SUN: REACHING FOR NOVELTY IN REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Exploration in reinforcement learning (RL) remains a fundamental challenge. Recent goal-conditioned RL strategies (which select goals to encourage broader state coverage) have shown promising results, but none scores a goal by novelty and reachability jointly: the two signals are traded off by hand, applied in sequence, or one is neglected outright. In this paper, we introduce a reachability-aware goal-selection framework that explicitly integrates these two aspects, and that can be seamlessly incorporated into any off-policy RL algorithm. To this aim, we propose *SUccessor-to-Novelty (SUN)*, an indicator derived from successor value functions to identify goals that are both novel and reachable. We prove that SUN recovers count-based bonuses in the limit, bounds short-horizon hitting probabilities, and provably rejects unreachable goals. We further present an adaptive goal-selection strategy that leverages these properties, along with an accurate yet lightweight pseudocount to avoid the overhead of classic methods. We back up all our claims with thorough benchmarks: SUN consistently outperforms state-of-the-art methods in standard and novel environments with unreachable or hard-to-reach states, irreversible transitions, obstacles, mazes, and unbounded spaces.

1 INTRODUCTION

Exploration is fundamental to reinforcement learning (RL): without effective exploration, agents collect uninformative data and fail to learn. Classical dithering schemes, such as ε-greedy and entropy regularization, ignore environment structure and are sample-inefficient. Provably efficient algorithms (Auer et al., 2002; Strehl and Littman, 2008; Jaksch et al., 2010) offer strong guarantees but do not scale to large state spaces. Intrinsic motivation methods (Pathak et al., 2017; Burda et al., 2019; Parisi et al., 2021) require careful tuning and are non-stationary by construction: as the agent explores, the intrinsic reward shifts beneath the policy trained on it, destabilizing learning (Burda et al., 2019).

A more recent family casts exploration as *goal-conditioned RL* (GCRL) (Liu et al., 2022; Colas et al., 2022), where the agent follows a goal-conditioned policy trained on a stationary goal-reaching objective. Different *goal-selection* mechanisms lead to different exploration strategies, but most of the existing work captures only half the picture. Density-based methods such as MEGA (Pitis et al., 2020), Skew-Fit (Pong et al., 2020), GoalGAN (Florensa et al., 2018), and Hindsight Goal Generation (Ren et al., 2019) score goals by novelty, committing to rare goals that may be unreachable. Conversely, methods based on distances or success probabilities (Schaul et al., 2015; Hartikainen et al., 2020) optimize reachability alone, neglecting rare but achievable goals. Neither extreme captures the right intuition: *a useful exploration goal is one that is novel **and** reachable*. Figure 1 summarizes this problem. In this paper, we address this gap with the following contributions.

(1) We present the *SUccessor-to-Novelty (SUN)* indicator to balance *novelty* and *reachability*. SUN is compatible with any off-policy RL algorithm; in this paper, we instantiate it with DQN (Mnih et al., 2015) and TD3 (Fujimoto et al., 2018).

(2) We present an accurate yet lightweight pseudocount that avoids the overhead of density-based methods, enabling efficient exploration with $\mathcal{O}(1)$ query cost.

(3) We show that SUN is the value of a goal-conditioned count-bonus reward, gives a closed-form lower bound on the short-horizon hitting probability, and provably suppresses unreachable goals.

(4) We introduce new benchmarking environments with unreachable states and irreversible transitions that directly stress-test reachability-aware exploration, and show that SUN consistently outperforms state-of-the-art methods on these and standard benchmarks.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

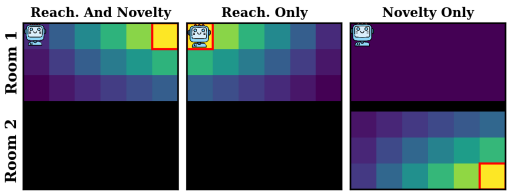


Figure 1: Reachability or novelty are not enough. At every episode, the agent spawns in one of two isolated rooms. After exploring for some time, the second room has been rarely visited due to its lower spawning rate. Heatmaps show the score assigned to each tile by different goal-selection scores when the agent is in the top-left corner (red boxes mark the selected goal). *Novelty alone* (e.g., visit counts inverse) picks tiles in the second room, which the agent cannot reach. *Reachability alone* (e.g., distance) picks the agent's current tile, leading to no exploration. Only *novelty and reachability combined* selects the least-visited tile *within reach*. While simple, this example shows the importance of considering both reachability and novelty in exploration, and raises the central question of this paper: how to encode, learn, and combine reachability and novelty? Our SUN indicator provides principled answers.

Together, these contributions establish SUN as a principled and practical solution to the long-standing tension between novelty and reachability in exploration.

SUN exploration fits in the field of reward-free exploration and goal-conditioned RL, and is especially close to the work of Tarbouriech et al. (2022) (AdaGoal) and Diaz-Bone et al. (2025) (DISCOVER) in its use of successor value functions to drive exploration. Both also balance novelty and reachability, but estimate novelty through critic-ensemble disagreement, which is computationally expensive and, as we show, leaves both methods poorly calibrated between the two signals. SUN sidesteps these issues with a lightweight pseudocount and a novel goal-selection strategy (Section 3). Across all our benchmarks, it consistently and substantially outperforms both methods.

2 Problem Setting

Optimal exploration. A *reward-free* Markov Decision Process (MDP) is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, p_0 \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}(s' \mid s, a)$ is the transition function, and p_0 is the initial state distribution. The objective is to explore the state space "optimally" without any task-specific reward. Two main lines of work formalize this notion of optimality differently.

The first line targets the state-visitation distribution: the goal is to learn a policy whose induced distribution maximizes a desired criterion, typically the entropy (Hazan et al., 2019; Lee et al., 2019; Mutti et al., 2021; Zhang et al., 2021; Jain et al., 2023; Adamczyk et al., 2026). A maximum-entropy state-visitation distribution corresponds to uniform coverage of the state space, and provably efficient algorithms exist for this objective in tabular MDPs.

The second line frames exploration as goal-conditioned RL (GCRL) and the agent learns goal-conditioned policies $\pi(a \mid s, g)$ (Lim and Auer, 2012; Tarbouriech et al., 2020; 2022). The goal $g \in \mathcal{G}$ may be a subset of the state, of the joint state-action, or of a learned representation thereof.[1] A goal g is said to be *reachable* from a reference state s_0 if there exists a policy π that reaches g from s_0 in bounded expected time. The objective is then to learn policies that can visit every goal reachable given different reference states. This formulation directly captures the intuition that exploration should focus on states the agent can actually reach, but does not specify a target visitation distribution.

Both objectives are principled but solving them exactly requires machinery — e.g., Frank-Wolfe schemes for max-entropy (Hazan et al., 2019), PAC-style algorithms for reachable coverage (Tarbouriech et al., 2020; 2022) — that does not scale to deep RL. Practical methods therefore approximate these objectives with greedy or local heuristics: thanks to careful goal-selection mechanisms, following $\pi(a \mid s, g)$ induces a state-visitation distribution with broad and uniform state coverage. Our work follows this pragmatic line: we design a goal-selection rule that, at each step, prefers goals that are both underrepresented in the agent's current visitation distribution *and* reachable.

Successor Value Functions. In the GCRL literature, reachability is often encoded with the *successor value function (SVF)* (Dayan, 1993; Kulkarni et al., 2016; Blier et al., 2021; Eysenbach et al., 2022; Zheng et al., 2024), which generalizes the value function and represents the cumulative γ-discounted

[1] The goal space is environment-dependent. For example, discrete actions may highlight relevant dynamics (e.g., "pick" or "push" may terminate the episode) and exploration should *explicitly* consider them. On the contrary, the state alone may be sufficient if it carries all information (e.g., agent pose in control tasks).

occurrence of a goal g under a policy π, i.e.,

$$V^\pi(s_t, g) = \mathbb{E}\left[\sum_{k=t}^{\infty} \gamma^{k-t} \mathbf{1}_{\{s_k=g\}} \,\middle|\, \pi, \mathcal{P}, s_t\right], \tag{1}$$

where $\gamma \in [0, 1)$ and $\mathbf{1}_{\{s_k=g\}}$ is the reward function returning 1 if $s_k = g$ and 0 otherwise. The state-action analogue $Q^\pi(s_t, a_t, g)$ is defined likewise, and both admit a Bellman recursion as in standard value functions, with $V^\pi(s_t, g) = \max_a Q^\pi(s_t, a, g)$. Similarly to classic value functions, SVFs are often approximated with parameterized functions $V^\theta(s_t, g)$, and training them is a well-studied problem. In this paper, we rely on Hindsight Experience Replay (HER) (Andrychowicz et al., 2017).

In GCRL, once the goal is given (e.g., a desired robot pose or an environment coordinate), greedily following the SVF leads the agent to it, as the value increases the fewer steps are needed to reach the goal.[2] This same mechanism extends naturally to reward-free exploration: *select a goal appropriately* — unlike in GCRL, the goal is not given by the task — and then follow the SVF to reach it. The goal-selection is what determines whether exploration is optimal: a well-designed mechanism would guarantee coverage and uniformity over the goal space $\mathcal{G} \subseteq \mathcal{S} \times \mathcal{A}$.

With these tools in hand, our method must address three concrete subproblems. First, how to combine the reachability and novelty signals into a single indicator. Second, how to design an effective goal-selection strategy given the above indicator. Third, how to compute a novelty signal that is both accurate and cheap enough to query at scale.

3 Exploration via SUN

We present our answer to the three subproblems above: **SUN** (**SU**ccessor-to-**N**ovelty), a goal-selection indicator that combines an SVF-based reachability signal with a novelty signal in a single score.

$$\mathrm{SUN}(g \,|\, s) \triangleq V^\pi(s, g) \cdot \nu(g), \qquad g_t = \arg\max_{g \in \mathcal{G}} \mathrm{SUN}(g \,|\, s_t), \tag{2}$$

where $V^\pi(s_t, g)$ is the SVF estimating reachability of g from s_t under the goal-conditioned policy, and $\nu(g)$ is a novelty signal. Computing the $\arg\max$ in Eq. (2) is not feasible in continuous or large goal spaces, so we restrict it to a finite candidate set $\mathcal{C}_t \subset \mathcal{G}$ sampled from a replay buffer. This choice pairs naturally with off-policy algorithms, which already maintain a buffer for training.

The rest of this section is organized as follows. Section 3.1 presents properties that justify the indicator; Section 3.2 describes a novel goal-selection strategy that leverages these properties; Section 3.3 introduces our novelty estimator; and Section 3.4 summarizes SUN and its relation to prior work.

3.1 The Indicator and Its Properties

We informally describe three properties of SUN that justify Eq. (2) and that we will invoke in subsequent sections; proofs are in Appendix A.

(a) *Count-bonus equivalence.* If $\nu(g) = 1/n_g$ where n_g is the goal visit count, SUN equals the value function of a reward inversely proportional to n_g, i.e., $\mathbb{E}[\sum_{k=t} \gamma^{k-t} \mathbf{1}_{\{s_k=g\}}/n_g]$. Thus SUN is not simply a product of two signals, but the value of a count-bonus objective.

(b) *Reachability guarantee.* The reachability factor $V^\pi(s, g)$ controls hitting time: a high SVF value implies a high probability of reaching g within a short horizon, $\Pr_{\pi_g}[\tau_g \le n] \ge V^{\pi_g}(s, g) - \gamma^{n+1}$, where τ_g is the hitting time. The horizon scales as $\mathcal{O}(\log(1/V)/(1-\gamma))$.

(c) *Unreachable goals are suppressed.* If no policy in the agent's class can reach g, then $V^\pi(s, g) = 0$ for all such policies, and $\mathrm{SUN}(g \,|\, s) = 0$ regardless of $\nu(g)$. This is the formal counterpart of Figure 1: novelty alone selects unreachable goals, while SUN does not.

Why not an additive indicator? Common exploration strategies combine reachability and novelty *additively* (Diaz-Bone et al., 2025). Indeed, SUN's indicator could equally be defined additively as $V^\pi(s, g) + \nu(g)$, which admits standard UCB-style confidence bounds and PAC guarantees when ν is based on visit counts (see Appendix K). However, additive formulations are sensitive to the relative scale of the two terms and typically require a tuning coefficient to balance them, especially if the

[2] This holds because goals are *terminal*: reaching g ends the episode, so Eq. (1) reduces to $V^\pi(s_t, g) = \mathbb{E}_\pi[\gamma^{\tau_g}] \in [0, 1]$, where τ_g is the hitting time of g. The SVF is thus a discounted reachability score.

SVF is approximated as in V^θ. The multiplicative form removes the need to *calibrate* the two terms against each other: they share a common "zero" (an unreachable or already-saturated goal scores zero on either factor and is rejected regardless of the other) and a common, known scale (both are non-negative and bounded by one). The trade-off is not thereby eliminated — in log-space it is set by $\kappa = -\log \gamma$ (Appendix A.5) — but it is fixed and inherited from the discount used for value learning, rather than being a free coefficient that must be re-tuned whenever the scale of the novelty signal changes (see Section 3.4). In Section 4.2 we compare SUN against an additive UCB-style variant and show that the multiplicative indicator performs significantly better.

3.2 When Should The Agent Select A Goal? Adaptive Goal-Selection Strategy

If V^θ were exact, acting greedily with respect to it would be optimal — the best goal would be selected and reached in finite time (Appendix A.7). Thus, *episodic goal-selection* — selecting the goal at the start of an episode and keeping it fixed until reached — would be optimal. However, V^θ is learned and approximate, and the agent may commit to unreachable goals, potentially not exploring at all. Similarly, under stochastic transitions, the agent may suddenly find itself in states where the previously-selected goal is no longer reachable. The opposite strategy, *per-step goal-selection*, compares the current goal against a fresh candidate set at every timestep to find a potentially better one. This can prevent commitment to unreachable goals, e.g., after a wrong action or a noisy transition. However, this strategy can be too unstable: as V^θ is being learned, goal values shift quickly and the agent may pick different goals at every timestep, acting near-randomly.

For these reasons, we propose a novel *adaptive* strategy, inspired by the theoretical properties of the SVF. Under deterministic dynamics, the true value at the current state should be monotonically non-decreasing along the trajectory toward the selected goal: as the agent moves closer, $V^\pi(s_t, g)$ grows. Thus, a drop in $V^\theta(s_t, g)$ signals that the goal is either unreachable from the current state, or that the *approximate* SVF was inaccurate at the time of selection — in either case, the goal is no longer a reliable target. Concretely, at each step t we compare the current value against the value at selection time t_{sel}: if $V^\theta(s_t, g) < V^\theta(s_{t_{\text{sel}}}, g)$, the current goal is discarded and a new one is selected from a freshly sampled candidate set; otherwise, the current goal is kept.[3]

This adaptive strategy preserves the stability of episodic commitment, and inherits per-step reselection's ability to escape bad commitments — but only if the SVF changes frequently (because it is still being learned) or if it signals that something has gone wrong (e.g., due to environment noise). Section 4.2 empirically validates our strategy.

3.3 Novelty Via Lightweight Pseudocounts

SUN combines two signals: reachability and novelty. Reachability is handled by learning SVFs with HER (Andrychowicz et al., 2017) (see Appendix C). Novelty $\nu(g)$ in Eq. (2) can be instantiated in many ways (Pathak et al., 2017; Burda et al., 2019). A principled choice is visit counts or density estimates (Bellemare et al., 2016; Tang et al., 2017), so that rarely-visited goals receive a high novelty score: $\nu(g) = 1/n_g$, where n_g is the number of times g has been visited. In continuous spaces n_g cannot be tracked exactly and must be approximated by a pseudocount. Since we query $\nu(g)$ against many candidate goals $\mathcal{C}_t$ multiple times per episode, the pseudocount must be lightweight to compute — standard approaches such as kernel density estimation (KDE) or neural density models do not satisfy this requirement. We instead propose a pseudocount that amortizes its cost into buffer insertion: counts are precomputed and stored alongside each buffer entry, making queries cheap.

For each buffer entry at index i, we store a count n_i of entries within a neighborhood of radius ρ of s_i in standardized feature space. When a new sample is inserted, we compute its count (number of neighbors plus one for itself) and increment the neighbors' counts. Figure 2 illustrates the procedure;

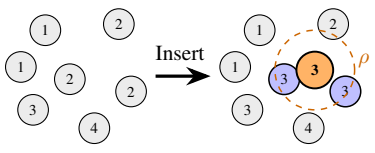


Figure 2: **Pseudocount via replay buffer neighbors.** Each buffer entry stores a count of its neighbors within radius ρ (*in standardized feature space*). When a new sample is inserted, its count is set to the number of neighbors within ρ plus one (for itself), and each neighbor's count is incremented.

[3]Note that under exact V^θ and deterministic dynamics, all three strategies coincide (Appendix A.7).

Appendix B gives the standardization details. This is a fixed-radius nearest-neighbor density estimator with the cost moved from query time to insertion time. Its benefits are the following.

- The radius ρ is a single hyperparameter: standardization makes one scalar meaningful across features, which would otherwise each need their own radius.
- By incrementing the count of every neighbor of the new sample, the stored counts are maintained across the entire buffer without ever recomputing them from scratch. At query time, the novelty of a candidate goal g is read directly from the replay buffer: $\nu(g) = 1/n_g$. This has cost $\mathcal{O}(1)$.
- The insertion cost is $\mathcal{O}(N \cdot M)$ in the buffer size N and goal dimensionality M. Classic KDE costs $\mathcal{O}(N \cdot M \cdot B)$ per step for B candidate goals, and neural density models can be even more expensive.

When goals may be reselected at any step, classic methods are prohibitively expensive while our pseudocount remains tractable. By moving the cost from query to insertion time and standardizing across features, we obtain a novelty estimator that is accurate yet lightweight. We validate its accuracy in Appendix H and report wall-clock costs in Appendix E.

3.4 Summary and Related Work

Algorithm 1: SUN Exploration (During One Episode)

Function `SelectGoal`$(s_t, \mathcal{D})$**:**
  $s_{t_{\text{sel}}} \leftarrow s_t$
  $\mathcal{C}_t \sim \mathcal{D}$
  return $\arg\max_{g \in \mathcal{C}_t} \text{SUN}(g \mid s_t)$

for $t = 0 \dots T$ **do**
  `begin` $\leftarrow t = 0$
  `reached` $\leftarrow \|s_t - g_t\| < \eta$
  `adapt` $\leftarrow V^\theta(s_t, g_t) < V^\theta(s_{t_{\text{sel}}}, g_t)$
  if *begin* **or** *reached* **or** *adapt* **then**
    $g_t \leftarrow$ `SelectGoal`$(s_t, \mathcal{D})$
  else
    $g_t \leftarrow g_{t-1}$
  $a_t \sim \pi(\cdot \mid s_t, g_t)$
  $s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t)$
  `Update`$(\mathcal{D}, s_t, a_t, s_{t+1})$

Algorithm 1 summarizes SUN exploration. At the beginning of an episode, the agent selects the goal g_0 according to Eq. (2), with candidates (and their pseudocounts) sampled from the replay buffer. The initial state s_0 is saved as $s_{t_{\text{sel}}}$ (state at selection time). At every timestep, if $V^\theta(s_t, g_t) < V^\theta(s_{t_{\text{sel}}}, g_t)$ or if the agent has reached the current goal, a new goal g_t is selected from a fresh batch of candidates and $s_{t_{\text{sel}}}$ is updated; if not, g_t is kept. Then, the agent acts to explore with $a_t \sim \pi(\cdot \mid s_t, g_t)$, the new sample is inserted into the replay buffer, and pseudocounts are updated. The policy and the SVF are trained with any off-policy algorithm (Appendix C).

This scheme can be applied to classic reward-driven RL: the goal-conditioned policy drives exploration to collect environment rewards, and task-specific value function and policy are trained off-policy.

