LEARNING IN A WORLD THAT REMEMBERS: WHEN PAST PERFORMANCE SHAPES FUTURE REWARD

Anonymous authors
Paper under double-blind review

ABSTRACT

An agent can improve its general competence by directing itself towards what it can learn but has not yet learned. Yet an agent may continue to receive reward for repeating previously learned behaviour, without any incentive to broaden its competence. In this paper, we introduce the *ArcadeWorld* problem, where an agent chooses both its actions within individual environments and which environments in a suite to engage with. The agent's goal is to develop general competence by improving across the entire suite within a single lifetime. We propose *leaderboard rewards* to formalize this goal: the agent's past performance is remembered, so that improvement is rewarded while previously attained performance receives diminishing rewards. To understand the challenges of learning with leaderboard rewards, we conduct experiments on a simple abstraction of ArcadeWorld where continual improvement is possible. We find that deep reinforcement learning algorithms can fail to match the performance of simple memory-bounded agents, with exploration being a key challenge. To investigate these challenges at a larger scale, we introduce *AtariWorld*, where an agent can play and switch among Atari games. Standard deep reinforcement learning algorithms similarly struggle in AtariWorld and are even outperformed by exploration methods that receive no external reward based on score.

1 INTRODUCTION

In continual learning problems, the exact same experience never repeats and exploration never ends. Consider a kid in an arcade. She plays Ms. Pac-Man for the first time ever and gets on the leaderboard. What should she do next? Try to beat her own high score, or try another game. The arcade always has a new game to play and a higher score to beat. Nothing in how we build and evaluate agents in reinforcement learning (RL) today resembles this arcade experience.

Beyond the fun metaphor, the arcade reflects many of the widely stated goals of continual RL research. Continual RL is the setting in which: 1) the best agents never stop learning (Abel et al., 2023); 2) the environment is non-ergodic, so some changes are irreversible (Elelimy et al., 2025); and 3) the world is bigger than the agent, so the agent cannot represent it or explore it in full (Javed & Sutton, 2024; Lewandowski et al., 2025). The arcade has all three. A score that was rewarded yesterday may not be rewarded today, because rewards depend on the leaderboard. What counts as good behavior keeps changing, and the best agents must keep learning. The leaderboard is part of the state and only ratchets upward, so the world never returns to an earlier state even though individual games can be replayed. Finally, with a fixed budget of experience and only one game played at a time, the arcade is bigger than current state-of-the-art deep RL agents.[1]

To the best of our knowledge, no existing benchmark captures all three properties. Jelly Bean World (Platanios et al., 2020) and Forager (Tang et al., 2026) are diagnostic gridworld domains. AgarCL (Mohamed et al., 2026) is non-episodic and large, but its reward function and opponents are fixed: its non-stationarity arises from the agent's own growth, so a fixed optimal policy still exists, in principle. XLand (Stooke et al., 2021) pushes task diversity to billions of procedurally generated games, but each task is still an episode with a fixed reward, sampled and reset, so the goal is to

[1]Prior work trained a single network on all 57 ALE games, but with actors playing every game in parallel, a value head per game using the game's identity, and roughly 10^{10} frames of experience (Hessel et al., 2019).

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.

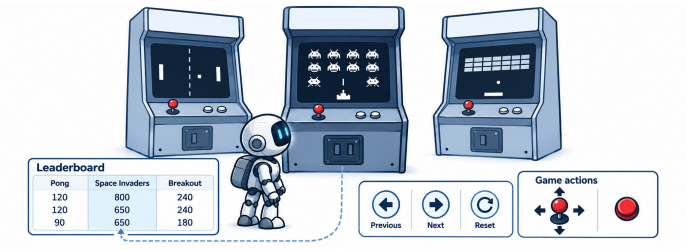

Figure 1: ArcadeWorld lets an agent play, reset, and switch among environments within a single continuing lifetime, while per-environment leaderboards track its scores and determine its rewards.

generalize to new tasks, not to keep improving in one persistent world. NetHack (Küttler et al., 2020), Craftax (Matthews et al., 2024), MineRL (Guss et al., 2019) and MineDojo (Fan et al., 2022) are vast open worlds, and Crafter (Hafner, 2021) is smaller, but all are used episodically with fixed reward functions. Resets undo any irreversible changes the agent makes, and an agent that masters the fixed objective can stop learning.

