have access to full state information during pre-training. DIAYN's discriminator predicts latent skills from encodings of visited states $q(z|f(s))$, where f coincides with ULEE's goal-mapping function and serves as a source of inductive bias. The Difficulty Predictor's inputs are $f_{\text{grid}}(s_g)$ (with $f(s_g) = g$) and $\xi_M = f_{\text{grid}}(s_0)$, where $s_0 \sim \rho_M$ (see 3.2.3). f_{grid} removes the agent's orientation and pocket information from the state data.

RND target and predictor: RND's target and predictor are MLPs with two hidden layers of size 256. They operate on encodings of observations. For benchmarks using f_{counts}, each observation is mapped to a vector that records the number of occurrences of each cell type; the MLP is applied directly to this vector and produces a 256-dimensional output. For benchmarks using f_{grid}, observations are first encoded with a small CNN (see *Input encodings* below); the MLP is then applied to this encoding with an output dimensionality of 64. RND is used as a baseline in Section 4.3.3, where agents are trained to solve $2\,048$ extrinsic tasks. Since a single policy is trained across all tasks, we also use a single shared target-predictor pair.

Input encodings: To process the symbolic grid observations and states, each cell's symbol is embedded via two learned embedding tables (shape and color). Then, a convolutional neural network

is applied, with the final output flattened. Additional inputs, when present, are concatenated to the flattened vector. Actions and skills have their own embedding tables. All learned embedding tables produce 16-dimensional vectors. For the Difficulty Predictor, the two input states are concatenated across the channel dimension before applying the CNN.

DIAYN skills: DIAYN is trained with 10 skills (following Gaya et al. (2021) and Kayal et al. (2025)). We add a fixed per-skill bias vector to the policy logits; these vectors are orthogonally initialized and scaled by a factor of 8. We find this addition important for learning distinguishable skills.

Training

The pre-training phase for ULEE and DIAYN spans 5 billion environment steps. We collect experience in parallel from batches of 2 048 environments and, after 5 120 steps per environment, resample a new batch from μ^{unsup}. ULEE updates its policy π every 256 steps per environment, while DIAYN updates every 512 steps. Policy updates for all algorithms use PPO with the Adam optimizer (hyperparameters in Table 2).

For ULEE, an additional 25% of steps (on top of the counts reported above) are allocated to the Goal-search Policy π_{gs}. For each newly sampled environment M, π_{gs} first executes two 256-step episodes to collect candidate goals. It adds one candidate to GC_M for every 15 states visited (Section 3.2.1). After the Pre-trained Policy and Difficulty Predictor are updated with data from M, π_{gs} executes three additional 256-step episodes during which it is trained to reach difficult goals (Section 3.2.1).

Task difficulty (Equation 2) is estimated empirically as the Pre-trained Policy's success rate over its last five episodes on that task. For bounded sampling, we use a difficulty lower bound LB of 0.1 and upper bound UB of 0.9. The goal buffer B_g used to train the Difficulty Predictor stores goals (together with their empirical difficulty estimates) from the last five batches of 2 048 intrinsic goals ($|B_g| = 5 \times 2048$). After the Pre-trained Policy π finishes training on a new batch and the corresponding goals are added to B_g, we train the Difficulty Predictor for 2 epochs over the entire buffer using a minibatch size of 256 and a learning rate of 1×10^{-4} (Sec. 3.2.3).

DIAYN's discriminator is updated every 512 steps per environment and is trained for a single epoch over all collected transitions with a learning rate of 3×10^{-4}.

For RND, we compute advantages for extrinsic and intrinsic rewards using the same γ and λ values as in Table 2. These are combined using an extrinsic-reward coefficient of 1 and an intrinsic-reward coefficient of 0.1 before policy updates. The predictor network is updated every 256 steps per environment and trained for a single epoch over all collected transitions. Its learning rate is 1×10^{-4} for benchmarks that use f_{counts} and 1×10^{-5} for benchmarks that use f_{grid}.