AdaGoal and DISCOVER. Throughout Sections 2–3, we have discussed how SUN relates to prior work along several axes: reward-free exploration (Hazan et al., 2019; Tarbouriech et al., 2020; 2022), GCRL (Schaul et al., 2015), hindsight relabeling (Andrychowicz et al., 2017; Eysenbach et al., 2022; Zheng et al., 2024), and count- or density-based novelty (Bellemare et al., 2016; Pong et al., 2020; Pitis et al., 2020; Burda et al., 2019). Here, we focus on the two methods closest in spirit to SUN: AdaGoal (Tarbouriech et al., 2022) and DISCOVER (Diaz-Bone et al., 2025). Both select goals using an ensemble of SVFs $\{V^{\theta_1}, \dots, V^{\theta_K}\}$, with mean $\mu(s, g)$ and standard deviation $\sigma(s, g)$. Formally,

$$\underbrace{g_t = \arg\max_{g \in \mathcal{C}_t} V^\theta(s_t, g) \,/\, n_g}_{\text{SUN}} \qquad \underbrace{g_t = \arg\max_{g \in \mathcal{C}_0} \mu(s_0, g) + \beta\, \sigma(s_0, g)}_{\text{DISCOVER}^4} \qquad \underbrace{g_t = \arg\max_{g \in \mathcal{C}_0} \sigma(s_0, g)}_{\text{AdaGoal}^5}$$

The mean μ serves as a reachability signal, and the disagreement σ as an epistemic-uncertainty signal (goals on which the ensemble disagrees are those the agent cannot reliably reach). AdaGoal explicitly frames this as "selecting uncertain goals" and proves PAC guarantees; DISCOVER frames the same quantity as "novelty" since uncertainty correlates with under-exploration.

Two main differences distinguish SUN from this line of work. *First*, AdaGoal and DISCOVER select the goal once at the beginning of the episode (g_t is selected once from $\mathcal{C}_0$). SUN instead continuously monitors the value of the current goal, and reselects it from a freshly-resampled candidate set $\mathcal{C}_t$ if

[4] This is the pure-exploration variant of DISCOVER (referred to as "achievability + novelty" in its paper). Its full objective, designed for GCRL, additionally combines this with a direction term toward a task-specific goal.

[5] This is the deep-RL variant of AdaGoal, where σ approximates a prediction-error. The tabular version constrains the $\arg\max$ to the expected hitting time as a proxy for reachability, but the deep-RL variant does not.

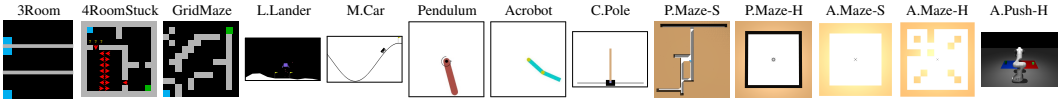


Figure 3: **Environments.** Control tasks are standard benchmarks. *Gridworlds are a novel contribution of this paper.* Blue tiles mark starting positions and green tiles mark terminal states. Tiles with a yellow `?` have a 50% chance of triggering a random movement; tiles with red arrows admit only the corresponding movement. In `FourRoomStuck`, this creates a trap: the agent can accidentally enter the bottom-left room, where any action that does not match the red arrow leaves the agent in place.

needed. *Second*, SUN replaces the ensemble with a lightweight pseudocount, removing the cost of training and querying K critics, and introduces an effective balance between reachability and novelty. Section 4 shows these differences matter: episodic commitment fails when goals become unreachable mid-episode, DISCOVER is sensitive to β, and AdaGoal has no effective reachability proxy.

Proto-Goals. Bagaria et al. (2023) also combine novelty and reachability, but *sequentially*: first, goal candidates are sampled proportionally to a count-based novelty; then, the one with highest SVF is pursued. Thus, reachability cannot recover from a novelty draw that misses, and novelty cannot override a reachability $\arg\max$. The authors report that local reachability hurt performance by biasing selection toward easy goals, until an additional timescale-stratification mechanism was introduced — exactly the bias our multiplicative indicator avoids (Section 4.2). Similarly, their goal is selected once per episode and pursued until achieved, and the authors list finer-grained goal switching (Pislar et al., 2022) as future work, which is precisely what our adaptive strategy provides.

Directed Exploration. Closest to our pseudocount is the episodic novelty module of NGU (Badia et al., 2020), which also estimates counts with a nearest-neighbor density estimator. Two differences matter. *First*, its memory is cleared at every episode, so its counts measure within-episode novelty, while SUN leverages lifetime novelty. *Second*, it recomputes the kernel sum at query time, which is affordable only for a memory at most one episode long.

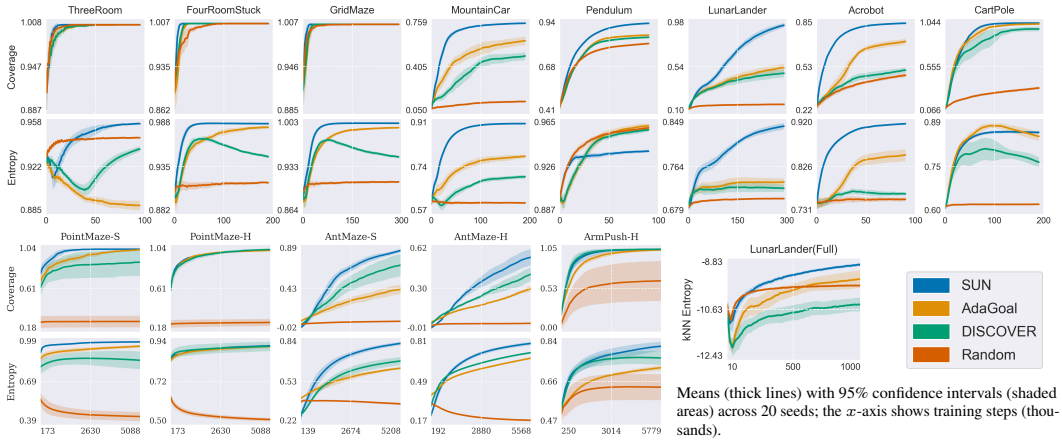
4 Experiments

We select benchmarks that illustrate three challenges in RL exploration: (a) unreachable states or irreversible transitions, (b) large or unbounded goal spaces, and (c) hard exploration (hard-to-reach states, obstacles, mazes). Goal dimensionality ranges from two (most environments) to eight (`LunarLander(Full)`). Details are in Figure 3 and Appendix D.

- **Gridworlds** (a, c): `ThreeRoom` is a larger variant of Figure 1; the agent spawns in a random isolated room, with higher spawn rate in the first two. In `FourRoomStuck`, the bottom-left room cannot be exited or traversed freely, and some transitions are stochastic. `GridMaze` features a narrow passage near the agent's starting state.
- **Classic control** (Towers et al., 2024) (a, b, c): `MountainCar` is a well-known hard-exploration benchmark. In `CartPole`, episodes terminate quickly when the pole falls, making the corners of the state space hard to reach. `LunarLander` has an unbounded state space (the agent can fly arbitrarily high). `Acrobot` has unreachable configurations, and `Pendulum` hard-to-reach ones.
- **GCRL control** (Bortkiewicz et al., 2025) (b, c): `PointMaze-S/H` and `AntMaze-S/H` are locomotion environments of increasing difficulty featuring corridors and dead-ends. `ArmPush-H` is a manipulation task where a Franka Panda pushes a cube on an unbounded plane.

Evaluations. *First*, we compare SUN against a random uniform exploration baseline, AdaGoal (Tarbouriech et al., 2022) and DISCOVER (Diaz-Bone et al., 2025). Not only are these two close to SUN, but they have achieved state-of-the-art performance and outperformed algorithms like MEGA (Pitis et al., 2020). SUN, AdaGoal, and DISCOVER all use HER (Andrychowicz et al., 2017) to learn their SVFs. *Second*, we ablate SUN components, i.e., its indicator and goal-selection strategy. *Finally*, we analyze why AdaGoal, DISCOVER, and non-adaptive goal-selection fail.

We evaluate two metrics over the goal space: **coverage** (fraction of space visited) and Shannon **entropy** of goal visits normalized to $[0, 1]$. We also report heatmaps for a qualitative analysis. In continuous spaces, we discretize them into 50 bins per dimension to compute these metrics. In `LunarLander(Full)`, where the goal is eight-dimensional, we report the entropy approximated via Kozachenko-Leonenko k-NN (Appendix D). All results are averaged across **20 seeds**.



Means (thick lines) with 95% confidence intervals (shaded areas) across 20 seeds; the x-axis shows training steps (thousands).

Figure 4: **Main results**. *SUN outperforms all baselines in all environments.* Note that full coverage is easy in Gridworlds, but uniform exploration (high entropy) is not.

4.1 SUN vs Baselines

Figure 4 shows that the two metrics are complementary: coverage measures whether a state has been visited at least once, while entropy measures how uniformly visits are distributed. A method can saturate coverage while still concentrating most of its visits in small regions of the state space, which is what happens in Gridworlds: all methods reach perfect coverage, yet their entropy values differ substantially, with SUN at the top. Indeed, Gridworlds are challenging due to their non-uniform initial-state distributions and unreachable goals (`ThreeRoom`), stochastic transitions and irreversible traps (`FourRoomStuck`), and bottlenecks that must be traversed to reach the rest of the grid (`GridMaze`). Coverage alone hides these difficulties; entropy reveals them.

This is confirmed by Figure 5, showing the visit distributions at the end of training (black regions are unvisited or unreachable). SUN always covers a broader portion of the goal space and produces smoother visit patterns, unlike AdaGoal and DISCOVER. E.g., in `FourRoomStuck` their visits are non-uniform inside the bottom-left room; in `GridMaze` they cluster near the bottom-left corner; in control environments they fail to explore far from the initial state. This confirms the entropy ranking of Figure 4: methods can cover many goals while still clustering their visits.

Continuous-control environments strengthen SUN's advantage: its curves rise faster and plateau only at (near-)full coverage. One exception is `Pendulum`, the only case where SUN's entropy decreases. The reason is structural: reaching some pendulum configurations requires accumulating momentum through swings, thus the agent must revisit the same low-momentum positions over and over. Rare states are therefore rare precisely because they can only be reached by re-traversing common ones many times; visiting the tail of the distribution comes at the cost of re-visiting the mode. Importantly, SUN also attains the best performance on `LunarLander(Full)`, despite its higher dimensionality. This supports the proposed pseudocount as an effective tool for estimating visits and encoding novelty.

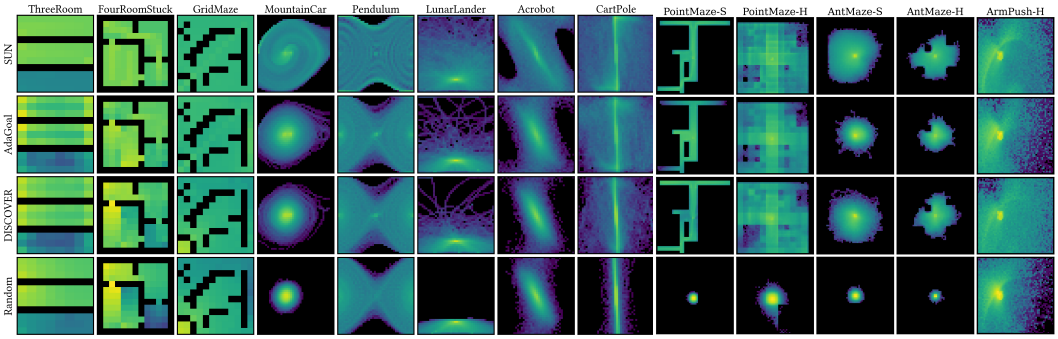


Figure 5: **Log-scale visitation heatmaps** of positional goals at the end of training (first seed). Details about positional goals are in Appendix D. *SUN is the only algorithm not clustering its visits, except in the regions corresponding to the episode's starting state (that are naturally visited more).*

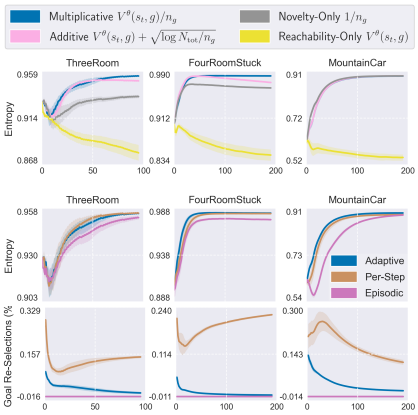


Figure 6: **Indicator ablation**. Driven by *reachability only*, the agent cannot explore (the most reachable state is the current one). *Novelty-only* suffices if goals are always reachable (`MountainCar`), but fails otherwise. The *additive indicator* saturates: as counts increase, the novelty bonus vanishes and the indicator reduces to pure reachability.

Figure 7: **Goal-selection ablation**. *Episodic* selection fails when goals become unreachable mid-episode (as in `FourRoomStuck`). *Per-step* selection avoids this failure mode, but its reselection rate never drops to zero, undermining overall performance. In contrast, the reselection rate of *adaptive* selection drops to zero, indicating that the agent has learned to reach the goals it selects.

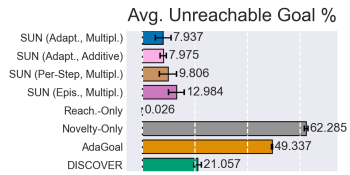


Figure 8: **Selection rates of unreachable goals** averaged over `ThreeRoom` and `FourRoomStuck`. All *SUN variants* select few of them, and only early in training (see Appendix I). *Novelty-only* and *AdaGoal* select them throughout, since neither has an effective reachability proxy (unlike *DISCOVER*). *Reachability-only* never selects any, but only because it simply targets the current state.

4.2 SUN Ablations

SUN has three components: the multiplicative indicator, the adaptive goal-selection, and the pseudocounts. Figure 6 ablates indicators, and reinforces the main motivation of this paper: *exploration is ineffective when guided by reachability or novelty alone; both must be considered.* Figure 7 ablates goal-selection and validates the importance of the adaptive strategy, highlighting the shortcomings of episodic[6] and per-step.[7] All plots are in Appendix G and I. We ablate pseudocounts in Appendix H. Figure 8 confirms that SUN rejects unreachable goals.

4.3 Why AdaGoal and DISCOVER Fail

The results so far are clear: SUN attains the best entropy and coverage, and the ablations validate the importance of its components. Yet another result stands out (Table 1): *all SUN versions are better than AdaGoal and DISCOVER* — strong evidence of SUN's indicator importance. Figure 8 gives part of the answer, but is limited to Gridworlds, where the reachable set is known. Figure 9 therefore shows the algorithms' goals selected over training in `MountainCar`. *DISCOVER* (last row) is biased toward reachability: it mostly selects goals close to the starting state (the center of the heatmap). This may stem from its coefficient β, which must balance the scales of the reachability and novelty terms, an issue no SUN version faces. *AdaGoal* (fifth row), conversely, is biased toward novelty: it selects goals far from the starting state, "skipping" intermediate ones. This is a sign that reachability is ignored, and it is not unexpected. As noted in Section 3.4, we evaluate the deep-RL variant of AdaGoal, which — unlike the tabular version — does not constrain goal-selection by the estimated goal-hitting time. Tarbouriech et al. argue that this is approximated implicitly by the disagreement of the critic ensemble. Our results suggest that this does not always hold. Novelty alone is a reasonable signal for goals the agent can actually reach, since visiting them resolves the disagreement at little cost.

	Coverage			Entropy		
	Ada	DISC	Rand	Ada	DISC	Rand
SUN (Adaptive, Multipl.)	**+23.6%**	**+23.1%**	**+209.6%**	**+7.2%**	**+8.4%**	**+42.2%**
SUN (Adaptive, Additive)	+12.4%	+14.2%	+165.4%	+0.5%	+1.9%	+27.8%
SUN (Per-Step, Multipl.)	+19.3%	+18.8%	+196.4%	+5.4%	+6.6%	+39.5%
SUN (Episodic, Multipl.)	+7.5%	+9.7%	+154.1%	+2.2%	+3.5%	+34.3%

Table 1: **Relative area-under-the-curve improvement** of each SUN version over the baselines, averaged over all environments except `LunarLander(Full)`. All SUN versions outperform the baselines.

[6] More specifically, we implement DISCOVER's episodic goal-selection, where the agent enters a "random exploration phase" if the goal is reached before the end of the episode (Diaz-Bone et al., 2025). The rationale is that a goal selected for its high novelty is itself a point of interest, so nearby states are likely to be novel as well.

[7] In *per-step* selection, a fresh batch of candidates is sampled at every step, and the goal changes whenever the batch contains a candidate with a higher SUN score (Eq. (2)) than the current one.

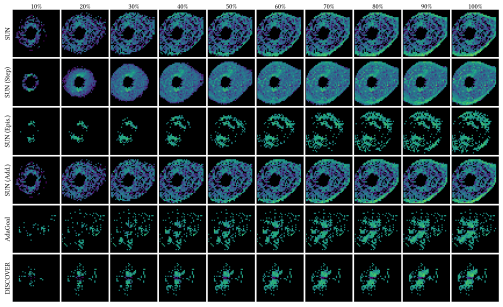


Figure 9: **Goals selected** over time in `M.Car`.

Unreachable goals, however, resolve only after enough failed attempts for every ensemble member to recognize them as such, and each of those attempts is an episode without novel data. *All SUN versions* (first three rows) instead *progressively select goals that are further away*. The contrast is clearest between SUN Episodic and AdaGoal: both select once per episode, yet AdaGoal only aims immediately to distant hard-to-reach states. The progression is therefore intrinsic to the indicator rather than a product of reselection, and it is what balances reachability against novelty.

The results in this section validate our claims. (1) SUN attains the best entropy and coverage across all environments, visiting the goal space significantly more uniformly than every baseline. (2) Its pseudocount effectively approximates novelty even in high-dimensional goal spaces. (3) Its multiplicative indicator does not vanish and selects goals that are both reachable and novel, and its adaptive goal-selection outperforms the baselines' episodic strategy.

5 Discussion

In this paper, we tackled exploration in RL via GCRL, by selecting goals that are simultaneously novel and reachable. We argued that existing methods cannot fully capture the tension between the two and end up neglecting one or the other. We thus proposed SUN, a unified and principled indicator and goal-selection rule grounded in SVFs and counts, and a lightweight pseudocount that scales to per-step goal-selection in continuous spaces. Then, we validated SUN on standard and new benchmarks, where it consistently outperforms state-of-the-art methods AdaGoal and DISCOVER.

Strengths. The strength of SUN lies in its principled and modular design. Its score admits a natural interpretation as the optimal value of a count-bonus exploration objective, driving the agent toward rarely-visited states *within reach*. This is made possible by our lightweight pseudocount, which avoids the overhead of classical pseudocount and density-based approaches while preserving their accuracy. Our results confirm this: AdaGoal and DISCOVER, which instead score goals by critic-ensemble disagreement, fail to balance the two signals and collapse toward one or the other.

Limitations and Future Work. *First*, although principled, count-based novelty is ineffective in large spaces, such as image observations, *regardless of their pseudocount approximations*. Since most observed states are effectively unique, counts become nearly uniform across the buffer and do not provide a useful novelty signal. In such regimes, exploration likely requires richer signals, such as learned curiosity (Raileanu and Rocktäschel, 2020; Parisi et al., 2021) or mutual information between trajectories and learned latents (Eysenbach et al., 2019; Sharma et al., 2020). Combining these signals with SUN's reachability factor is a promising direction.

Second, SUN score does not account for the states traversed to reach a goal. A longer path through many novel states may be preferable to a shorter path to a marginally rarer goal — a distinction the current formulation cannot make. Extending SUN indicator to a cumulative form would more directly target a maximum-entropy state distribution (Hazan et al., 2019; Mutti et al., 2021).

Third, the SVF is learned on the binary reward of Eq. (1), which is sparse by construction and needs hindsight relabeling. Yet the state occupancy that Eq. (1) defines can be estimated directly by contrastive learning without without relabeling (Eysenbach et al., 2022; Jiang et al., 2025; Mohamed et al., 2026). Bae et al. (2024) further use a quasimetric variant (Wang et al., 2023) as the reward of the goal-conditioned policy. Supplying SUN's reachability factor this way is a natural next step.