We contribute ArcadeWorld, a modification that turns any suite of episodic environments into one continuing world for continual learning. The agent lives a single unbroken lifetime, learns how to play each game and which games are worth playing, and is rewarded for improvement. A leaderboard of the agent's own past scores defines the reward. The leaderboard is part of the world's state, so the world is non-ergodic. It remembers what the agent has done and changes in response. We build two such worlds. AtariWorld applies ArcadeWorld to the Arcade Learning Environment. The agent plays the 57 Atari games and switches among them at will. AtariWorld is a challenge problem in the sense that the Arcade Learning Environment was one (Bellemare et al., 2013; Machado et al., 2018), compute-intensive and beyond the algorithms of the day. BitWorld is its partner. It has one game, with the simplest dynamics that keep the leaderboard's effect: the agent advances along an infinite bit-string by guessing each bit, and a wrong guess returns it to the start. The leaderboard records how far the agent has gone, and the reward changes with that progress. An experiment in BitWorld takes minutes, and what is achievable can be computed exactly, so BitWorld supports the rapid iteration and rigorous analysis that a challenge problem alone cannot (Patterson et al., 2024). In both, experience never repeats and exploration never ends.

We formally define leaderboard rewards and show that they have properties desirable for continual reinforcement learning. We define a controlled bit-string process and show how far agents with perfect memory, bounded memory, or no memory can advance in it. We run experiments in both BitWorld and AtariWorld. In BitWorld the agents perform below a simple bounded-memory rule-based agent. In AtariWorld, they struggle to improve even within a single game. These results suggest a need for new algorithms to confront big worlds.

2 ARCADEWORLD

ArcadeWorld gives an agent access to a suite of environments within a single continuing lifetime. The agent interacts with one environment at a time and controls its movement through the suite using three navigation actions: reset the current environment, switch to the next environment, or switch to the previous environment. We refer to an episode within an individual environment as a *period of play* within the agent's continuing lifetime. Unlike an episode, a period of play is not counted as an episode and is not treated as a termination. Instead, the agent continues to interact with the environments as an uninterrupted stream. Apart from these navigation actions and the reward described below, the interaction between the agent and its environment is unchanged.

Within ArcadeWorld, the agent's goal is to develop general competence by learning across the suite. For clarity, we use *points* to refer to an environment's original rewards and *score* to measure the agent's performance (*e.g.*, the discounted or undiscounted sum of points over a period of play). The points provided by the environment are not compatible with this goal because agents can favour specializing in the single environment that provides the most points. We thus seek a reward that encourages learning across the suite by continuing to reward improvement in scores while diminishing the reward for previously attained score levels.

Specifically, we replace each environment's original reward with a *leaderboard reward*. Following the arcade analogy, we maintain a leaderboard for each environment. After every transition, the agent's resulting score is added to the leaderboard, even if it has been recorded before. Unlike conventional game leaderboards, we record intermediate as well as final scores to track performance throughout each period of play.[2] The placement of the agent's current score is one plus the number of recorded scores greater than or equal to it. For a given environment in the suite, the leaderboard reward depends on that environment's leaderboard and the scores before and after the transition. These rewards then encourage the agent to improve further within the current environment or to explore another environment where improvement is possible.