A.2.4 Experimental Procedures

Exploration: For each seed in Figure 2, results are obtained from evaluation on a set of 16 384 environments sampled from μ^{eval}. Methods interact for 20 sequential episodes with each environment. A goal is considered reached at episode j if it was achieved in any episode $\leq j$. For DIAYN, we run two episodes with each skill, randomizing their order. Each skill is used at least once before any repeat.

Fast adaptation: For each seed in Figures 3a and 3b, results are obtained from evaluation on a set of 16 384 environments sampled from μ^{eval}. Methods interact for 30 sequential episodes with each environment. For DIAYN, we run two episodes conditioning on each skill, followed by ten episodes with the best-performing skill. Figure 3a shows results across all 30 episodes, whereas Figure 3b reports an aggregate performance over the last 10. Figure 3c tracks performance on μ^{eval} throughout pre-training on μ^{unsup}; at each evaluation point, we sample a fresh set of 1 024 environments from μ^{eval} and measure the mean performance over the last 5 episodes of 25-episode interactions.

Fine-tuning on fixed tasks: For Section 4.3.3, each seed is fine-tuned on a fixed set of 2 048 environments sampled from μ^{eval}. Figure 4 tracks performance on this fixed set throughout training, with performance measured as the mean over the last 10 episodes of 30-episode interactions. For DIAYN, before fine-tuning we run 10 episodes of each skill per environment. In each environment,

Table 2: Hyperparameters for policy updates. All policies are updated using PPO with the Adam optimizer. Entropy coefficients are listed separately as they vary across different methods.

Hyperparameter	**Value**
Learning rate	2×10^{-4}
Adam ϵ	1×10^{-5}
Discount factor γ	0.99
Update epochs	1
Number of minibatches	16
Clipping ϵ	0.2
GAE λ	0.95
Value loss coefficient	0.5
Max gradient norm	0.5
Entropy coefficient:	
Goal-search Policy	0.01
Pre-trained Policy	0.005
DIAYN	0.03
DIAYN when fine-tuning	0.01
PPO (from scratch)	0.005
RND	0.005
RL^2 (from scratch)	0.005

we select the best-performing skill and keep it fixed during fine-tuning. These preparatory steps are excluded from the figure.

Fine-tuning with supervised meta-learning: Figure 5 reports performance on μ^{eval} throughout meta-learning on μ^{train}. At each evaluation point, we sample a fresh set of $1\,024$ environments from μ^{eval} and measure the mean performance over the last 5 episodes of 25-episode interactions.

MiniGrid: In Section 4.3.5, results for each seed on each MiniGrid environment are averaged over $2\,048$ runs; each run consists of 30 sequential episodes, and we report the performance over the last 10 episodes.

A.2.5 Hyperparameter Selection and Stability

As is common in meta-learning and unsupervised pre-training, conducting an exhaustive hyperparameter search was infeasible due to the length and computational cost of each run. Our hyperparameter studies therefore aimed to: (1) assess whether ULEE requires careful tuning in order to learn, and (2) identify a reasonable, stable configuration rather than optimize for the best possible values. Given this constrained scope, we do not present quantitative comparisons or prescriptive recommendations on the choice of hyperparameters. Instead, this section documents the explorations that were performed and the observations that informed our final choices.

Our investigations found that ULEE does not require careful tuning to learn effectively. Across all configurations we tried, we observed no signs of collapse, and the results reported in the main paper were obtained after limited tuning. All tuning and stability-analysis experiments were run on seeds that were different from those used in the final experimental results reported in Section 4.

When selecting hyperparameters for each method, we focused mainly on the following criteria:

- DIAYN: We monitored performance on tasks sampled from μ^{train} throughout unsupervised pre-training on reward-free environments from μ^{unsup}. We additionally inspected skill distinguishability (via per-skill heatmaps on empty environments and via the discriminator loss) and the entropy of each skill. When searching for fine-tuning hyperparameters (Section 4.3.3), we evaluated performance on fixed extrinsic-reward tasks as fine-tuning progressed.
- RL^2 (from scratch): We monitored performance on tasks sampled from μ^{train} during supervised meta-learning on tasks from μ^{train}.