Finally, SUN draws goals from the replay buffer, thus it can only propose goals it has already encountered. A natural way to relax this is to exploit the geometry of the goal space. Recent work on temporal distances and quasimetric value functions (Myers et al., 2024) learns goal-space metrics that generalize beyond observed pairs; Laplacian-style representations (Shehmar et al., 2026) provide a similar latent geometry. Bae et al. (2024) already select goals in such a space but without a reachability term conditioned on the current state. Separately, studies on out-of-distribution generalization in goal-conditioned RL (Yang et al., 2023) characterize the design choices that enable extrapolation to unseen goals. Combining SUN's reachability with such learned metrics is a promising direction.

AI USE STATEMENT

In this work, we used generative AI tools (LLM-based coding assistants) to assist us for implementing baseline methods and scripts to render figures. Additionally, we used generative AI tools to search for and summarize related work. We have *not* used generative AI tools to develop the theoretical results, formulate mathematical claims, or write proofs, nor to propose hypotheses or design the method; generating synthetic datasets, translation, dataset cleaning, and qualitative data analysis are not applicable to this work. We have reviewed all AI-assisted work: we reviewed all AI-written code and tested it against the equations of the original papers, checked every AI-suggested reference against its original source, and regenerated all reported numbers and figures from the logged experiment data. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The supplementary material contains the full source code: a PyTorch implementation for the Gridworlds and classic-control tasks and a JAX implementation for the JaxGCRL tasks. Each comes with a README listing the commands and configuration files behind every reported experiment, the seeds used, and the scripts that compute the reported metrics and draw the figures. The new Gridworld environments are included in the code and described in Appendix D. Complete proofs of the theoretical results, with their assumptions, are in Appendix A. The pseudocount radius and feature standardization are specified in Appendix B, implementation notes in Appendix C, all hyperparameters and network architectures in Appendix F, and hardware and runtimes in Appendix E.

REFERENCES

J. Adamczyk, A. Kamoski, and R. V. Kulkarni. Maximum entropy exploration without the rollouts. *arXiv:2603.12325*, 2026.

M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

M. G. Azar, I. Osband, and R. Munos. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

A. P. Badia, P. Sprechmann, A. Vitvitskyi, Z. D. Guo, B. Piot, S. Kapturowski, O. Tieleman, M. Arjovsky, A. Pritzel, A. Bolt, and C. Blundell. Never give up: Learning directed exploration strategies. In *International Conference on Learning Representations (ICLR)*, 2020.

J. Bae, K. Park, and Y. Lee. TLDR: Unsupervised goal-conditioned RL via temporal distance-aware representations. In *Conference on Robot Learning (CoRL)*, 2024.

A. Bagaria, R. Jiang, R. Kumar, and T. Schaul. Scaling goal-based exploration via pruning proto-goals. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2023.

M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

L. Blier, C. Tallec, and Y. Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *arXiv:2101.07123*, 2021.

D. Borsa, A. Barreto, J. Quan, D. Mankowitz, R. Munos, H. Van Hasselt, D. Silver, and T. Schaul. Universal successor features approximators. In *International Conference on Learning Representations (ICLR)*, 2019.

M. Bortkiewicz, W. Pałucki, V. Myers, T. Dziarmaga, T. Arczewski, Ł. Kuciński, and B. Eysenbach. Accelerating goal-conditioned reinforcement learning algorithms and research. In *International Conference on Learning Representations (ICLR)*, 2025.

Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. In *International Conference on Learning Representations (ICLR)*, 2019.

C. Colas, T. Karch, O. Sigaud, and P.-Y. Oudeyer. Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. *Journal of Artificial Intelligence Research (JAIR)*, 74:1159–1199, 2022.

P. Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

L. Diaz-Bone, M. Bagatella, J. Hübotter, and A. Krause. DISCOVER: Automated curricula for sparse-reward reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

M. Elsayed, E. S. Lupu, G. Vasan, and A. R. Mahmood. Streaming deep reinforcement learning finally works. *arXiv:2410.14606*, 2026.

B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations (ICLR)*, 2019.

B. Eysenbach, T. Zhang, S. Levine, and R. R. Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning (ICML)*, 2018.

S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in Actor-Critic methods. In *International Conference on Machine Learning (ICML)*, 2018.

I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *International Conference on Machine Learning (ICML)*, 2013.

K. Hartikainen, X. Geng, T. Haarnoja, and S. Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. In *International Conference on Learning Representations (ICLR)*, 2020.

E. Hazan, S. Kakade, K. Singh, and A. Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning (ICML)*, 2019.

A. K. Jain, L. Lehnert, I. Rish, and G. Berseth. Maximum state entropy exploration using predecessor and successor representations. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

T. Jaksch, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 11:1563–1600, 2010.

Y. Jiang, Q. Liu, Y. Yang, X. Ma, D. Zhong, H. Hu, J. Yang, B. Liang, B. XU, C. Zhang, and Q. Zhao. Episodic novelty through temporal distance. In *International Conference on Learning Representations (ICLR)*, 2025.

L. F. Kozachenko and N. N. Leonenko. Sample estimate of the entropy of a random vector. *Problems of Information Transmission*, 23(2):9–16, 1987.

T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman. Deep successor reinforcement learning. *arXiv:1606.02396*, 2016.

L. Lee, B. Eysenbach, E. Parisotto, E. Xing, S. Levine, and R. Salakhutdinov. Efficient exploration via state marginal matching. *arXiv:1906.05274*, 2019.

S. H. Lim and P. Auer. Autonomous exploration for navigating in MDPs. In *Conference on Learning Theory*, 2012.

M. Liu, M. Zhu, and W. Zhang. Goal-conditioned reinforcement learning: Problems and solutions. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

F. Mohamed, C. Ji, B. Eysenbach, and G. Berseth. Temporal representations for exploration: Learning complex exploratory behavior without extrinsic rewards. In *International Conference on Learning Representations (ICLR)*, 2026.

M. Mutti, L. Pratissoli, and M. Restelli. Task-agnostic exploration via policy gradient of a non-parametric state entropy estimate. In *AAAI Conference on Artificial Intelligence*, 2021.

V. Myers, C. Zheng, A. Dragan, S. Levine, and B. Eysenbach. Learning temporal distances: Contrastive successor features can provide a metric structure for decision-making. In *International Conference on Machine Learning (ICML)*, 2024.

S. Parisi, V. Dean, D. Pathak, and A. Gupta. Interesting object, curious agent: Learning task-agnostic exploration. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, 2017.

M. Pislar, D. Szepesvari, G. Ostrovski, D. Borsa, and T. Schaul. When should agents explore? In *International Conference on Learning Representations (ICLR)*, 2022.

S. Pitis, H. Chan, S. Zhao, B. Stadie, and J. Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2020.

T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9): 1481–1497, 1990.

V. Pong, M. Dalal, S. Lin, A. Nair, S. Bahl, and S. Levine. Skew-Fit: State-covering self-supervised reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2020.

R. Raileanu and T. Rocktäschel. RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments. In *International Conference on Learning Representations (ICLR)*, 2020.

Z. Ren, K. Dong, Y. Zhou, Q. Liu, and J. Peng. Exploration via hindsight goal generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *International Conference on Machine learning (ICML)*, 2015.

A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations (ICLR)*, 2020.

D. Shehmar, M. Schlegel, M. E. Taylor, and M. C. Machado. Laplacian representations for decision-time planning. In *International Conference on Machine Learning (ICML)*, 2026.

A. L. Strehl and M. L. Littman. An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences (JCSS)*, 74(8):1309–1331, 2008.

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

H. Tang, R. Houthooft, D. Foote, A. Stooke, O. X. Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel. #Exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

J. Tarbouriech, M. Pirotta, M. Valko, and A. Lazaric. Improved sample complexity for incremental autonomous exploration in MDPs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

J. Tarbouriech, O. D. Domingues, P. Ménard, M. Pirotta, M. Valko, and A. Lazaric. Adaptive multi-goal exploration. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv:2407.17032*, 2024.

H. van Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

T. Wang, A. Torralba, P. Isola, and A. Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning (ICML)*, 2023.

C. J. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

R. Yang, Y. Lin, X. Ma, H. Hu, C. Zhang, and T. Zhang. What is essential for unseen goal generalization of offline goal-conditioned RL? In *International Conference on Machine Learning (ICML)*, 2023.

C. Zhang, Y. Cai, L. Huang, and J. Li. Exploration by maximizing Renyi entropy for reward-free RL framework. In *AAAI Conference on Artificial Intelligence*, 2021.

C. Zheng, R. Salakhutdinov, and B. Eysenbach. Contrastive difference predictive coding. In *International Conference on Learning Representations (ICLR)*, 2024.

APPENDICES

A Theoretical Properties **15**
A.1 Setup and Notation . . . 15
A.2 Count-Bonus Equivalence . . . 15
A.3 Hitting-Probability Bound . . . 16
A.4 Rejection of Unreachable Goals . . . 17
A.5 Log-space Decomposition and Connections to Prior Work . . . 17
A.6 Extension to Stochastic Dynamics . . . 17
A.7 Adaptive Goal-Selection: Theoretical Consistency and Practical Motivation . . . 18

B Pseudocount Radius and Standardization **19**

C Practical Notes **19**

D Environment Details **20**

E Source Code, Compute Details, and Runtimes **22**

F Training Hyperparameters **23**
F.1 Gridworlds and Classic Control . . . 24
F.2 GCRL Control . . . 24
F.3 Network Architecture . . . 25

G Indicator and Goal-Selection Ablations **27**

H Pseudocounts Ablation **27**

I Detailed Analysis Of All Environments **28**

J Successor Value Function Visualization **33**

K SUN-UCB: A Structural Connection to PAC Analysis **35**
K.1 Setup . . . 35
K.2 SUN-UCB Algorithm . . . 35
K.3 Conjectured Sample Complexity . . . 36
K.4 Partial Analysis . . . 36
K.5 Discussion . . . 38
K.6 Additive vs. Multiplicative SUN . . . 38

A THEORETICAL PROPERTIES

This section analyzes the SUN selection rule under an oracle: the SVF $V^\pi(s, g)$ and the count n_g are exact, the replay buffer is frozen, and the goal-conditioned policy π acts to reach g. The analysis is structural — it characterizes what SUN does given perfect estimates, not what it learns from samples. Within this scope, we establish three properties (Sections A.2–A.4) that together formalize how SUN balances novelty and reachability, and we draw connections to existing methods through a log-space decomposition (Section A.5).

A.1 SETUP AND NOTATION

Let $\mu(g)$ denote the replay marginal over goals — the empirical distribution of states stored in the buffer. The pseudocount n_g from Section 3.3 is a finite-sample estimate of $\mu(g)$ up to a normalization constant; under the oracle, we treat n_g as exact and $\mu(g) > 0$ for every candidate. We define the *log-rarity potential*

$$\Phi(g) \triangleq \log \mu(g), \tag{3}$$

which is monotone in n_g: small n_g corresponds to small $\mu(g)$, hence small (very negative) $\Phi(g)$. The SUN novelty score $\nu(g) = 1/n_g$ in Section 3 therefore corresponds to $-\Phi(g)$ in log-space, up to a constant.

Throughout the analysis, π denotes the goal-conditioned policy targeting g, and $\tau_g \triangleq \inf\{t \geq 0 : s_t = g\}$ is its (random) hitting time of g from a starting state s. The SVF (Eq. 1) under π admits the equivalent form

$$V^\pi(s, g) = \mathbb{E}_\pi[\gamma^{\tau_g} \mathbf{1}_{\{\tau_g < \infty\}}], \tag{4}$$

when g is terminal (i.e., once reached, the episode is considered ended). This identity will be used repeatedly.

Assumption 1 (Oracle setting)**.** *From now on, we assume the following conditions are true.*

(C1) *The state space $\mathcal{S}$ is finite.*

(C2) *Transitions are deterministic.*

(C3) *The replay marginal μ, the SVF V^π, and the count n_g are frozen during the analysis.*

(C4) *Under π, if g is reachable the trajectory hits it after $k_\pi(s, g) - 1 \in \{0, 1, 2, \dots\}$ steps, where $k_\pi(s, g)$ counts the states on the path from s to g inclusive (so $k_\pi(g, g) = 1$), and the episode is considered ended at the first hit (i.e., goal-reaching transitions are terminal). If g is unreachable, $k_\pi(s, g) = \infty$. We further assume π is optimal for its own goal, i.e. $k_\pi(s, g) = \min_{\pi'} k_{\pi'}(s, g)$ for all s; this is what licenses the shortest-path inequality in the proof of Proposition 1(ii).*

Assumption 1 mirrors the structural setup used in the autonomous-exploration literature (Lim and Auer, 2012; Tarbouriech et al., 2020; 2022). Section A.6 relaxes (C2)–(C3) to stochastic dynamics with almost-sure hitting.

A.2 COUNT-BONUS EQUIVALENCE

The first result links SUN to the count-based intrinsic motivation literature (Bellemare et al., 2016): SUN is exactly the value of a goal-specific count-bonus reward.

Theorem 1 (Count-bonus equivalence)**.** *Under Assumption 1, for any goal g and any policy π, define the goal-specific reward $r_t^g \triangleq \mathbf{1}_{\{s_t = g\}}/n_g$, where n_g is the (frozen) count from Assumption 1(C4). Then*

$$\mathbb{E}_\pi\left[\sum_{t \geq 0} \gamma^t r_t^g \;\middle|\; s_0 = s\right] = \frac{V^\pi(s, g)}{n_g} = \mathrm{SUN}(g \,|\, s). \tag{5}$$

Proof. By Assumption 1(C4), n_g is frozen during the analysis and therefore independent of the trajectory. It thus factors out of the expectation:

$$\mathbb{E}_\pi\left[\sum_{t\geq 0}\gamma^t\, r_t^g \;\middle|\; s_0 = s\right] \;=\; \mathbb{E}_\pi\left[\sum_{t\geq 0}\gamma^t\,\frac{\mathbf{1}_{\{s_t=g\}}}{n_g} \;\middle|\; s_0 = s\right] \tag{6}$$

$$= \;\frac{1}{n_g}\,\mathbb{E}_\pi\left[\sum_{t\geq 0}\gamma^t\,\mathbf{1}_{\{s_t=g\}} \;\middle|\; s_0 = s\right] \tag{7}$$

$$= \;\frac{V^\pi(s,g)}{n_g} \;=\; \mathrm{SUN}(g\,|\,s), \tag{8}$$

where the last equality uses the definition of the SVF (Eq. 1) and the SUN score (Eq. 2). □

This identity is the simplest interpretation of SUN: maximizing $\mathrm{SUN}(g\,|\,s)$ over g is equivalent to acting greedily with respect to the optimal value of an exploration reward that pays inversely proportional to how often g has been visited. SUN is therefore not just a heuristic combination of two signals — it is the value of a single, principled exploration objective.

Remark 1 (Frozen vs. online counts). The equivalence holds when n_g is treated as fixed during the trajectory — the standard frozen-replay setting (C4). When counts are updated online (as in our practical algorithm and in the standard count-bonus literature), the surrogate reward becomes non-stationary and the equivalence holds only approximately, with the size of the gap controlled by how much n_g changes over the trajectory.

Remark 2 (Form vs. exponent). Theorem 1 holds for any $\nu(g) = f(n_g)$, since n_g is frozen and factors out of the expectation. It therefore justifies the multiplicative *form* but not the exponent: $\nu(g) = 1/n_g$ is a choice, with $1/\sqrt{n_g}$ (Bellemare et al., 2016) the natural alternative.

A.3 Hitting-Probability Bound

The SVF $V^\pi(s,g)$ is a discounted occupancy, but the property we ultimately care about is whether the agent reaches g from s within a reasonable horizon. The next theorem connects the two for the goal-conditioned policy π, under the terminal-goal assumption (C4) of Assumption 1: once the agent reaches g, it stays. In this setting, $V^\pi(s,g) = \mathbb{E}[\gamma^{\tau_g}\mathbf{1}_{\{\tau_g<\infty\}}] \leq 1$ (Section A.1), and a high SVF guarantees a high hitting probability over a short horizon.

Theorem 2 (Short-horizon hitting bound)**.** *Under Assumption 1 with (C2) relaxed to stochastic transitions (Section A.6), for every horizon $n \geq 0$,*

$$\Pr_\pi[\tau_g \leq n] \;\geq\; V^\pi(s,g) \;-\; \gamma^{n+1}. \tag{9}$$

In particular, choosing n such that $\gamma^{n+1} \leq V^\pi(s,g)/2$ gives $\Pr_\pi[\tau_g \leq n] \geq V^\pi(s,g)/2$. This horizon is $n = \mathcal{O}(\log(1/V^\pi)/(1-\gamma))$.

Proof. Under Assumption (C4), $V^\pi(s,g)$ admits the equivalent form $V^\pi(s,g) = \mathbb{E}[\gamma^{\tau_g}\mathbf{1}_{\{\tau_g<\infty\}}]$ derived in Section A.1. We split the expectation by hitting horizon:

$$V^\pi(s,g) \;=\; \mathbb{E}[\gamma^{\tau_g}\,\mathbf{1}_{\{\tau_g<\infty\}}] \tag{10}$$

$$\;=\; \mathbb{E}[\gamma^{\tau_g}\,\mathbf{1}_{\{\tau_g\leq n\}}] \;+\; \mathbb{E}[\gamma^{\tau_g}\,\mathbf{1}_{\{n<\tau_g<\infty\}}]\,. \tag{11}$$

We bound each term separately.
For the first term, $\gamma^{\tau_g} \leq 1$ when $\tau_g \leq n$, so

$$\mathbb{E}[\gamma^{\tau_g}\,\mathbf{1}_{\{\tau_g\leq n\}}] \;\leq\; \mathbb{E}[\mathbf{1}_{\{\tau_g\leq n\}}] \;=\; \Pr_\pi[\tau_g \leq n]. \tag{12}$$

For the second term, $\gamma^{\tau_g} \leq \gamma^{n+1}$ when $\tau_g > n$, so

$$\mathbb{E}[\gamma^{\tau_g}\,\mathbf{1}_{\{n<\tau_g<\infty\}}] \;\leq\; \gamma^{n+1}\,\Pr_\pi[\tau_g > n] \;\leq\; \gamma^{n+1}. \tag{13}$$

Combining the two bounds gives $V^\pi(s,g) \leq \Pr_\pi[\tau_g \leq n] + \gamma^{n+1}$, and rearranging yields

$$\Pr_\pi[\tau_g \leq n] \;\geq\; V^\pi(s,g) \;-\; \gamma^{n+1}. \tag{14}$$

For the horizon claim, set $\gamma^{n+1} \leq V^\pi/2$ and solve: $n+1 \geq \log(2/V^\pi)/\log(1/\gamma)$, which gives $n = \mathcal{O}(\log(1/V^\pi)/(1-\gamma))$ since $\log(1/\gamma) \geq 1-\gamma$ for $\gamma \in (0,1)$. □

The horizon scales logarithmically in $1/V^\pi$ and inversely in $1-\gamma$: high-SVF goals are hit quickly with high probability, while low-SVF goals require long horizons. This is the formal version of the intuition that the SVF is a reachability signal.

A.4 Rejection of Unreachable Goals

The third result is the dual of Theorem 2: goals that no policy can reach receive zero score, so SUN never selects them.

Theorem 3 (Unreachability rejection)**.** *Let Π be any class of policies. If g is unreachable from s under every $\pi \in \Pi$ — that is, $\Pr_\pi[\tau_g < \infty] = 0$ for all $\pi \in \Pi$ — then*

$$\sup_{\pi \in \Pi} V^\pi(s, g) = 0, \qquad \mathrm{SUN}(g \,|\, s) = 0, \tag{15}$$

regardless of n_g.