2.1 From episodic environments to arcades

Concretely, we consider a suite of environments, $\{\mathcal{E}_i\}_{i=1}^{N}$, that share an observation space $\mathcal{O}$ and an action space $\mathcal{A}$. The initial environment index i is sampled uniformly from $\{1, \ldots, N\}$. The agent begins in $\mathcal{E}_i$ and interacts with the current environment through its original observation and action spaces. ArcadeWorld also provides the agent with three additional actions for navigating the suite: $\mathcal{A}_{\texttt{nav}} = \{\texttt{next}, \texttt{previous}, \texttt{reset}\}$. The `reset` action reinitializes the current environment, whereas the `next` and `previous` actions move cyclically through the suite, wrapping between environments N and 1. Each navigation action ends the current period of play.

For each transition within the current environment, the agent receives a leaderboard reward determined by the scores before and after the transition and that environment's leaderboard. After evaluating each transition's reward, the resulting score is added to that environment's leaderboard. Let z_t and z_{t+1} denote the scores before and after a transition, and let L_t denote the leaderboard of scores previously recorded for the agent in the current environment at time t, including duplicates. The placement of the resulting score is denoted by $P_{L_t}(z_{t+1})$. Each environment's leaderboard is initialized with a single entry z_0, the initial score for a period of play. Write $H(L_t)$ for the highest recorded score. Under this initialization and recording rule, the current score is always represented in the leaderboard.

Definition 2.1 (Leaderboard reward)**.** A *leaderboard reward* is a nonnegative function $r(z_t, L_t, z_{t+1})$ of the current score, the leaderboard, and the resulting score satisfying the following properties for configurations permitted by the initialization and recording rule:

1. **Score monotonicity.** For a fixed current score z_t and leaderboard L_t, if two possible resulting scores satisfy $y \geq y'$, then $r(z_t, L_t, y) \geq r(z_t, L_t, y')$.
2. **Vanishing reward for poor placement.** For every $\varepsilon > 0$, there is a placement threshold K, independent of the current score, resulting score, and leaderboard, such that $r(z_t, L_t, z_{t+1}) \leq \varepsilon$ whenever $P_{L_t}(z_{t+1}) \geq K$.
3. **Persistent reward for improvement.** For every $\delta > 0$, there is a constant $c_\delta > 0$, independent of the current score and leaderboard, such that
$$r(z_t, L_t, H(L_t) + \delta) \geq c_\delta$$
for every current score z_t and leaderboard L_t for which $H(L_t) + \delta$ is a possible resulting score.

Leaderboard rewards provide relative feedback on performance. For example, top-score improvement, $\max\{0, z_{t+1} - H(L_t)\}$, rewards the magnitude of improvement beyond the recorded high score. This reward is positive only for a new top score and satisfies persistent reward for improvement with $c_\delta = \delta$.

Rewards in ArcadeWorld depend on the agent's interaction history, through the leaderboard, which introduces three challenges. First, the leaderboard can grow without bound and, because the leaderboard is needed to calculate the reward, it cannot be fully observed in the general case. Second, the reward is nonstationary even when the underlying environment is stationary: the same behaviour early in learning can yield a higher reward than the same behaviour later. Third, the underlying process is non-ergodic: once a leaderboard configuration is left, it can never be revisited. These three challenges are all driven by the agent's internal dynamics, rather than the external dynamics common in other continual reinforcement learning environments.

[2] Appendix C discusses when this recording convention can be relaxed.

2.2 STATE, OBSERVATIONS, AND NON-ERGODICITY

The state of ArcadeWorld, or *world state*, is the tuple $s^{\text{AW}} = (i, s, \mathbf{L}, z)$ where i is the index corresponding to the current environment, s is the state of that environment, $\mathbf{L} = (L^{(1)}, \ldots, L^{(N)})$ is the collection of leaderboards for all N environments in the suite, and z is the current score.

An agent in ArcadeWorld only receives the observations from the current environment. Here, we consider observations that provide a sufficient representation of the environment state, and include the current score z. Thus, the observations provide the state information needed to select actions that maximize the score within each environment. The environments' dynamics governing the observations are otherwise unchanged.