- PPO (from scratch): We monitored performance on a set of $2\,048$ tasks sampled from μ^{train} while training on them.
- RND: We monitored performance on a set of $2\,048$ tasks sampled from μ^{train} while training on them.
- ULEE: We monitored performance on tasks sampled from μ^{train} throughout unsupervised pre-training on reward-free environments from μ^{unsup}. Importantly, during the selection process, we did *not* inspect any of the evaluation metrics presented in Figures 2, 3, 4, and 5. In particular, we did not test or select hyperparameters based on downstream fine-tuning performance. We also did not tune any hyperparameters specific to fine-tuning policies pre-trained with ULEE; for those experiments we reused the values employed during pre-training.

Below we summarize which hyperparameters were varied and which were kept fixed. For those that were varied, our tuning process was limited in several ways: at most three hyperparameters were varied at a time, each combination was tested with only 1–2 seeds, experiments were run exclusively on the 4Rooms-Small benchmark, and runs were shorter than those used in the final results.

Network architectures: We did not perform any hyperparameter tuning on network architectures.

Number of steps and parallel environments: We did not tune the number of parallel environments, the split of $\{$goal, rules, initial objects$\}$ tuples between μ^{train} and μ^{eval}, the maximum number of steps per episode, the number of steps per environment before resampling, or the total number of training steps.

Policy updates: We tested variations of the entropy coefficient, discount factor γ, and learning rate. All other hyperparameters were fixed in advance. The values explored were:

- DIAYN: entropy coefficient $\{0.003, 0.005, 0.01, 0.02, 0.03, 0.05, 0.1\}$, learning rate $\{3 \times 10^{-5}, 1 \times 10^{-4}, 2 \times 10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}\}$.
- DIAYN (fine-tuning): entropy coefficient $\{0.005, 0.01, 0.03\}$, learning rate $\{2 \times 10^{-4}\}$.
- RL^2: entropy coefficient $\{0.001, 0.003, 0.005, 0.007, 0.01, 0.03, 0.05\}$, learning rate $\{1 \times 10^{-4}, 2 \times 10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}\}$, discount factor $\{0.99, 0.995, 0.999\}$.
- PPO: entropy coefficient $\{0.003, 0.005, 0.01\}$, learning rate $\{3 \times 10^{-5}, 1 \times 10^{-4}, 2 \times 10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}\}$.
- RND: entropy coefficient $\{0.003, 0.005, 0.01\}$, learning rate $\{3 \times 10^{-5}, 2 \times 10^{-4}, 5 \times 10^{-4}\}$.
- ULEE Pre-trained Policy: entropy coefficient $\{0.003, 0.005, 0.01, 0.03\}$, learning rate $\{3 \times 10^{-5}, 1 \times 10^{-4}, 2 \times 10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}\}$.
- ULEE Goal-search Policy: entropy coefficient $\{0.003, 0.005, 0.01, 0.03\}$, learning rate $\{2 \times 10^{-4}\}$.

We found that learning rates between 1×10^{-4} and 3×10^{-4} worked well across all methods. Using discount factors of 0.995 and 0.999 in RL^2 strongly hindered performance.

DIAYN-specific hyperparameters: We varied the dimensionality of skill embeddings $\{16, 128\}$, the discriminator learning rate $\{3 \times 10^{-5}, 1 \times 10^{-4}, 3 \times 10^{-4}\}$, and the number of steps per environment per update $\{256, 512, 1024\}$. We also experimented with adding a fixed per-skill bias vector to the policy logits (Appendix A.2.3). This proved important for obtaining distinguishable skills in setups where the agent's and objects' initial positions are randomized. We explored two initializations for the bias vectors $\{$orthogonal, random directions in the unit sphere$\}$ and several scale factors $\{1, 5, 8, 10, 15, 20\}$. Both initialization schemes were effective and scale factors between 5 and 10 performed best, though smaller and larger values still helped. Additionally, we tested disabling the policy updates for the first 10 batches of environments, which did not help, and varying the initialization of the discriminator final layer, which improved stability.