Proof. For any $\pi \in \Pi$, the unreachability hypothesis $\Pr_\pi[\tau_g < \infty] = 0$ means the reward $\mathbf{1}_{\{\tau_g < \infty\}}$ is zero with probability one. Hence

$$V^\pi(s, g) = \mathbb{E}_\pi[\gamma^{\tau_g}\, \mathbf{1}_{\{\tau_g < \infty\}}] = 0, \tag{16}$$

and $\mathrm{SUN}(g \,|\, s) = 0/n_g = 0$. □

This property fails for novelty-only scores ($1/n_g$ alone), which assign maximal value to never-visited unreachable goals — exactly the failure mode shown in Figure 1. SUN is therefore guaranteed to ignore goals that the agent cannot reach, regardless of how rare they are.

A.5 Log-space Decomposition and Connections to Prior Work

A useful consequence of the count-bonus form is that $\log \mathrm{SUN}$ decomposes additively:

$$\log \mathrm{SUN}(g \,|\, s) \;=\; \log V^\pi(s, g) \;-\; \Phi(g). \tag{17}$$

Under Assumption 1, $V^\pi(s, g) = \gamma^{k_\pi(s,g)-1}$ for reachable g, so the score takes the closed form

$$\log \mathrm{SUN}(g \,|\, s) \;=\; -\big(k_\pi(s, g) - 1\big)\, \kappa \;-\; \Phi(g), \qquad \kappa \triangleq -\log \gamma > 0. \tag{18}$$

SUN therefore takes the form of a soft Lagrangian: $-\Phi(g)$ is the rarity reward, $(k_\pi - 1)\kappa$ is the discounted distance cost, and κ is *derived* from the discount γ rather than introduced as a separate hyperparameter. This is structurally similar to AdaGoal (Tarbouriech et al., 2022), which solves a hard-constrained version with a user-specified radius. The two are not equivalent — AdaGoal uses sample-variance epistemic uncertainty as its novelty signal, while SUN uses distributional rarity — but both balance the same two ingredients.

The log-space form also yields a state-dependent admissibility condition: a goal g is preferred over staying at s if and only if

$$k_\pi(s, g) \;<\; 1 \;+\; \frac{\Phi(s) - \Phi(g)}{\kappa}. \tag{19}$$

A goal that is no rarer than the current state ($\Phi(g) \geq \Phi(s)$) is never preferred, and the maximum admissible distance scales with the rarity gap $\Phi(s) - \Phi(g)$. This is a tighter analogue of the "neither too easy nor too hard" intuition behind AdaGoal and DISCOVER (Diaz-Bone et al., 2025), with the trade-off automatically calibrated by the discount.

A.6 Extension to Stochastic Dynamics

Theorems 1 and 3 hold beyond deterministic dynamics (Theorem 2 is already stated in that setting). Replacing (C2)–(C3) with stochastic transitions and the assumption that τ_g is almost-surely finite for reachable g, the equivalent SVF form $V^\pi(s, g) = \mathbb{E}[\gamma^{\tau_g}]$ continues to hold, and both theorems transfer verbatim. The closed-form $V^\pi = \gamma^{k_\pi - 1}$ in Eq. 18 no longer holds in general — k_π becomes a random hitting time — but the log-space decomposition (Eq. 17) is unchanged.

A.7 Adaptive Goal-Selection: Theoretical Consistency and Practical Motivation

A natural concern with the SUN formulation is that the SVF $V^\pi(s, g)$ is defined under a policy π that pursues g for the entire trajectory, while SUN's adaptive strategy may reselect the goal mid-episode based on a value-consistency check. We show that this concern dissolves in two ways. In the deterministic, oracle setting, the SUN indicator is monotone along any trajectory that pursues its own $\arg\max$: the value never decreases, so the adaptive check never fires, and the goal remains fixed for the entire episode (Proposition 1). In stochastic or approximate settings, monotonicity can fail — but this failure is precisely what the adaptive check is designed to detect.

Proposition 1 (Value monotonicity and goal stability under deterministic dynamics)**.** *Under Assumption 1 (deterministic dynamics, known V^π and n_g, frozen replay), if SUN selects g_t^* at state s_t and the agent takes one step under $\pi_{g_t^*}$ to reach s_{t+1}, then:*

(i) the value along the pursued goal is non-decreasing: $V^\pi(s_{t+1}, g_t^*) \geq V^\pi(s_t, g_t^*)$;
(ii) g_t^* remains the SUN $\arg\max$ at s_{t+1}:

$$g_t^* \;=\; \arg\max_{g \in \mathcal{G}} \frac{V^\pi(s_{t+1}, g)}{n_g}. \tag{20}$$

By (i), the adaptive check $V^\pi(s_t, g_t^*) < V^\pi(s_{t_{\text{sel}}}, g_t^*)$ never fires; by (ii), the argmax is stable. By induction, the same goal is pursued throughout the episode until g_t^* is reached.

Proof. Let $k_g \triangleq k_\pi(s_t, g)$. Under deterministic dynamics, one step along the optimal path to g_t^* reduces its hitting time by exactly one: $k_\pi(s_{t+1}, g_t^*) = k_{g_t^*} - 1$. Applying $V^\pi(s, g) = \gamma^{k_\pi(s,g)-1}$ gives $V^\pi(s_{t+1}, g_t^*) = \gamma^{-1} V^\pi(s_t, g_t^*) \geq V^\pi(s_t, g_t^*)$ (since $\gamma \in (0, 1]$), proving (i).

For (ii), for any other goal g, the triangle inequality on hitting times (a one-step transition can increase the distance to g by at most one) gives $k_\pi(s_{t+1}, g) \geq k_g - 1$, hence $V^\pi(s_{t+1}, g) \leq \gamma^{-1} V^\pi(s_t, g)$. By Assumption 1(C4), μ is frozen, so the counts n_g do not change between t and $t+1$. (Even outside this assumption, $n_{g_t^*}$ would not increment within an episode since g_t^* has not yet been reached.) Therefore

$$\frac{V^\pi(s_{t+1}, g_t^*)}{n_{g_t^*}} = \gamma^{-1} \frac{V^\pi(s_t, g_t^*)}{n_{g_t^*}},$$
$$\frac{V^\pi(s_{t+1}, g)}{n_g} \leq \gamma^{-1} \frac{V^\pi(s_t, g)}{n_g} \quad \forall g.$$

By the optimality of g_t^* at s_t, the ordering is preserved at s_{t+1}. Theorem 3 ensures unreachable goals remain excluded. □

Proposition 1 shows that, in the oracle setting, the adaptive check never fires, so adaptive selection, per-step selection, and fixed-goal commitment produce the same trajectory. The fixed-goal semantics of V^π and the adaptive semantics of SUN are therefore consistent.

Stochastic dynamics. Under stochastic transitions, the next state s_{t+1} is random. The hitting-time identity $k_\pi(s_{t+1}, g_t^*) = k_\pi(s_t, g_t^*) - 1$ holds only *in expectation*, and a realized transition may push the agent to a state where $V^\pi(s_{t+1}, g_t^*) < V^\pi(s_{t_{\text{sel}}}, g_t^*)$ — exactly the condition the adaptive check flags for reselection. A fixed-goal policy would continue pursuing g_t^* regardless; per-step reselection would reselect at every step, discarding stable information. The adaptive check reselects only when the SVF signals a value drop, which under stochastic dynamics corresponds to a transition into a genuinely less favorable region.

Approximate V^θ. The same argument applies when V^θ is learned rather than exact. Errors in V^θ can cause monotonicity to fail even under deterministic dynamics: if $V^\theta(s_t, g_t^*)$ was overestimated at selection time, the true value at s_{t+1} may be lower. The adaptive check detects this and triggers reselection, effectively withdrawing commitment when the SVF's estimates are proved unreliable by their own subsequent values.

SUN's adaptive strategy is therefore the right choice in both regimes. When V^θ is exact and dynamics are deterministic, Proposition 1 shows that the check never fires and the agent commits to a single goal per episode — matching fixed-goal commitment and inheriting its optimality. When V^θ is inaccurate or dynamics are stochastic, the check acts as an SVF-driven trigger: it withdraws commitment precisely when the SVF signals that the current goal is no longer reliable.

B Pseudocount Radius and Standardization

The pseudocount of Section 3.3 relies on a single scalar radius ρ in standardized feature space. We elaborate here on how the scales are estimated and why they must be held fixed.

Robust per-feature scale. A natural choice is to divide each feature by its standard deviation σ_m, so that a single radius ρ has the same meaning across features. Two problems arise. First, visitation distributions are heavy-tailed — a few excursions to a distant part of the state space inflate σ_m for the whole buffer — so the standard deviation does not reflect the spread of the region the agent actually occupies. We therefore use the interquartile range, rescaled so that it matches the standard deviation for Gaussian data,

$$\hat{\sigma}_m \;=\; {Q_{75}(s_m) - Q_{25}(s_m)}/{1.349}, \tag{21}$$

where the quantiles are taken over the stored buffer entries. Second, a nearly constant feature has $\hat{\sigma}_m \approx 0$, and dividing by it amplifies tiny variations, so two points that are almost identical in dimension m end up far apart. We therefore floor $\hat{\sigma}_m$ at a small fraction of the median scale across the non-degenerate features,

$$\tilde{\sigma}_m \;=\; \max\!\left(\hat{\sigma}_m,\; \zeta \cdot \mathrm{median}\{\hat{\sigma}_{m'} : \hat{\sigma}_{m'} > 0\}\right), \qquad \zeta = 10^{-2}, \tag{22}$$

and fall back to $\tilde{\sigma}_m = 1$ for features that are exactly constant across the buffer. This caps the ratio between the widest and narrowest scale without forcing genuinely thin directions to inherit the spread of wide ones. The squared distance used by the pseudocount is then

$$\|s - s'\|^2_{/\tilde{\sigma}} \;=\; \sum_m {(s_m - s'_m)^2}/{\tilde{\sigma}_m^2}. \tag{23}$$

Why the scales must be fixed. Inserting a sample updates only the counts of its own neighbors, so every stored n_g equals the number of buffer entries within ρ of g only as long as $\tilde{\sigma}$ is the same at every insertion. If the scales change, two stored entries can become neighbors under the new metric while neither is a neighbor of the newly inserted sample — neither count is corrected and the error is never recovered. With $\rho = 0.1$ in one dimension, for instance, the points 0 and 1 are two standardized units apart and each hold a count of one; after inserting 100 the scale grows and their separation falls below ρ, yet neither lies within ρ of the new point and both counts stay at one. Recomputing the scales at every insertion would therefore make n_g depend on the order in which samples arrived.

Schedule. We avoid this as follows. First, no counts are maintained during warm-up (i.e., random exploration, see Section F). When the warm-up ends, the scales are computed from all the samples collected so far and every count is built from them in a single pass, which is quadratic in the buffer size but runs at a point where the buffer is still small. The scales are then recomputed every 10^3 insertions and compared with the previous estimate; once the largest relative per-feature change falls below 10^{-2} they are frozen for the remainder of training, and all counts are rebuilt once under them.

Eviction. When the buffer is full, inserting a sample evicts one, and the evicted entry's neighbors are decremented before the new entry's neighbors are incremented, at the same per-step cost. In our experiments the buffer is large enough that no eviction occurs.

C Practical Notes

SVF reward. In RL literature, the reward in Eq. (1) is sometimes replaced by alternatives that target the same quantity through different reward shapings. For example, returning -1 until s_i is reached and 0 thereafter recovers a negative-distance interpretation (Schaul et al., 2015; Andrychowicz et al., 2017). However, those rewards performed worse in our experiments.

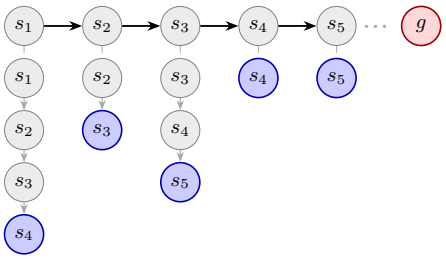


Figure 10: **Example of HER "future" relabeling.** The agent explores states $s_1 \dots s_5$ (top row, black arrows) while trying to reach goal g (red). To provide *positive* rewards, each state s_t is assigned a goal g_t (blue) among future states within the same trajectory. For example, s_1 is assigned $g_1 \leftarrow s_4$. TD targets for s_t are then computed according to the sub-trajectory from s_t to g_t (gray downwards arrows).

Goal relabeling. Training the SVF is a self-supervised process: given tuples (s_t, a_t, g_t, s_{t+1}), V^π can be trained with TD learning using the reward in Eq. (1), i.e., $r_t = \mathbf{1}_{\{s_t=g\}}$. Effective training requires tuples in which the agent both reaches and fails to reach the goal, i.e., *positive* and *negative* samples. Early in training, however, the agent rarely reaches its commanded goal g_t, so the replay buffer contains almost exclusively negatives. Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) addresses this imbalance by relabeling g_t with a state *from the trajectory itself*: the selected state fires the reward and yields positive samples. We adopt HER's "future" strategy: given a trajectory of T steps, g_t is sampled from $\{s_t, s_{t+1}, \dots, s_T\}$. Figure 10 illustrates the procedure. Note that training is off-policy by construction: the relabeled goal is not the goal under which the trajectory was collected. Note that future relabeling only provides positive training samples; negatives are still needed. Common strategies either relabel part of the mini-batch, or duplicate it keeping the original on-policy goals as negatives and relabeling the copies with positives.

Replay buffer eviction. If the buffer is large enough (as in our experiments), no data eviction occurs when new samples are inserted (Algorithm 1:17). If eviction happens (e.g. in a fixed-size FIFO buffer), the principled option is to decrement the counts of the evicted sample's neighbors, preserving exactness at the same per-step cost as insertion. A simpler approximation is to leave counts as-is, but this introduces an upward bias on the evicted sample's neighbors — their stored counts no longer reflect the current buffer.

Action-level noise. Ideally, the exploration policy would act greedily with respect to the SVF (Algorithm 1:15). However, since V^θ is inaccurate early in training, it may be appropriate to inject a small amount of noise into $\pi(a \mid s_t, g_t)$. More details are in Appendix F.

D Environment Details

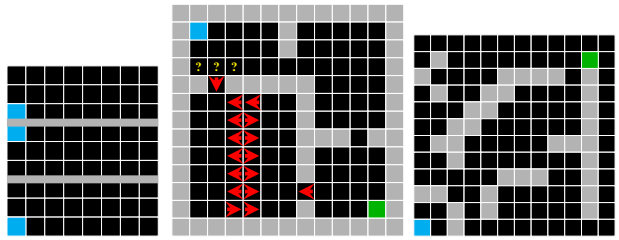
Gridworlds. Novel environments shown in Figure 11. The observation is a one-hot encoding of the agent's tile. The goal space is $\mathcal{S} \times \mathcal{A}$, i.e., the agent should do *every action in all states*.

- `ThreeRoom`: three rooms separated by walls. The agent spawns non-uniformly across three positions (cyan tiles): 47.5% chance in the first room, 47.5% in the second, 5% in the third. There are four actions: left, right, up, down. Episode horizon: 100 steps.
- `FourRoomStuck`: a variation of the classic four-room (Sutton et al., 1999). The bottom-left room can be entered but not exited, and cannot be traversed freely due to one-way tiles. The three tiles above its north entrance (marked with a yellow question mark) are noisy, i.e., movement is randomized with 50% probability. There are four actions: left, right, up, down. Episode horizon: 200 steps. Episodes end when the agent reaches the green tile.
- `GridMaze`: maze with nine actions (left, right, up, down, up-left, down-left, up-right, down-right, stay). Exploration is hard due to the narrow passage near the starting position, that can be traversed only with "up-right". Episode horizon: 200 steps. Episodes end on action "stay" in the green tile.

Classic control. Open-sourced classic RL benchmarks (Towers et al., 2024) shown in Figure 12. The goal space is $\tilde{\mathcal{S}} \times \mathcal{A}$, where $\tilde{\mathcal{S}}$ is a subset of the state space that depends on the environment.[8]

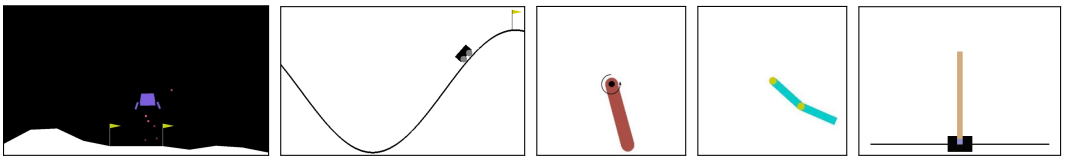
- `LunarLander`: the (x, y) coordinate of the lander, in $[-1, 1] \times [-0.59, \infty]$ (unbounded).
- `LunarLander(Full)`: full eight-dimensional state space.

[8] Gymnasium's (Towers et al., 2024) documented bounds are "advisory" and do not correspond to the actual region the agent can visit. For example, `CartPole`'s documented bounds are $[-4.8, 4.8]$ and $[-24°, 24°]$, but episodes terminate if the cart leaves $[-2.4, 2.4]$ or the pole falls outside $[-12°, 12°]$. Similarly, `LunarLander` episodes terminate if x leaves $[-1, 1]$, despite documented bounds of $[-2.5, 2.5]$.



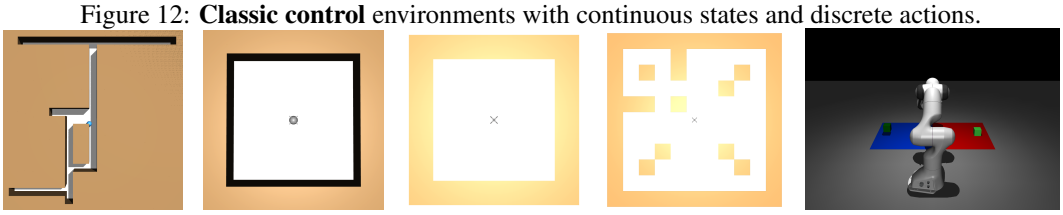
(a) `ThreeRoom` (b) `FourRoomStuck` (c) `GridMaze`

Figure 11: **Gridworlds.** Black tiles are empty; gray tiles are walls; cyan tiles are starting positions; green tiles are terminal positions where the episode ends. Red arrows mark one-way tiles where only the action matching the arrow succeeds. In yellow `?` tiles, movement is randomized with 50% probability.



(a) `LunarLander` (b) `MountainCar` (c) `Pendulum` (d) `Acrobot` (e) `CartPole`

Figure 12: **Classic control** environments with continuous states and discrete actions.



(a) `PointMaze-S` (b) `PointMaze-H` (c) `AntMaze-S` (d) `AntMaze-H` (e) `ArmPush-H`

Figure 13: **GCRL environments** with continuous states and actions. `PointMaze-H` is actually four-dimensional, and Figure (b) shows the two-dimensional projection.

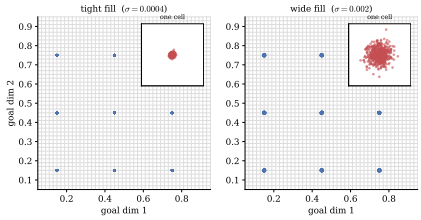
- `MountainCar`: the position $x \in [-1.2, 0.6]$ and the velocity $\dot{x} \in [-0.07, 0.07]$.
- `Pendulum`: the whole state space, i.e., the sine and cosine of the pendulum angle (both bounded in $[-1, 1]$) and its angular velocity (bounded in $[-8, 8]$). The default one-dimensional action is continuous and bounded; here we discretize it into eight equally spaced discrete actions.
- `Acrobot`: the sine and cosine of the joint angles, for a total of four dimensions bounded in $[-1, 1]$.
- `CartPole`: the x coordinate of the cart and the angle of the pole, in $[-2.4, 2.4] \times [-12°, 12°]$.

GCRL Control. Open-sourced GCRL benchmarks (Bortkiewicz et al., 2025) shown in Figure 13. The goal space is $\mathcal{S}$; the action is continuous and not part of the goal space.