Although the state provides sufficient information to select score-maximizing actions, the learning signal can be non-Markov and appear non-stationary relative to the agent's observations. This is because the leaderboard reward depends on the agent's interaction history: the same observed transition can yield different rewards at different points in the agent's lifetime.

Providing the leaderboard alongside the underlying environment observation would make the reward Markov and stationary, but would not remove all learning challenges. First, exposing the full leaderboard would introduce an observation whose size grows throughout the agent's lifetime. Second, even when the leaderboard admits a bounded summary, such as the top score for top-score improvement reward, this summary is subject to irreversible changes. Once the top score increases, the agent can never return to a state with a lower recorded top score. Observing the leaderboard therefore does not eliminate the learning challenges associated with this non-ergodicity. We have the agent receive each environment's original observations, allowing us to investigate how leaderboard rewards affect learning without changing the observation space.

3 CHARACTERIZING LEADERBOARD REWARDS

An agent's goal in ArcadeWorld is to improve its top-scores. Here we prove that leaderboard rewards can incentivize an agent to pursue this goal. We first show a necessary and sufficient condition that characterizes a leaderboard reward which orders agents in terms of their top score.

Proposition 3.1 (Characterization of score-ordering rewards)**.** *Let $r(z_t, L_t, z_{t+1})$ be a leaderboard reward satisfying Definition 2.1. Fix a common initial score z_0 and leaderboard L_0 with $H(L_0) = z_0$. Across all finite score sequences starting from this configuration, cumulative undiscounted reward and final top score induce the same ordering, including ties, if and only if*

$$r(z_t, L_t, z_{t+1}) = F(\max\{H(L_t), z_{t+1}\}) - F(H(L_t)) \tag{1}$$

for some strictly increasing function F. In this case, cumulative reward after T transitions is

$$F(H(L_T)) - F(z_0), \tag{2}$$

where L_T is the leaderboard at time T.

The proof is given in Appendix D.1. Taking $F(z) = z$ therefore gives the top-score improvement reward discussed in Section 2.

While the top-score reward ensures that agents with better scores receive more reward, it also introduces two assumptions that may limit its effectiveness as a learning signal. First, it provides no reward for repeating a score or for getting close to the current highest score: the reward is sparse, and thus provides no incentive for the agent to maintain its competence in the game after achieving its best score. Second, it assumes that the scores across games are fungible: an agent is thus incentivized to play games that provide higher scores, even if that score is not comparable across environments in the suite. While the reward is able to distinguish an agent's ability to achieve high scores, other forms of leaderboard reward can still be consistent with pursuing top-scores.

The next proposition shows that, under certain conditions, any discounted leaderboard reward favours a top-score improvement, when it is possible, over permanently foregoing it.

Proposition 3.2 (Incentive for top-score improvement)**.** *Let $r(z_t, L_t, z_{t+1})$ be a bounded leaderboard reward satisfying Definition 2.1 and fix $0 < \gamma < 1$. Suppose possible scores are bounded above and distinct scores are separated by at least $\delta > 0$.*

Starting from a leaderboard with high score h and a current score at most h, suppose there exists a possible greater high score $h' > h$. Then every stochastic score sequence that almost surely never

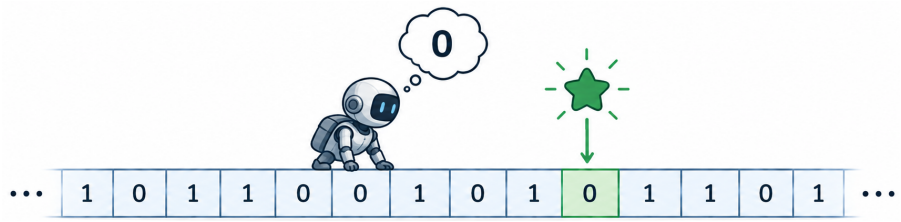

Figure 2: In BitWorld, the agent guesses the bit at its position; a correct guess advances it, and a wrong one returns it to the start. Its previous best position (top-score) is marked by the star.

exceeds h has strictly smaller expected discounted cumulative reward than some score sequence that exceeds h almost surely.
The proof is given in Appendix D.2.