RND-specific hyperparameters: We varied the dimensionality of the embedding produced by the target network $\{32, 64, 256\}$, the intrinsic-reward coefficient $\{0.02, 0.05, 0.1, 0.3, 0.5, 1, 2\}$, and the predictor-network learning rate $\{1 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}, 3 \times 10^{-4}\}$.

ULEE-specific hyperparameters:

Difficulty Predictor. We varied its learning rate $\{3 \times 10^{-5}, 1 \times 10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}\}$ and the number of batches of intrinsic goals stored in the training buffer $\{1, 3, 5\}$. ULEE learned under all configurations. Differences were generally modest; a learning rate of 5×10^{-4} produced the largest performance drop.

Goal-search Policy and entropies. As noted earlier, we varied the entropy coefficient of the Goal-search Policy and Pre-trained Policy updates. For the Goal-search Policy, we observed little effect on performance. For the Pre-trained Policy, the different values influenced the shape of pre-training curves but did not reveal a clear best choice. We also varied the number of Goal-search Policy updates per batch of environments (`num_gs_updates` in Algorithm 1) $\{3, 6\}$ and observed similar results.

Goal sampling strategy. We tested three sampling schemes: (1) uniform sampling from goals within difficulty bounds (Section 3.2.2), with bounds $\{(0.1, 0.9), (0.3, 0.7)\}$; (2) sampling a target difficulty for each environment M from a Gaussian with mean in $\{0.4, 0.6, 0.8\}$ and standard deviation in $\{0.2, 0.4\}$ and picking the goal from GC_M whose predicted difficulty is closer to that value; (3) assigning weights to candidate goals in GC_M (based on their estimated difficulty) using a Gaussian density function with mean in $\{0.5, 0.6\}$ and standard deviation in $\{0.2, 0.4\}$ and sampling in accordance with these weights. ULEE learned under all schemes and hyperparameters tried. The best performance was obtained with scheme 2 and 3 using a Gaussian mean of 0.6 and standard deviation of 0.2, and with scheme 1 using a lower bound LB of 0.1 and upper bound UB of 0.9.

Steps per update. We varied the number of steps per environment per update $\{256, 512\}$ and observed faster learning with the smaller value.

No search was performed over hyperparameters not mentioned in this section. As noted earlier, our hyperparameter search was limited in scope: the goal was to identify reasonable values and evaluate whether ULEE exhibited failure modes or strong sensitivity to particular hyperparameters. Thus, the final set of hyperparameters should not be interpreted as either optimal or deeply tuned. Ablations used the same hyperparameters selected for ULEE.

A.3 Learning Progress

A complementary line of work in automatic curriculum generation employs *learning progress* (LP), rather than difficulty, as the guiding metric for goal selection (Baranes & Oudeyer, 2013; Colas et al., 2019; Forestier et al., 2022; Kovač et al., 2022). Learning progress is estimated by tracking an agent's performance on tasks over time and quantifying its rate of change. Tasks exhibiting the largest improvements (or deteriorations) are prioritized. Typically, LP is computed from changes in the policy's *immediate* performance as training proceeds, without allowing for any task-specific adaptation. Analogous to difficulty-based methods, LP can be adapted so as to measure changes in *post-adaptation* performance. Let τ denote training time and π_τ the policy after training for τ. If $\tilde{d}(g; \pi, M)$ denotes a post-adaptation performance metric on task (g, M), e.g., an empirical estimate of Eq. 2, then a possible LP-style goal desirability score is $LP_{\text{post}}(g, M; \tau) = |\tilde{d}(g; \pi_\tau, M) - \tilde{d}(g; \pi_{\tau-\Delta\tau}, M)|$, where $\Delta\tau > 0$ is the measurement window. Investigating post-adaptation learning progress and its interplay with post-adaptation difficulty-based curricula and unsupervised goal generation is a promising direction for future work.

A.4 Use of Large Language Models

Large language models (LLMs) were used solely as coding assistants and for polishing the paper's writing. They were not used to generate, develop, or analyze the underlying scientific content or technical contributions.