- `PointMaze-S`: a point-mass agent navigates a maze. The goal space is its (x, y) planar position, bounded by the maze layout, i.e., $x \in [-11, 12], y \in [-11, 12]$.
- `PointMaze-H`: the area (and the goal space) the agent navigates is four-dimensional, i.e., $[-6, 7]^4$. Its heatmaps only show the first two dimensions.
- `AntMaze-S/H`: like `PointMaze-S`, but the agent is an ant-like quadruped. Goal bounds are $x \in [-17.5, 17.5], y \in [-17.5, 17.5]$ (S) and $x \in [-27.5, 27.5], y \in [-27.5, 27.5]$ (H).
- `ArmPush-H`: a Franka Panda pushes a green cube on a plane. The goal space is the cube's planar (x, y) unbounded position. The blue/red region of the plane in Figure 13, located at $x \in [-0.45, -0.35], y \in [0.60, 0.70]$, is where the cube spawns at the beginning of an episode.

Evaluation metrics. In Gridworlds, Shannon entropy and coverage can be computed exactly, since the set of visitable states is finite and known. In control tasks, we discretize the continuous goal space into 50 bins per dimension, which gives accurate estimates of both metrics. These estimates can be conservative, though: `Acrobot` and `MountainCar`, for example, have unreachable position-velocity configurations, yet we normalize entropy and coverage as if the entire binned space were visitable. Also note that the goal spaces of `LunarLander` and `ArmPush-H` are unbounded. In the former, the y position has no upper limit, and we compute metrics using $[-0.25, 10]$ as its bounds. In the latter, both x, y are unbounded, and we compute metrics using $[-1.00, 1.00] \times [0.15, 1.15]$ as its bounds.

For `LunarLander(Full)`, the goal space is eight-dimensional: the first six coordinates are continuous, and the last two are binary. Binning the six continuous dimensions is challenging: fine binning is computationally expensive, whereas coarse binning is inaccurate. For example, Figure 15 shows



	σ (spread)	H_{Shannon}	$H_{\text{k-NN}}$
tight fill	0.0004	2.1972	-10.6512
wide fill	0.0020	2.1972	-7.4323

Figure 14: Two distributions occupying the *same* nine bins with equal counts. The insets zoom into one bin to reveal the tight vs. wide fill. Shannon entropy is identical ($\log 9$); k-NN differential entropy differs by >3 nats. Note that differential entropy can be negative, unlike discrete Shannon entropy.

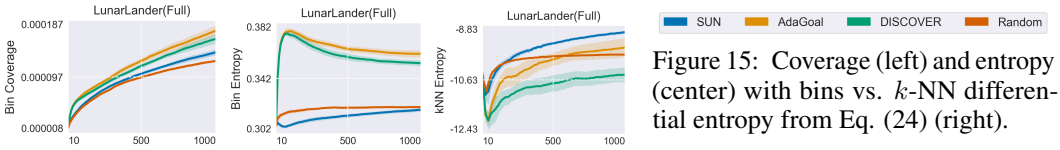


Figure 15: Coverage (left) and entropy (center) with bins vs. k-NN differential entropy from Eq. (24) (right).

coverage and entropy when the first six coordinates are discretized into twelve bins. Coverage is extremely low (below 0.0002%), so entropy is largely determined by the number of occupied cells when they carry comparable mass. Consequently, an algorithm with slightly higher coverage (even as little as <0.0002%, as in DISCOVER) over clustered bins may exhibit higher entropy than one with slightly lower coverage over more dispersed bins (such as SUN). Figure 14 illustrates this issue.

This is why in Figure 4 (and again below in Figure 15) we report approximate continuous differential entropy via the Kozachenko-Leonenko k-NN estimator (Kozachenko and Leonenko, 1987). This estimator does not suffer from the binning issue: it measures the continuous density around each sample, so it keeps tracking how the buffer redistributes throughout training and separates methods cleanly. This too must be approximated, however: brute-force computation of the k-NN entropy on the full buffer costs $\mathcal{O}(N^2 d)$, where d is the goal dimensionality and N the buffer size, which is infeasible as the buffer grows. We therefore average the estimator over $n_{\text{rep}} = 5$ random subsamples of size $n_{\text{sub}} = 10^4$, making the cost independent of N. On each subsample,

$$\hat{H} = \frac{d}{m} \sum_{i=1}^{m} \log \rho_k(i) + \log(m-1) - \psi(k) + \log c_d, \tag{24}$$

where $\rho_k(i)$ is the Euclidean distance from s_i to its k-th nearest neighbour ($k = 3$), ψ is the digamma function, and $c_d = \pi^{d/2}/\Gamma(d/2+1)$ is the volume of the unit ball in $\mathbb{R}^d$. Observations are min-max normalized to $[0,1]^d$ using the observation-space bounds, so that no dimension dominates the Euclidean metric; the correction $\sum_{j=1}^{d} \log(u_j - \ell_j)$ recovers the entropy in the original units. For discrete actions, we report the joint entropy $\hat{H}(s,a) = \hat{H}(a) + \sum_a \hat{p}(a)\, \hat{H}(s \mid a)$, estimating each conditional $\hat{H}(s \mid a)$ on the corresponding action subset.

E Source Code, Compute Details, and Runtimes

Source code available at [**supplementary material, see attachments**].

We ran experiments on SLURM-based clusters, always saving all data and statistics available (e.g., visit and goal counts maps). Runs were parallelized whenever possible. For Gridworlds and Classic control, code is in PyTorch and ran on AMD Turin 9965 CPUs. For GCRL control, code is in JAX and ran on a mix of NVIDIA V100 (32 GB), A100 (80 GB) and GH200 (96 GB) GPUs. Operations are JIT-compiled and execute on a single GPU per run.

Wall-clock time (Table 2) varies with environment dimensionality and training steps. Note that the comparison is not like-for-like: AdaGoal, DISCOVER, and SUN Episodic score a larger batch of candidates once per episode (Section F), whereas SUN scores a smaller batch whenever its adaptive check fires — often early in training, and progressively less as the agent is trained (Section 4.2). Note that AdaGoal and DISCOVER run in comparable time to SUN despite training an ensemble of critics:

in PyTorch (Gridworlds and Classic control) they update a single randomly selected critic per step (Section F.1), and in JAX (GCRL control) the critic updates are parallelized.

F TRAINING HYPERPARAMETERS

Hyperparameters have not been tuned, with the exception of the pseudocount radius ρ (see Appendix H). For GCRL control, we used the official implementation of DISCOVER (TD3). For Gridworlds and Classic control, we implemented a simple version of DQN and used common hyperparameters without any tuning.

Replay buffer. Before training starts, the replay buffer is "warmed up" with data collected with a random-action policy. For Gridworlds and Classic control, we collect 10,000 samples (except `ThreeRoom`, where we collect 5,000). For GCRL control, we collect 1,000. The replay buffer then stores all samples collected until the end of training, i.e., no data is ever evicted.

Goal-selection. SUN and its ablated versions draw a pool of 256 candidates from the replay buffer at every re-selection. Episodic algorithms (SUN Episodic, DISCOVER, AdaGoal) instead draw 2500 candidates once, at the beginning of the episode.

Goal-reached. In GCRL tasks, the threshold η in Algorithm 1:7 is given by the environment. In Classic control, it is the bin width of the discretized counts used for the heatmaps. Gridworlds states are discrete and no threshold is needed.

Algorithm-specific. SUN's pseudocounts are computed using $\rho = 0.25$. DISCOVER's novelty coefficient is $\beta = 10$ (for Gridworlds and Classic control) and $\beta = 1$ (for GCRL control).

Table 2: **Average wall-clock runtime**, in minutes.

	ThreeR.	FourR.Stuck	GridMaze	M.Car	Pend.	L.Lander	L.Lander(Full)	Acrobot	C.Pole	P.Maze-S	P.Maze-H	A.Maze-S	A.Maze-H	ArmPush-H
SUN (Adapt., Multipl.)	76.4	164.6	268.4	211.2	106.3	457.9	1715.2	119.5	233.9	15.5	16.8	29.5	36.7	116.3
SUN (Adapt., Additive)	91.0	158.3	256.9	230.0	141.3	438.6	—	167.9	256.4	15.5	17.3	29.6	36.6	114.7
SUN (Per-Step, Multipl.)	102.6	215.7	284.2	247.4	149.2	471.5	—	167.4	294.3	15.3	16.4	29.3	36.6	115.0
SUN (Epis., Multipl.)	97.2	161.8	261.1	223.7	139.1	436.4	—	166.0	248.6	15.2	16.3	29.3	36.7	115.8
Reach.-Only	79.3	184.2	288.6	231.0	133.8	452.1	—	141.8	263.8	15.8	17.7	29.7	37.0	112.4
Novelty-Only	83.9	178.5	275.1	239.0	138.7	480.5	—	156.1	267.6	14.1	14.4	29.5	36.8	112.3
AdaGoal	94.9	225.3	292.0	225.5	119.2	493.7	1859.3	138.6	215.9	14.8	14.1	24.8	35.6	110.0
DISCOVER	96.7	222.1	270.7	213.6	107.6	533.2	2034.9	125.3	198.9	14.8	14.1	25.0	35.6	110.5
Random	7.7	15.4	33.8	18.9	11.0	53.7	247.1	10.7	22.6	10.9	12.4	18.4	26.5	102.0

Table 3: **Hyperparameters** used in our experiments.

Gridworlds and Classic control	
Discount factor γ	0.99
Trace factor λ	0.95
Target network copy frequency	1
Target network Polyak coefficient	0.001
Minibatch size	16
TD(λ) horizon T	16
Update frequency	1 step
Updates per step	1
Clip reward	False
Optimizer	AdamW
Learning rate	10^{-3}
Loss	Huber
Gradient norm clipped at	1.0

GCRL control	
Discount factor γ	0.99
Trace factor λ	0.95
Target network Polyak coefficient (critic)	0.005
Target network Polyak coefficient (actor)	$5{\times}10^{-7}$
Target networks copy frequency	2
Policy delay	2
Minibatch size	256
TD(λ) horizon T	3
Update frequency	1 step
Updates per step	1
Target-policy smoothing std	0.2
Target-action noise clip	0.5
Optimizer	AdamW
Learning rate	10^{-3}
Critic loss	MSE
Gradient norm clipped at	—

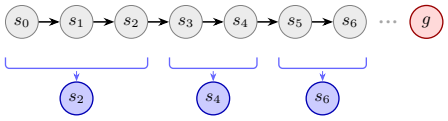


Figure 16: **Example of HER "segmented future" relabeling.** The trajectory is split into *segments*: a *future* cut timestep k is sampled, its state s_k (blue) is assigned as the goal to all timesteps in the segment, and the next segment starts at $k+1$. Here, the cuts fall at $2, 4, 6$.

F.1 GRIDWORLDS AND CLASSIC CONTROL

The goal is composed of a state component g_s (environment-specific, see Section D), and an action component g_a. That is, we explicitly learn SVFs whose goal is to perform specific actions in specific states. Because actions are discrete, we learn $Q^\theta(s, a, g_s, g_a)$: the Q-network takes state and the goal-state as input, and outputs the action-value for every goal-action.

Q^θ is trained with Double DQN (van Hasselt, 2010) with TD(λ) using Watkins's cutting traces (Watkins and Dayan, 1992) with two practical modifications motivated by the bounded scale of the SVFs in $[0, 1]$.[9] *First*, to mitigate overestimation bias, TD targets clip $\max_a Q^\theta(s', a, g_s, g_a)$ to 1. *Second*, we relax the strict $\arg\max$ used by Watkins's cutting: an action is considered greedy if its Q-value lies within $1-\gamma$ of the maximum. This tolerance matches the scale of one-step Bellman backups, and prevents traces from being cut too aggressively by neural network approximation error, to which a strict $\arg\max$ is overly sensitive.

The policy in Algorithm 1 is ε-greedy ($\varepsilon = 0.1$) with respect to the critic Q^θ. Action-level noise is needed with neural-network approximators, and is aligned with DISCOVER official implementation (with built-in noise via Gaussian perturbations on the actor's output, see below).

AdaGoal and DISCOVER use four critics, each with its own target network (DQN-style). At every update, one randomly selected critic is updated, using the target network of another randomly selected critic.[10] Their policy is ε-greedy with respect to one fixed critic per episode, selected randomly.

For training, we uniformly sample 16 batches from the replay buffer, and append all the following $T-1$ samples, for a total of 256 samples. These sequences may have samples from different consecutive trajectories, but are kept separate thanks to truncation flags. Then, we relabel goals with HER (Section C) with one modification. The original HER "future" strategy assigns each timestep its own future goal, yielding T sub-sequences of *varying* length. For more efficient batch training, we implement a "segmented" variant: we concatenate contiguous sub-sequences (each with its own future goal) so that their total length matches the original T (Figure 16). Relabeling is repeated 4 times per sequence, so each state is trained against 4 future goals. These samples are all positives; to balance them, we draw 4 random negatives from the replay buffer (one per full sequence of length $T = 16$). As a result, each DQN update uses $16 \times 16 \times 4 \times 2 = 2,048$ data points.

F.2 GCRL CONTROL

We build on DISCOVER official implementation, and learn $V^\theta(s, g_s)$ and a goal-conditioned policy $\pi(a \mid s, g_s)$. The action (continuous) is not part of the goal. V^θ is trained with TD3 (Fujimoto et al., 2018) with TD(λ) targets *without* Watkins's cutting traces (with continuous actions, $\arg\max$ is infeasible) and *without* clipping $V^\theta(s', g_s)$ to 1. Once the goal is selected, the policy explores with noise $\mathcal{N}(0, 0.4)$ added to the action, as in the official DISCOVER implementation.

AdaGoal and DISCOVER use an ensemble of four critics, each with its own target network (DQN-style). All are trained at every update, with different random batches, and with TD targets from its own target network. There is one policy, trained against the average value returned by all critics.

All algorithms relabel goals with HER (Section C), without modification. At each update, a batch of 256 samples is drawn uniformly from the replay buffer. For a randomly selected half, we sample a future goal from within the next $T_{\max} = 50$ steps of the original trajectory (or until termination, whichever comes first) and compute TD(λ) targets up to that goal. For the other half, we sample the relabel goal uniformly from the buffer and compute one-step TD targets.[11] Note that, unlike in DQN, the intermediate steps between a batch point and its relabeled goal are used only to compute the TD(λ) target, *not* for gradient updates. Each TD3 update therefore uses exactly 256 data points.

[9] When $(g_s, g_a) = (s, a)$ the reward in Eq. (1) is 1 and the transition is terminal, otherwise the reward is 0.

[10] Updating all critics at every step performed worse on average and increased wall-clock time by $\times 3.5$.

[11] The original DISCOVER implementation uses one-step TD targets rather than TD(λ). In our experiments, however, eligibility traces improved the performance of all algorithms.

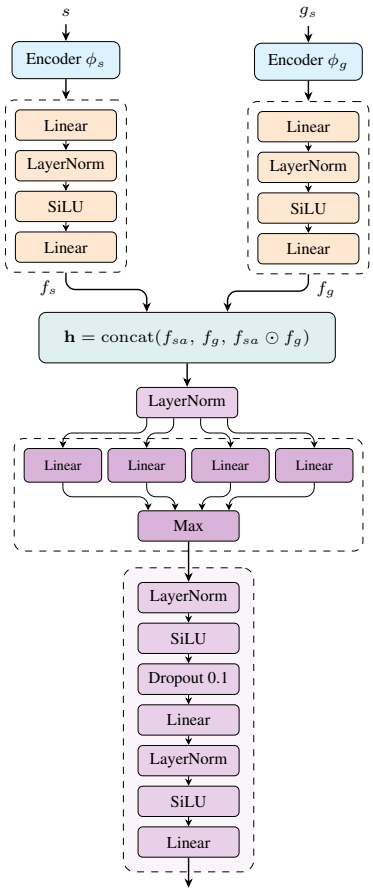


Figure 17: **Network architecture.**

In $Q^\theta(s, a, s_g, a_g)$, linear layers size is 64 and the network output shape is $|\mathcal{A}|\times|\mathcal{A}|$ — state-action value for every goal-action in a given goal-state. In $V^\theta(s, s_g)$ and $\pi(a \,|\, s, s_g)$, linear layers size is 256 and output shapes are 1 and $|\mathcal{A}|$, respectively. The two networks do not share any layer.

Encoder. Gridworlds do not need an encoder because their observation is already appropriate (one-hot encoding of the agent's position). In continuous control environments, the state and state-goal encoders are a radial basis function layer (Poggio and Girosi, 1990). The layer places C centers $\mu_1, \ldots, \mu_C$ per input dimension, initialized uniformly between $v_{\min}$ and $v_{\max}$, and learns both the centers and bandwidths $h_c > 0$ end-to-end via gradient descent, jointly with the rest of the network. For each scalar input x, the layer computes Gaussian activations $\varphi_c(x) = \exp(-0.5((x - \mu_c)/h_c)^2)$. To prevent vanishing gradients for inputs outside $[v_{\min}, v_{\max}]$, we add a linear leak to the boundary tiles, so that the activation grows linearly with distance once x falls below μ_1 or above μ_C. We use $C = 20$ centers initialized with $v_{\min} = -3$, $v_{\max} = 3$. Environment raw observations are standardized using running mean and standard deviation tracked with Welford's online algorithm.

Feature fusion. We combine the state and goal features by concatenating f_s, f_g, and their element-wise product $f_s \odot f_g$. The element-wise product provides a multiplicative interaction term that makes pairwise alignment between corresponding components of f_s and f_g directly available to the downstream layers, complementing the information carried by the concatenation of f_s and f_g alone.

Maxout. After feature fusion we apply a *Maxout* unit (Goodfellow et al., 2013), which computes four parallel linear projections of its input and takes the element-wise maximum across them.

F.3 Network Architecture

Figure 17 shows the network architecture used by all algorithms. We observed that the following components improved the performance of *all* algorithms.

- The Gaussian encoding helped mildly in all environments but significantly in `LunarLander`. We suspect this is due to `LunarLander`'s distinctive observation space: it is unbounded, asymmetric, and its elements have differing scales.
- Feature fusion helped significantly in all environments. Concatenation alone forces the network to learn the interaction between state and goal from scratch, whereas the element-wise product $f_s \odot f_g$ makes the alignment of corresponding features directly available to the downstream layers.
- Having separate layers before the fusion was necessary. Encoding state and goal with separate streams and combining them late is standard in goal- and task-conditioned value functions (Borsa et al., 2019; Eysenbach et al., 2022). One may be tempted to use a single shared set of layers when the observation space and the goal space are identical (e.g., in Gridworlds, `Pendulum`, or `MountainCar`). Yet this made learning accurate SVFs significantly harder, especially in Gridworlds where some states are unreachable. With shared layers, states and goals are mapped by the same function, so two tiles that look similar receive similar representations regardless of which is the agent's position and which the target. The network then generalizes reachability across them, and assigns non-zero value to goals that are near in observation space but unreachable in the environment.
- SiLU activation performed better than ReLU and LeakyReLU in all environments.
- Combining LayerNorm with near-zero initialization and no bias in all but the last linear layer improved stability, in line with the findings of Elsayed et al. (2026).

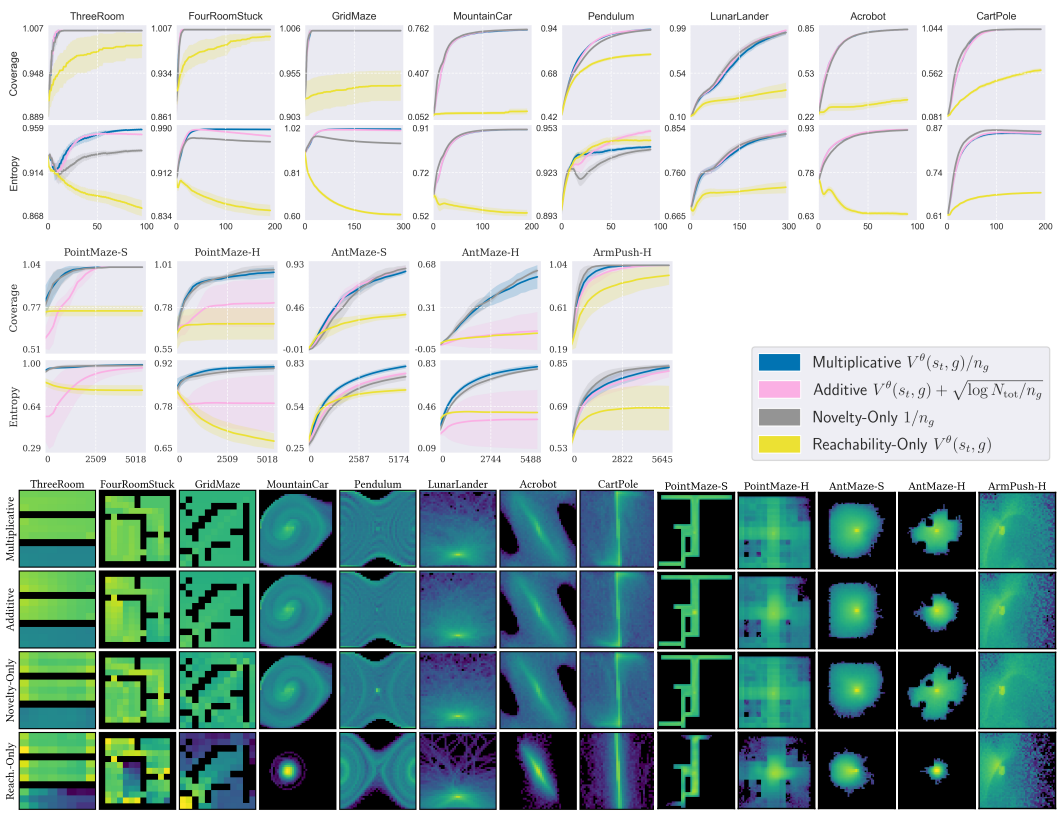


Figure 18: **Indicator ablation.** Full version of Figure 6 with heatmaps.

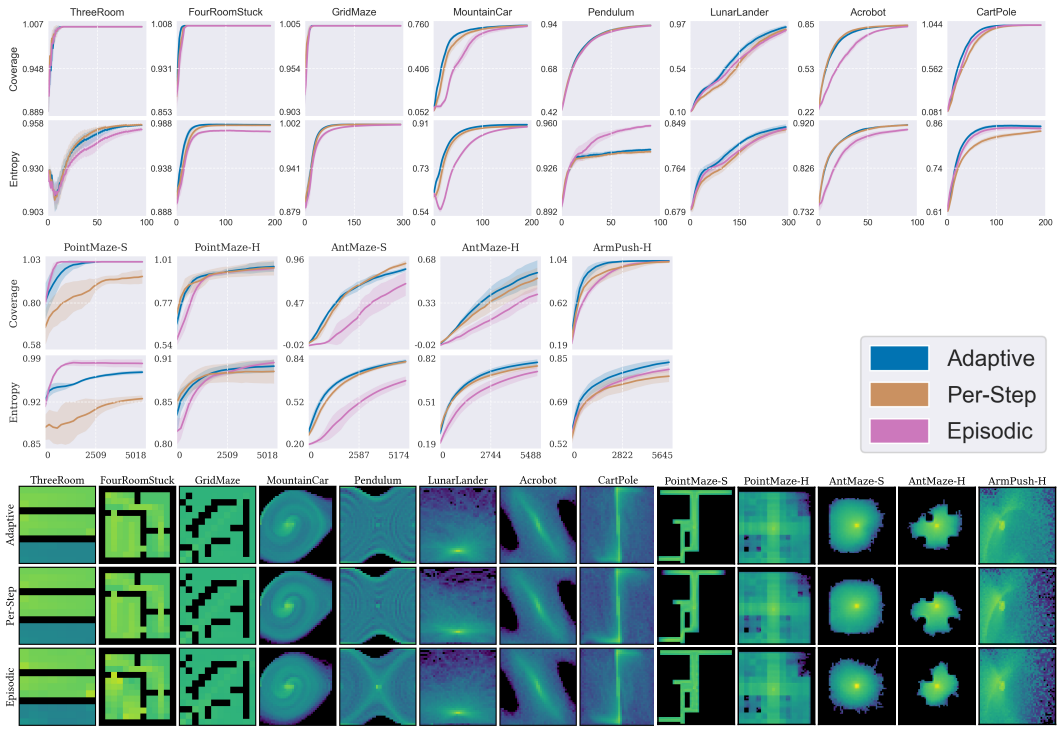


Figure 19: **Goal-selection ablation.** Full version of Figure 7 with heatmaps.

G Indicator and Goal-Selection Ablations

Figures 18 and 19 report the full training curves and end-of-training visitation heatmaps for the ablations discussed in Section 4.2. We exclude `LunarLander(Full)` due to its computational expense. Per-environment goal-selection heatmaps with additional statistics are in Appendix I

The ablations exhibit the same trends described in Section 4.2, with SUN performing the best. The only exceptions are `Pendulum` and `PointMaze-S`. In the former, SUN Additive performs best, because its novelty term collapses as counts grow and the indicator reduces to pure reachability: the agent then stops targeting the rare states that can only be visited by repeatedly traversing the same states, which is what drives SUN's entropy down in Section 4. In the latter, SUN Episodic is the best. `PointMaze-S` is a small environment where most goals can be reached within a third of the episode, so committing to a goal and then exploring randomly around it, as episodic selection does, is effective. The same strategy is detrimental where goals are hard to reach or random actions barely move the agent: in `AntMaze` it is the worst of the three.

H Pseudocounts Ablation

Here we evaluate how ρ affects the pseudocount approximation (Section B). Figure 20 shows SUN performance for varying ρ against *true counts*, i.e., counts obtained by discretizing the goal space with 50 bins per dimension. Figure 21 shows instead pseudocounts computed *on the same data* (collected while training with *true counts*). Results show that our pseudocounts closely track the coverage and entropy of training with true counts across all environments for most values of ρ, except for values that are too small. This is expected: the estimator is a uniform-kernel density estimate and ρ is its bandwidth, so an extreme value in either direction makes $\nu(g) = 1/n_g$ constant across candidates.

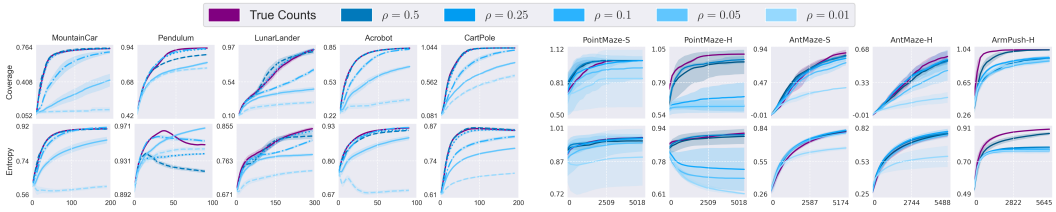


Figure 20: **Pseudocounts vs. true counts.** Gridworlds are omitted, since there pseudocounts coincide with the counts. True counts improve coverage and entropy in almost all environments. One exception is `Pendulum`, where entropy drops sharply, the same behavior discussed in Section 4.1, amplified by exact counting. Another is `PointMaze-S`, where $\rho = \{0.25, 0.5\}$ attains the best entropy and the fastest coverage growth, although all radii reach the same terminal coverage. We attribute this to the narrow corridors of this maze, which make true counts sensitive to the bin discretization.

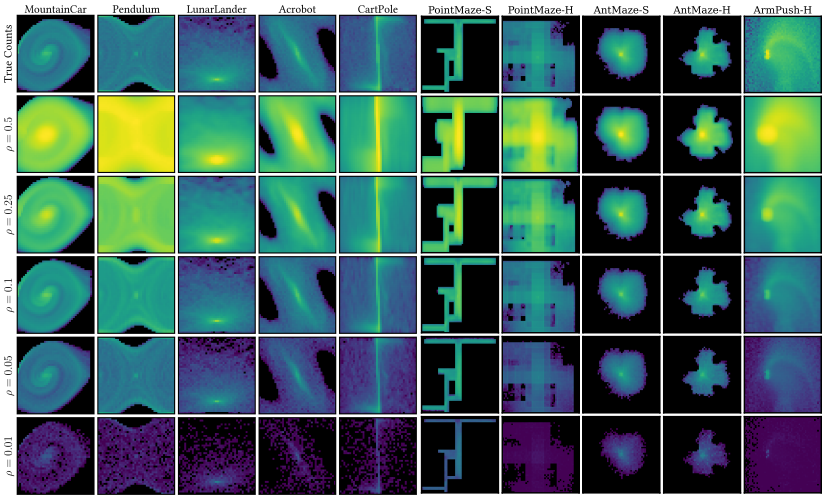


Figure 21: **Log-scale pseudocount heatmaps.** The top row shows the true visit counts at the end of training (first seed). Pseudocounts (rows below) are computed on that same data at decreasing radii ρ. At any radius, pseudocounts preserve the shape of the visited set (e.g., `MountainCar` spiral).

When ρ is too small, the ball around an inserted sample rarely contains any other buffer entry, so most samples end up with small *uniform* counts. When ρ is too large, the opposite degeneracy occurs: the ball contains a large fraction of the buffer for every entry, so counts become comparable across the visited set. In both cases, the $\arg\max$ of Eq. (2) reduces to an $\arg\max$ over the SVF alone, which is sensitive to approximation noise when V^π is a neural network, especially early in training.

In practice only the first regime is reached. Across the suite, entropy and coverage degrade as ρ decreases, and the corresponding heatmaps in Figure 21 show zero-count areas at the smallest radii, while performance is largely flat for $\rho \geq 0.1$. This is a consequence of the robust per-feature scale of Section B: because each dimension is normalized by its own interquartile range, a ball of radius ρ covers a comparable fraction of the support in every environment, and even $\rho = 0.5$ preserves the shape of the visited set — the mazes' corridors, the spiral of `MountainCar`, the arc of `ArmPush-H`.

I Detailed Analysis Of All Environments

For each environment, we report the following goal-related statistics, characterizing the behavior of SUN (with adaptive, episodic, and per-step goal-selection), AdaGoal, and DISCOVER.

- **Goals selected and reached**, measured at four stages of training (25/50/75/100% of the training steps). All heatmaps share the same log-scale color range.
- **Goal success**: the ratio of goals reached to goals selected.
- **Steps-to-goal**: the average number of steps taken to reach a goal.
- **Goal reselections** (SUN Adaptive and Per-Step): the fraction of steps at which $g_t \neq g_{t-1}$, normalized to $[0, 1]$ by the total number of steps. The two differ in what triggers a reselection. In *adaptive* SUN, the goal is discarded and a fresh candidate batch is sampled only when $V^\theta(s_t, g_t) < V^\theta(s_{t_{\text{sel}}}, g_t)$ fires. In *per-step* SUN, a fresh batch is sampled at every step, and the goal changes if the batch has a candidate with a higher SUN score (Eq. (2)) than the current one.
- **Random exploration** (SUN Episodic, AdaGoal, DISCOVER): the fraction of steps at which the agent acts randomly. In *episodic* goal-selection, this is triggered once the agent reaches its goal, following the original DISCOVER implementation.

The following important trends emerge.

- Episodic selection fails when goals can become unreachable mid-episode (Figures 23 and 24).
- In-episode reselection avoids this problem, but Per-Step reselects far more often than Adaptive, which hurts performance. In almost all environments, Adaptive's reselection rate drops to zero over training, indicating that the agent has learned to reach the goals it commits to.
- DISCOVER is biased toward reachability: across all heatmaps it selects far more goals near the starting states than the other algorithms, and attains high goal success from the very beginning. This may stem from its coefficient β, which must balance the scales of the reachability and novelty terms — a problem SUN sidesteps entirely thanks to its multiplicative indicator.
- AdaGoal is biased toward novelty at the expense of reachability: across all heatmaps it selects mostly goals far from the starting states, "skipping" intermediate ones, and in Figure 22 it often selects unreachable goals. This is not unexpected. As noted in Section 3.4, we evaluate the deep-RL variant of AdaGoal, which — unlike the tabular version — does not constrain goal selection by the estimated goal-hitting time. Tarbouriech et al. argue that this is approximated implicitly by the disagreement of the value ensemble. Our results suggest that this does not always hold. Novelty alone is a reasonable signal for goals the agent can actually reach, since visiting them resolves the disagreement at little cost. Unreachable goals, however, resolve only after enough failed attempts for every ensemble member to recognize them as such, and each of those attempts is an episode without novel data. The rule cannot distinguish the two cases, and where the reachable set is a small fraction of the goal space the latter dominates — precisely the behavior the reachability constraint was meant to prevent.

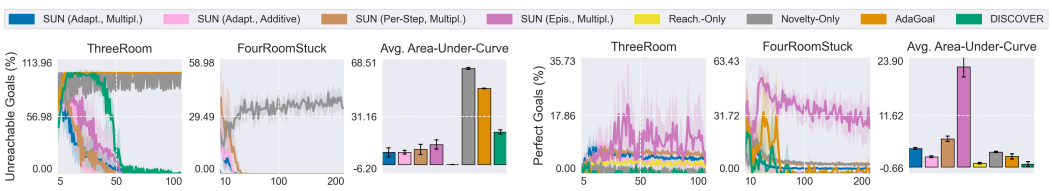
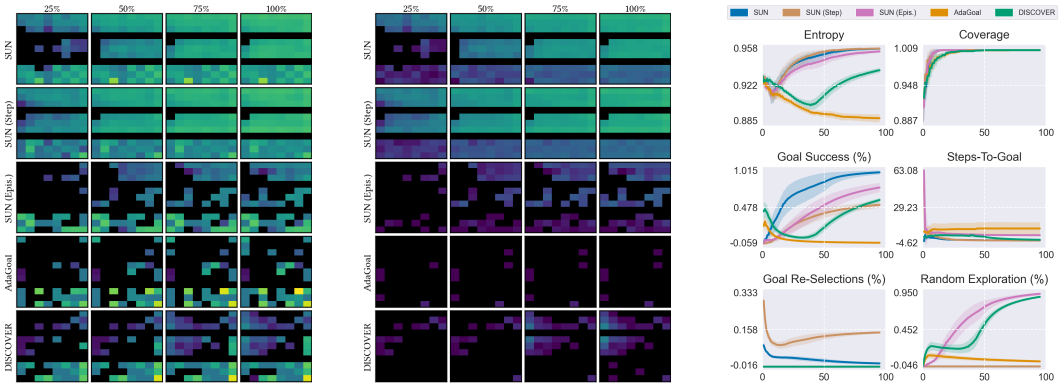


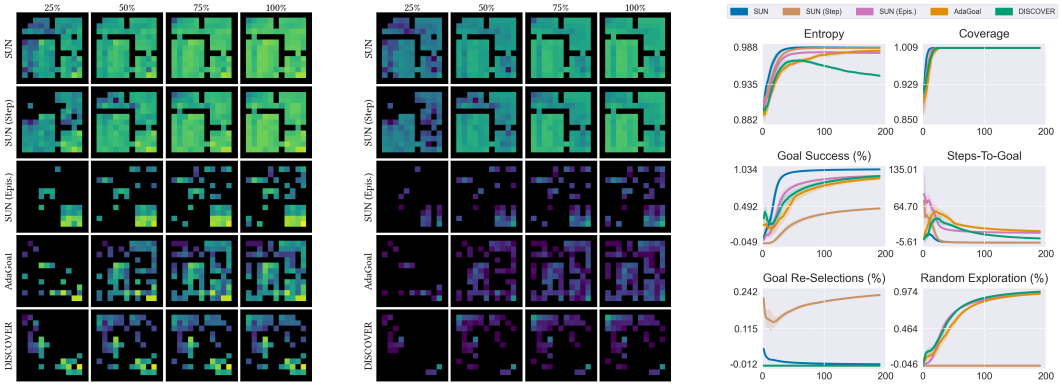
Figure 22: **Selection rates of unreachable and perfect goals** in `ThreeRoom` and `FourRoomStuck`.

- In Gridworlds, SUN tends to select short-horizon goals, as shown by steps-to-goal tending to one. In control tasks the opposite happens: the metric grows over training, meaning SUN selects goals that are progressively further away. This is expected, since control tasks require coherent action sequences to reach distant states — e.g., in `MountainCar` the agent must build up momentum.
- This is consistent with Figure 22, where SUN learns to never select unreachable goals yet still selects *"imperfect"* ones. This is not unexpected, because a "perfect" goal is the least visited one regardless of distance ($\arg\min_g \{n_g : V^*(s_t, g) > 0\}$, where V^* is the *optimal* SVF), whereas SUN weighs novelty against reachability ($\arg\max_g V^\theta(s_t, g)/n_g$). SUN Episodic selects the most perfect goals of all versions, but this does not translate into better exploration: a goal being reachable in principle does not mean the agent can reach it now, and the least-visited reachable goal is typically the farthest one. Episodic selection commits to it for the rest of the episode, whereas Adaptive discards it as soon as the SVF value drops and replaces it with a closer one.



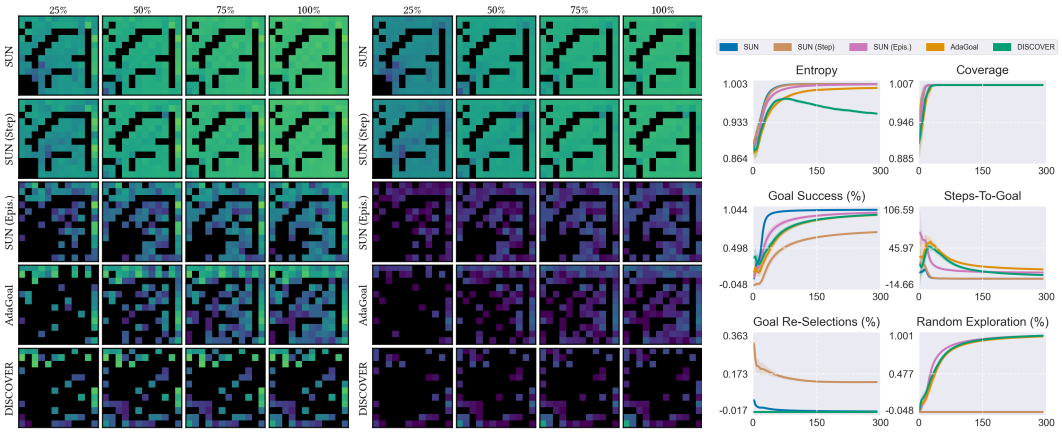
(a) Goals selected. (b) Goals reached. (c) Training curves.

Figure 23: `ThreeRoom`. This environment shows the superiority of the SUN indicator, and the importance of properly trading off novelty and reachability. AdaGoal fails because it frequently selects unreachable goals: in (a), it is the only method that does *not* under-select the bottom room (which is often unreachable). DISCOVER, conversely, is biased toward reachability: it selects fewer goals far from the starting states (left of the rooms). The curves in (c) confirm this: AdaGoal has the lowest success rate, while DISCOVER reaches a high success rate almost immediately — it selects easily-reachable goals early on — and therefore spends most of its time exploring randomly. All three versions of SUN, in contrast, achieve high entropy: over time they select goals uniformly and progressively further to the right (away from starting states).



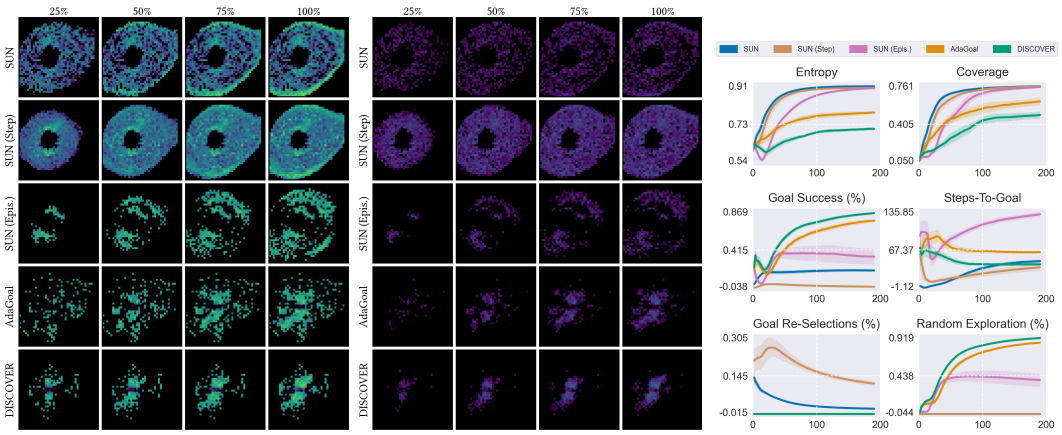
(a) Goals selected. (b) Goals reached. (c) Training curves.

Figure 24: `FourRoomStuck`. SUN Episodic, DISCOVER, and AdaGoal perform poorly: the agent can accidentally get trapped in the bottom-left room, at which point the previously selected goal becomes unreachable yet stays fixed for the rest of the episode. SUN Adaptive and Per-Step avoid this through goal reselection, and both attain high entropy. Per-Step entropy increases more slowly, though: it reselects goals far more often than Adaptive, making it unstable early in training.



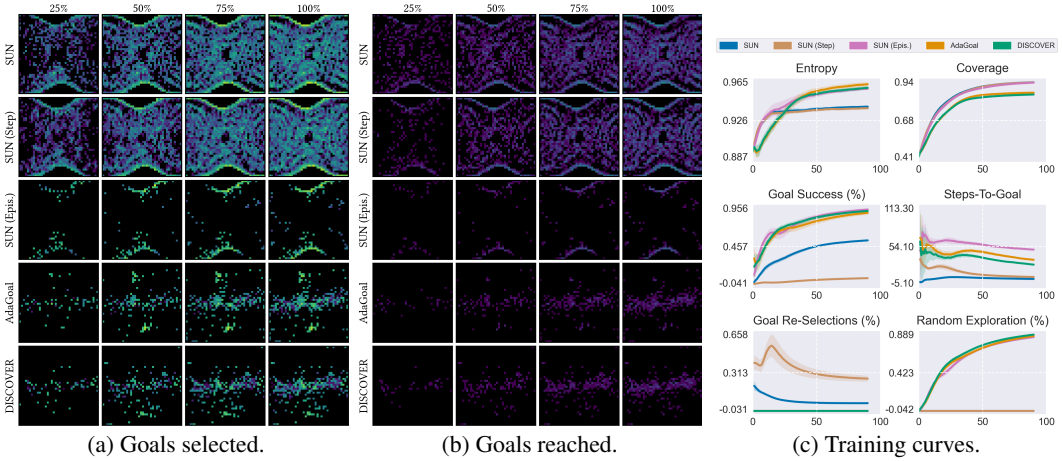
(a) Goals selected. (b) Goals reached. (c) Training curves.

Figure 25: `GridMaze`. Similar trends to Figure 24 emerge, e.g., DISCOVER bias towards easy-to-reach goals and Per-Step higher reselection rate.



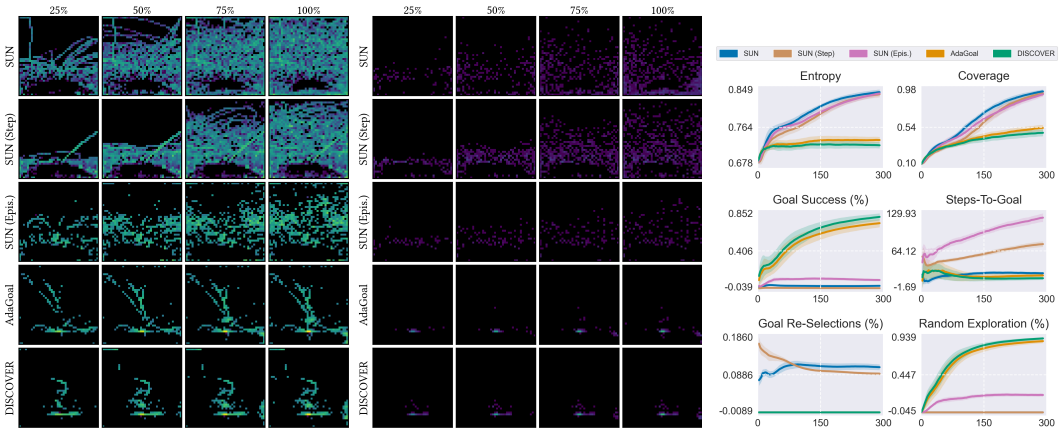
(a) Goals selected. (b) Goals reached. (c) Training curves.

Figure 26: `MountainCar`. The main difference from Gridworlds is that SUN's steps-to-goal increases over training. This is expected, as `MountainCar` requires coherent action sequences to build up the momentum needed to escape the valley. This environment further highlights DISCOVER's reachability bias, as most of its goals lie close to the agent's starting position.



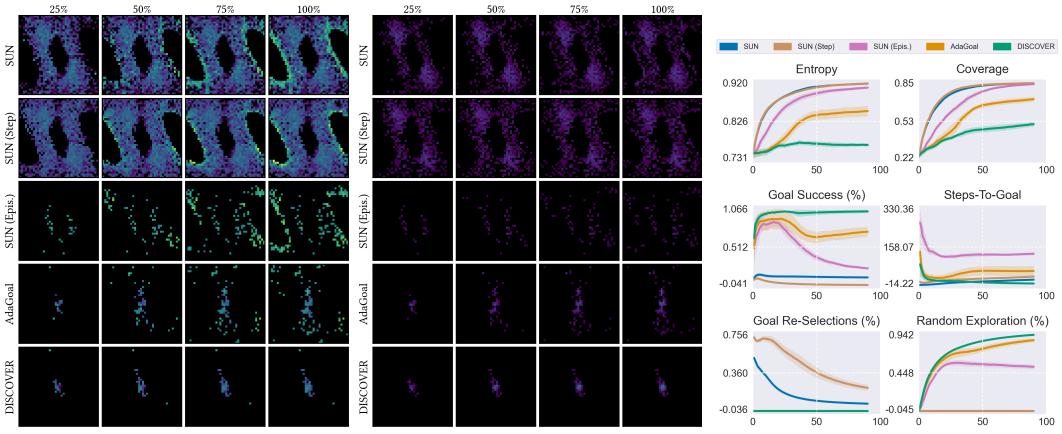
(a) Goals selected. (b) Goals reached. (c) Training curves.

Figure 27: `Pendulum`. As discussed in Section 4, the distinctive trait of this environment is the presence of hard-to-reach goals that can only be visited by repeatedly traversing easy-to-reach ones, causing entropy to decay even as coverage increases.



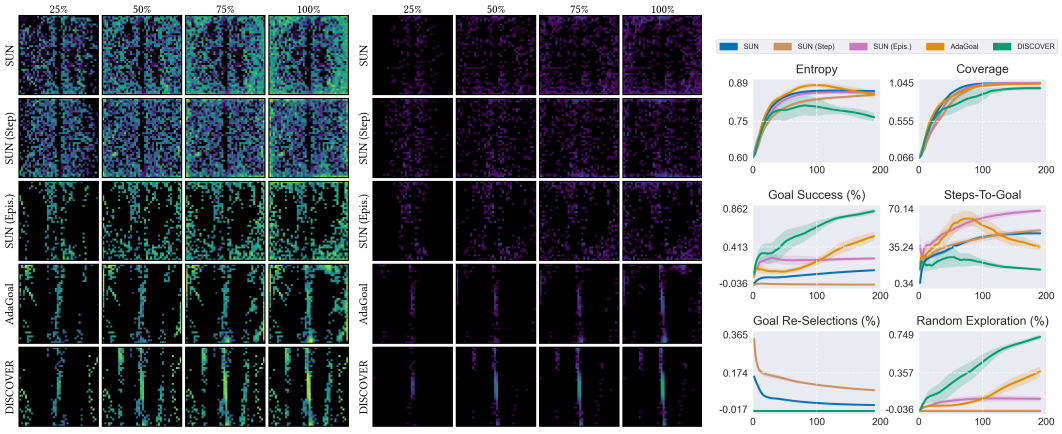
(a) Goals selected. (b) Goals reached. (c) Training curves.

Figure 28: `LunarLander`. Among the Classic control tasks, this is the most challenging: it has harder dynamics, higher dimensionality, and a larger goal space — reasons why SUN's reselection rate does not tend to zero. The difference between SUN and AdaGoal/DISCOVER is more apparent here: SUN's selected goals are spread more uniformly across the whole space. DISCOVER selects mostly goals near the starting state, whereas AdaGoal often picks goals too far away for the agent to reach (e.g., the isolated goal at the top of its heatmaps), confirming its bias toward novelty.



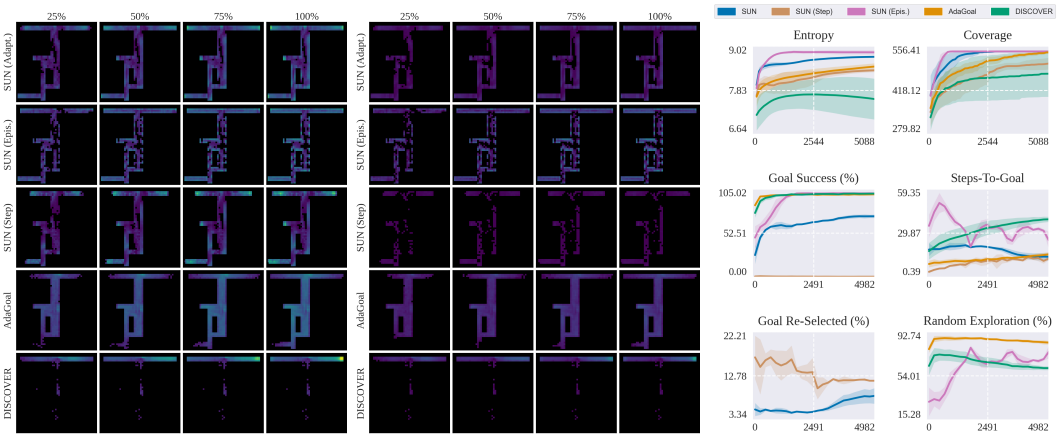
(a) Goals selected. (b) Goals reached. (c) Training curves.

Figure 29: `Acrobot`. All trends discussed so far emerge here as well, most notably the different spread of goals selected by SUN compared to AdaGoal and DISCOVER.



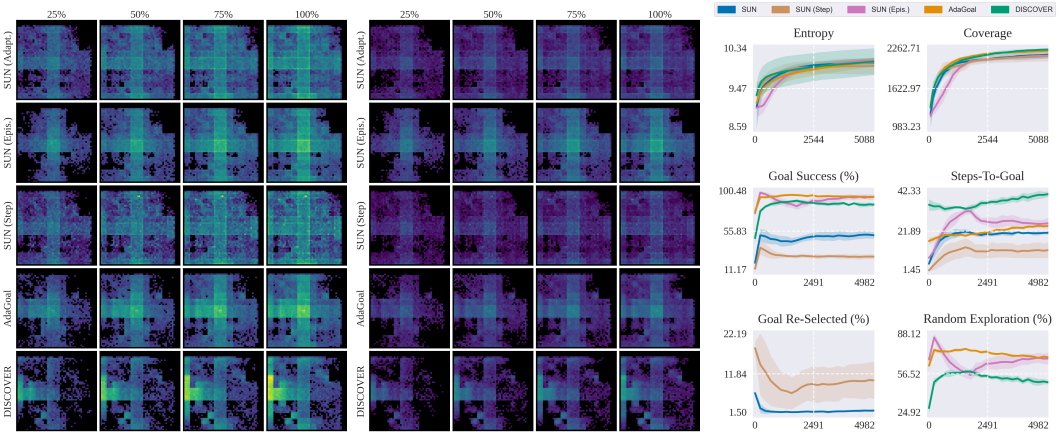
(a) Goals selected. (b) Goals reached. (c) Training curves.

Figure 30: `CartPole`. All trends discussed so far emerge here as well.



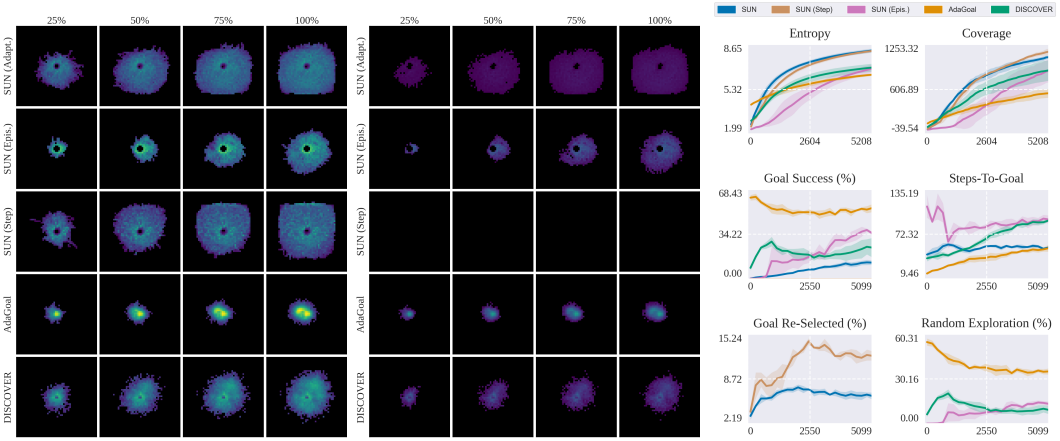
(a) Goals selected. (b) Goals reached. (c) Training curves.

Figure 31: `PointMaze-S`.



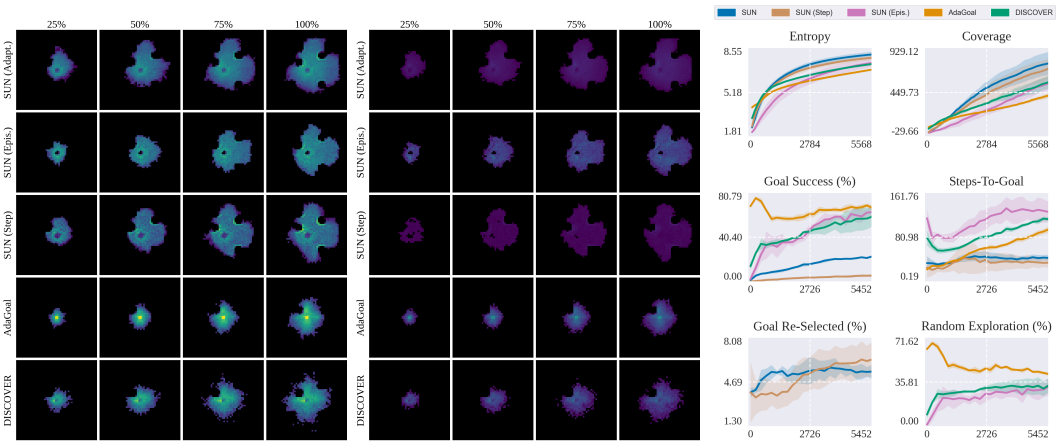
(a) Goals selected. (b) Goals reached. (c) Training curves.

Figure 32: `PointMaze-H`.



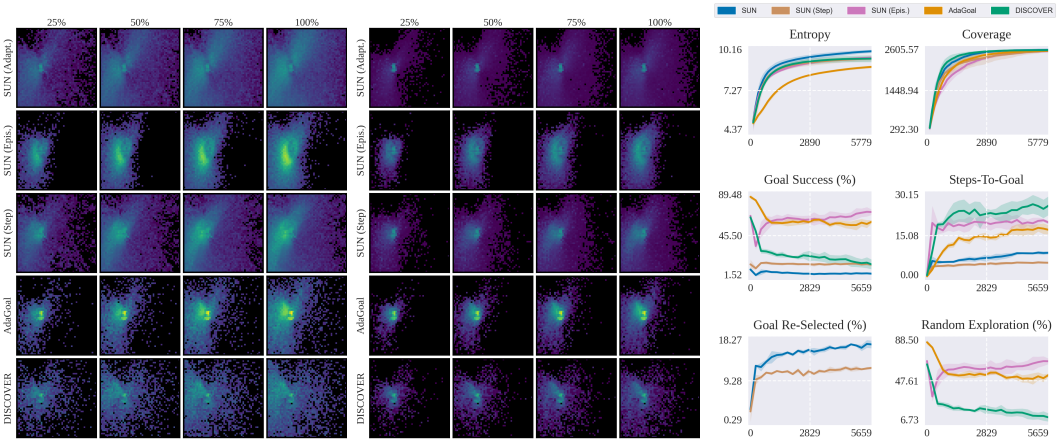
(a) Goals selected. (b) Goals reached. (c) Training curves.

Figure 33: `AntMaze-S`.



(a) Goals selected. (b) Goals reached. (c) Training curves.

Figure 34: `AntMaze-H`.



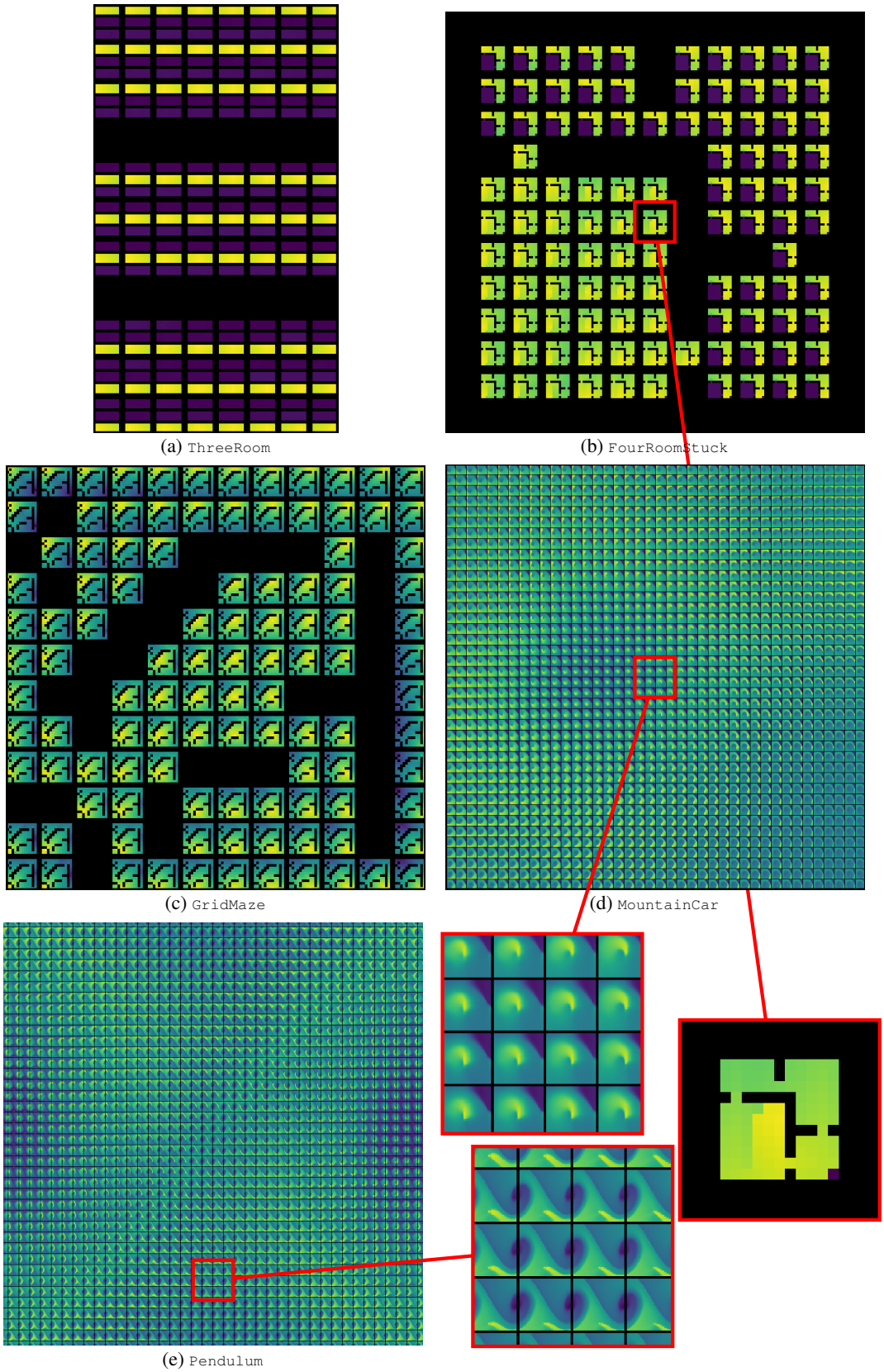
(a) Goals selected. (b) Goals reached. (c) Training curves.

Figure 35: `ArmPush-H`. Reached goals concentrate near the cube's start region (the red/blue area in Figure 13). To move it, the agent must first make contact with it.

J Successor Value Function Visualization

In Figure 36, we visualize the SVFs learned by SUN, showing that they are indeed accurate. This is possible only for environments with two-dimensional state and goal spaces, and for `Pendulum`, whose sine/cosine state can be transformed into an angle. Note that the agent learns $Q^\theta(s, a, g_s, g_a)$, and we visualize $V^\theta(s, g_s) = \max_{a,g_a} Q^\theta(s, a, g_s, g_a)$.

Each heatmap is composed of many sub-heatmaps, one per goal state. For example, the magnified region of `FourRoomStuck` shows $V^\theta(s, g_{\text{middle}})$, the value of reaching the middle tile from every other tile. The pattern is clear: states near the goal have higher value. Zooming in other sub-heatmaps, one can see that tiles inside the bottom-left room have almost-zero value for goals outside it, reflecting the room's irreversible transitions. The bottom-right tile also has almost-zero value in all heatmaps, as it is a terminal state. Goals corresponding to walls are never visited but occasionally take non-zero value due to their proximity to reachable goals. Similar patterns appear across all heatmaps.



(a) `ThreeRoom` (b) `FourRoomStuck` (c) `GridMaze` (d) `MountainCar` (e) `Pendulum`

Figure 36: **SVFs** V^{θ} learned by SUN. Each figure has many sub-heatmaps, one per goal state (see magnified regions). In `MountainCar` and `Pendulum`, axes are position (x) and velocity (y).

K SUN-UCB: A STRUCTURAL CONNECTION TO PAC ANALYSIS

The structural results in Appendix A characterize SUN's selection rule under an oracle but do not provide sample-complexity guarantees. Here we sketch how a UCB-like variant of SUN (with additive reachability and novelty) connects to the PAC framework of Tarbouriech et al. (2022). We state the algorithm, prove its unreachability filter sound, and show that its selection rule requires the two terms to be rescaled against each other (Remark 3) — the tabular counterpart of the design argument in Section 3.1. We do *not* claim a complete sample-complexity result. Two components are missing. First, the observed-transition filter is sound but not complete: it never selects an unreachable goal, but nothing in the algorithm guarantees that every L-reachable goal eventually enters it, and a passive filter admits initialization failures in which the reachable set never grows. A rigorous bound would require an explicit frontier-expansion mechanism with a discovery guarantee, as in Lim and Auer (2012) and Tarbouriech et al. (2020). Second, our concentration statement bounds the deviation of the empirical mean from the average value of the executed policies, which does not by itself certify near-optimality of the returned policies. We therefore present the analysis as a structural connection rather than a proof, and leave the complete argument to future work.

K.1 SETUP

We consider tabular MDPs $\langle \mathcal{S}, \mathcal{A}, P, s_0 \rangle$ with finite spaces $|\mathcal{S}| = S$ and $|\mathcal{A}| = A$, and stochastic transitions $P(s' \mid s, a)$. We adopt all assumptions in Assumption 1 except (C2): we relax deterministic dynamics to stochastic with almost-sure hitting (as in Section A.6). All counts n_g are exact (tabular setting). Under first-hit termination (C4), the SVF $V^\pi(s, g) = \mathbb{E}_\pi[\gamma^{\tau_g} \mathbf{1}_{\{\tau_g < \infty\}}] \in [0, 1]$.

We use $\delta \in (0, 1)$ as the confidence parameter: the statements below hold with probability at least $1 - \delta/3$.

Let $\mathcal{G}_L \triangleq \{g \in \mathcal{S} : \exists \pi \text{ s.t. } \mathbb{E}_\pi[\tau_g] \leq L\}$ be the set of L-reachable goals, with $|\mathcal{G}_L| \leq S$. We set the episode length equal to the reachability horizon L, as in AdaGoal, and reset to s_0 at the end of each episode.

K.2 SUN-UCB ALGORITHM

Estimators. For each goal g, let $n_t(g)$ be the number of episodes up to time t in which g was the selected goal, and let $\hat{V}_t(s_0, g)$ be the empirical mean of the returns collected in those episodes, with $\hat{V}_t(s_0, g) \triangleq 0$ when $n_t(g) = 0$. Define the uncertainty

$$U_t(g) \triangleq \min\left\{1, \sqrt{\frac{\log(SAT/\delta)}{n_t(g)}}\right\}, \qquad U_t(g) \triangleq 1 \text{ when } n_t(g) = 0. \tag{25}$$

The cap and the convention at $n_t(g) = 0$ are well defined because $V^\pi(s_0, g) \in [0, 1]$ under first-hit termination, so no uncertainty larger than the value range is informative.

Truncation. An episode targeting g contributes the sample γ^{τ_g} if g is hit at some step $\tau_g \leq L$, and 0 otherwise. The return is therefore truncated at the episode horizon, and $\hat{V}_t$ estimates $\mathbb{E}[\gamma^{\tau_g} \mathbf{1}_{\{\tau_g \leq L\}}]$ rather than $V^\pi(s_0, g)$, a downward bias of at most γ^{L+1}. Taking $L \geq \log(2/\varepsilon)/\log(1/\gamma)$ bounds this by $\varepsilon/2$, which is absorbed into the accuracy target of Conjecture 1.

Empirical reachability filter. Let $E_t \triangleq \{(s, s') : \exists a \text{ s.t. } N_t(s, a, s') > 0\}$ be the set of transitions observed up to time t, and let

$$\mathcal{G}_t^{\text{reach}} \triangleq \{g \in \mathcal{S} : g \text{ is reachable from } s_0 \text{ along edges of } E_t\}. \tag{26}$$

The filter is computed from observed transitions, not from value estimates. This matters: a filter of the form $\{g : \hat{V}_t(s_0, g) > 0\}$ would exclude every goal that has never been *hit*, not merely every goal that is *unreachable*, and would therefore block exploration toward the frontier. A goal that has been observed but never targeted lies in $\mathcal{G}_t^{\text{reach}}$ and remains selectable.

At step t, given s_t and $n_t(\cdot)$:

1. Update $\hat{V}_t$, n_t, and E_t from the last episode.

2. Compute the filter $\mathcal{G}_t^{\text{reach}}$ of Eq. (26).
3. Select

$$g_t \;=\; \arg\max_{g \in \mathcal{G}_t^{\text{reach}}} \hat{V}_t(s_0, g) \;+\; \beta\, U_t(g), \qquad \beta = 2/\varepsilon. \tag{27}$$

4. Run UCBVI (Azar et al., 2017) targeting g_t for one episode of length L; record whether g_t was hit and at which step.

The filter in step 2 is the algorithmic counterpart of unreachability rejection (Theorem 3). The coefficient β in step 3 is not cosmetic: Lemma 3 fails without it, for reasons discussed in Remark 3.

K.3 Conjectured Sample Complexity

Conjecture 1 (Sample complexity of SUN-UCB)**.** *Let $\varepsilon \in (0,1)$ and $\delta \in (0,1)$. Suppose SUN-UCB is augmented with a frontier-expansion mechanism guaranteeing that every $g \in \mathcal{G}_L$ enters $\mathcal{G}_t^{\text{reach}}$ within $\tilde{\mathcal{O}}(L^3SA/\varepsilon^2)$ steps. Then, with probability at least $1-\delta$, it returns goal-conditioned policies $\hat{\pi}$ satisfying $V^{\hat{\pi}}(s_0, g) \geq V^\star(s_0, g) - \varepsilon$ for all $g \in \mathcal{G}_L$ after at most $T = \tilde{\mathcal{O}}(L^3SA/\varepsilon^2)$ exploration steps.*

K.4 Partial Analysis

We prove four components — concentration of $\hat{V}$, soundness of the reachability filter, uncertainty of the selected goal, and a pigeonhole over goals inside the filter — and then identify what a complete argument would additionally require.

Step 1: Concentration.

Lemma 1 (Concentration)**.** *There exists an event $\mathcal{E}_1$ of probability at least $1-\delta/3$ on which, for every $g \in \mathcal{S}$ and every t with $n_t(g) \geq 1$,*

$$\left|\hat{V}_t(s_0, g) - \bar{V}_t(s_0, g)\right| \;\leq\; U_t(g), \tag{28}$$

where $\bar{V}_t(s_0, g)$ is the mean of the truncated values $\mathbb{E}_{\hat{\pi}_j}[\gamma^{\tau_g}\mathbf{1}_{\{\tau_g \leq L\}}]$ over the $n_t(g)$ episodes j in which g was the selected goal.

Proof. The samples take values in $[0,1]$, and the j-th sample has conditional mean $\mathbb{E}_{\hat{\pi}_j}[\gamma^{\tau_g}\mathbf{1}_{\{\tau_g \leq L\}}]$ given the history preceding episode j. The centered samples therefore form a bounded martingale difference sequence with respect to the filtration generated by the history, and Azuma–Hoeffding gives

$$\Pr\left[\left|\hat{V}_t(s_0, g) - \bar{V}_t(s_0, g)\right| > \sqrt{\frac{\log(SAT/\delta)}{n_t(g)}}\right] \;\leq\; \frac{\delta}{3ST} \tag{29}$$

for a fixed pair (g, t), after adjusting constants inside the logarithm. A union bound over the at most S goals and T steps yields $\mathcal{E}_1$. The deviation is also trivially bounded by 1, since both quantities lie in $[0,1]$, which justifies the cap in Eq. (25). □

Note that $\bar{V}_t$ is a historical average over the policies actually executed, not the optimal value $V^\star(s_0, g)$ nor the value of the returned policy. Bridging that difference is one of the two gaps discussed below.

Step 2: Soundness of the reachability filter.

Lemma 2 (Filter soundness and monotonicity)**.** *Deterministically, for every t:*

(i) $\mathcal{G}_t^{\text{reach}} \subseteq \{g \in \mathcal{S} : g \text{ is reachable from } s_0\}$;
(ii) $\mathcal{G}_t^{\text{reach}} \subseteq \mathcal{G}_{t+1}^{\text{reach}}$.

Proof. (i) Every edge $(s, s') \in E_t$ was traversed by the agent, so $P(s' \mid s, a) > 0$ for the action a that produced it. A path from s_0 to g using only edges of E_t is therefore a path of positive probability in the true MDP, and g is reachable. (ii) $N_t(s, a, s')$ is non-decreasing in t, hence $E_t \subseteq E_{t+1}$ and reachability along E_t implies reachability along E_{t+1}. □

SUN-UCB therefore *never selects a truly unreachable goal*, regardless of how large the bonus $U_t(g)$ may be for it — and unlike a value-based filter, this holds deterministically rather than on a high-probability event. This is the formal counterpart of Theorem 3 in the learning setting. By (ii), no reachable goal is permanently excluded once a path to it has been observed. Note, however, that (ii) only *preserves* what has been discovered; it does not guarantee discovery, which is the first gap discussed below.

Step 3: The selected goal has near-maximal uncertainty.

Lemma 3 (High-uncertainty selection)**.** *On $\mathcal{E}_1$, at any step t at which some $g \in \mathcal{G}_L \cap \mathcal{G}_t^{\mathrm{reach}}$ satisfies $U_t(g) > \varepsilon$, the selected goal satisfies*

$$U_t(g_t) \;\geq\; \tfrac{1}{2} \max_{g \in \mathcal{G}_L \cap \mathcal{G}_t^{\mathrm{reach}}} U_t(g). \tag{30}$$

Proof. Let $g_t^\star \triangleq \arg\max_{g \in \mathcal{G}_L \cap \mathcal{G}_t^{\mathrm{reach}}} U_t(g)$, so that $U_t(g_t^\star) > \varepsilon$ by hypothesis. Both g_t and $g_t^\star$ lie in $\mathcal{G}_t^{\mathrm{reach}}$, so by optimality of g_t in Eq. (27),

$$\hat{V}_t(s_0, g_t) + \beta\, U_t(g_t) \;\geq\; \hat{V}_t(s_0, g_t^\star) + \beta\, U_t(g_t^\star). \tag{31}$$

Bounding $\hat{V}_t(s_0, g_t) \leq 1$ and $\hat{V}_t(s_0, g_t^\star) \geq 0$ and rearranging,

$$U_t(g_t) \;\geq\; U_t(g_t^\star) - \frac{1}{\beta} \;=\; U_t(g_t^\star) - \frac{\varepsilon}{2} \;\geq\; U_t(g_t^\star) - \frac{U_t(g_t^\star)}{2} \;=\; \frac{U_t(g_t^\star)}{2}, \tag{32}$$

where the second inequality uses $\varepsilon < U_t(g_t^\star)$. □

Remark 3 (The additive form requires a scale coefficient). With $\beta = 1$ the same argument yields only $U_t(g_t) \geq U_t(g_t^\star) - 1$, which is vacuous because $U_t \leq 1$ by construction — and vacuous precisely in the regime of interest, where all uncertainties have already fallen below the range of $\hat{V}$. The additive rule therefore tracks uncertainty only once its bonus is scaled to dominate that range, with β tied to the target accuracy ε. This is the tabular counterpart of the scaling sensitivity discussed in Section 3.1: an additive combination of reachability and novelty carries a free coefficient that must be set correctly for the rule to work at all, whereas the multiplicative form of Eq. (2) carries none.

Step 4: Pigeonhole on goal samples. This step bounds the number of episodes needed to drive the uncertainty of goals *already in the filter* below ε; whether every $g \in \mathcal{G}_L$ enters the filter is addressed separately below. For $U_T(g) \leq \varepsilon$ it suffices that $n_T(g) \geq \log(SAT/\delta)/\varepsilon^2$. Consider any episode started at a step t at which some goal in $\mathcal{G}_L \cap \mathcal{G}_t^{\mathrm{reach}}$ still has $U_t(g) > \varepsilon$. By Lemma 3, $U_t(g_t) > \varepsilon/2$, and inverting Eq. (25),

$$n_t(g_t) \;<\; \frac{4 \log(SAT/\delta)}{\varepsilon^2}. \tag{33}$$

Every such episode therefore increments the count of a goal whose count is still strictly below $4\log(SAT/\delta)/\varepsilon^2$. Since at most S distinct goals can ever be selected, at most

$$N_{\mathrm{ep}} \;=\; \frac{4\, S \log(SAT/\delta)}{\varepsilon^2} \tag{34}$$

such episodes can occur before every $g \in \mathcal{G}_L \cap \mathcal{G}_T^{\mathrm{reach}}$ satisfies $U_T(g) \leq \varepsilon$. At L steps per episode, this phase costs $\tilde{\mathcal{O}}(SL/\varepsilon^2)$ exploration steps.

What remains. Two components are needed for a complete result, and neither follows from the steps above.

Discovery. The filter of Eq. (26) is sound but not complete. Nothing in the algorithm guarantees that every $g \in \mathcal{G}_L$ eventually enters $\mathcal{G}_t^{\mathrm{reach}}$, and Lemma 2(ii) only preserves goals already discovered. A passive filter admits initialization failures. Consider a two-state deterministic MDP with two actions, one moving from s_0 to a distinct goal g and one staying at s_0, with the identity of each unknown. Before any edge is observed, $\mathcal{G}_0^{\mathrm{reach}} = \{s_0\}$; selecting s_0 triggers first-hit termination at time zero, so no new edge is ever observed and the filter never grows. The two MDPs obtained by swapping the actions remain indistinguishable. A complete argument therefore requires an explicit frontier-expansion mechanism with its own discovery guarantee, as in Lim and Auer (2012) and Tarbouriech et al. (2020); a finite warm-up phase would equally require one.

Policy optimality. Lemma 1 bounds the deviation of $\hat{V}_t$ from $\bar{V}_t$, the average value of the *executed* policies. Conjecture 1 instead concerns the returned policies $\hat{\pi}$ relative to $V^\star$. Closing this requires a stopping rule, a specification of which policy is returned for each goal, and an argument bounding its suboptimality — none of which is supplied by invoking a finite-horizon regret analysis such as Azar et al. (2017), since that analysis addresses a single fixed objective rather than the all-goal scheduling problem SUN-UCB poses.

K.5 Discussion

Comparison with AdaGoal. Conjecture 1 targets the same rate as AdaGoal-UCBVI (Tarbouriech et al., 2022), but the two address different objectives: AdaGoal's guarantee concerns expected hitting-time accuracy over an incrementally identified reachable set, whereas ours concerns discounted first-hit value error. A reduction between the two would be needed before either rate or its associated lower bound could be inherited. The two methods do filter unreachable goals through related mechanisms: AdaGoal imposes an explicit constraint on the estimated hitting time $\mathcal{D}_k(g) \leq L$, while SUN-UCB restricts selection to goals reachable along observed transitions (Lemma 2). Both filters are sound — neither can select a goal outside the true reachable set — and both grow monotonically as data accumulates. The difference is that AdaGoal pairs its filter with an expansion procedure that provably grows the reachable set, which is precisely the component SUN-UCB lacks.

Role of unreachability rejection. Theorem 3 (proved in the oracle setting in Section A.4) is used *algorithmically* in step 2 of SUN-UCB, and is the load-bearing property in Lemma 2. Without a filter, unreachable goals would be selected repeatedly, since the bonus $U_t(g)$ is largest exactly where $n_t(g) = 0$, and the pigeonhole of Step 4 would range over the whole of $\mathcal{S}$ rather than over the reachable set. What this analysis adds to the oracle statement is that the filter must be built from observed *transitions* rather than from value estimates, since the latter cannot distinguish an unreachable goal from a reachable one that has not yet been targeted.

K.6 Additive vs. Multiplicative SUN

The analysis above concerns the additive score $V + \beta\, U_t(g)$, which differs from the multiplicative form $V \cdot 1/n_g$ used in our experiments (Section 4). Remark 3 makes the difference concrete: the additive rule tracks uncertainty only once β is scaled to the target accuracy, since with $\beta = 1$ the novelty term is dominated by the range of V. In the tabular setting this is a mild requirement, as ε is given and $V \in [0, 1]$ is known exactly. In deep RL neither holds: V^θ is approximate, its effective range varies across environments and over training, and there is no target accuracy from which to derive β. The coefficient must therefore be tuned — which is precisely the failure mode we observe for DISCOVER in Section 4.3. SUN's multiplicative form carries no such coefficient: the two signals share a common "zero" (an unreachable or already-saturated goal scores zero on either factor and is rejected regardless of the other) and a common scale (both lie in $[0, 1]$).

We already validated this empirically in Section 4.2 (Figure 1) and Appendix G. Note that for the sake of simplicity, we used the UCB1-style form $\sqrt{\log N_{\text{tot}}/n_g}$, with $N_{\text{tot}} = \Sigma_g n_g$, rather than the scaled bonus $\beta\, U_t(g)$ of Eq. (27).[12]

[12] Both forms decrease in n_g and grow logarithmically in the total count, and $\Sigma_g n_g = T$ after T exploration steps, so they differ in the placement of n_g and in constants rather than in behavior. The UCB1-style coefficient does not, however, reproduce the ε-dependent scaling $\beta = 2/\varepsilon$ that Lemma 3 requires, so the additive variant we evaluate is the unscaled one — the regime Remark 3 predicts should fail. Its poor entropy in Figure 1 is consistent with that prediction.