Leaderboard rewards favour relative improvements in the agent's current score, including those improvements below the current top score. They may favour such intermediate improvements over immediately setting a new record. Under the stated assumptions, however, maximizing expected discounted leaderboard return cannot favour foregoing a new record forever when a higher score is possible. This motivates using leaderboard rewards as a learning signal while evaluating agents by their achieved top scores.

3.1 Reciprocal-placement as a count over score thresholds

In addition to favouring relative improvements in the agent's current score, leaderboard rewards also encourage a form of exploration. Specifically, placement can be expressed through counts over lower score thresholds, where a recorded score z' counts as one visit to every threshold $z \leq z'$. Thus, a higher score also contributes to the visit counts of lower score thresholds.

Fix an environment whose leaderboard is initialized with z_0 and retains every recorded score, including duplicates. Let z_m be the resulting score at its mth transition, so m counts transitions within this environment. For each score threshold z, let $N_m(z)$ count transitions whose resulting score meets or exceeds z. These counts are initialized and updated by

$$N_0(z) = 0, \qquad N_{m+1}(z) = N_m(z) + \mathbf{1}\{z_{m+1} \geq z\}. \tag{3}$$

Write $N_m^+(z) = N_m(z) + \mathbf{1}\{z_0 \geq z\}$ for the count including the initial leaderboard entry. Write $P_m(z)$ for the placement of a candidate score z against the initial entry and the m resulting scores recorded so far, with existing entries preceding the candidate in a tie.

Proposition 3.3 (Reciprocal-placement reward as a count-based bonus)**.** *Define reciprocal-placement reward by $r_m(z) = 1/P_m(z)$. This reward is an inverse-count exploration bonus over score thresholds, with an offset accounting for the initial leaderboard entry:*

$$r_m(z) = \frac{1}{N_m(z) + 1 + \mathbf{1}\{z_0 \geq z\}}. \tag{4}$$

The proof is given in Appendix D.3. Although its counts measure score-threshold attainments, reciprocal-placement reward shares the inverse-count form of the Bayesian Exploration Bonus (Kolter & Ng, 2009). In contrast, MBIE-EB and density-model pseudo-count bonuses use inverse-square-root scaling (Strehl & Littman, 2008; Bellemare et al., 2016).

4 BitWorld: A Simple Setting for Continual Improvement

To better understand the challenges of learning in ArcadeWorld, we consider the *BitWorld* abstraction that captures two essential properties. Like ArcadeWorld, BitWorld both permits continued improvement and requires it for sustained success. To capture the possibility of continued improvement, we consider a countably infinite state space that the agent must advance through (illustrated in Figure 2). To capture the demand for improvement to sustain success, we have the agent learn from leaderboard rewards.

In BitWorld, an agent lives in an infinite bit-string. The agent observes a *position* p_t on the string and at every step takes an action corresponding to a bit, $a_t \in \{0, 1\}$. If the action bit agrees with the bit-string at position p_t, then the agent advances to position $p_t + 1$; otherwise, the period of play ends and the agent returns to position 0 without being given a termination flag. These dynamics define the following controlled Markov process.

Definition 4.1 (BitWorld)**.** Fix a bit-string $b = (b_0, b_1, \ldots) \in \{0, 1\}^{\mathbb{N}_0}$, where $\mathbb{N}_0 = \{0, 1, \ldots\}$. The state is the agent's position $p_t \in \mathbb{N}_0$; the action space is $\mathcal{A} = \{0, 1\}$, and the initial position is $p_0 = 0$. The deterministic transition rule is

$$p_{t+1} = \begin{cases} p_t + 1, & a_t = b_{p_t}, \\ 0, & a_t \neq b_{p_t}. \end{cases} \tag{5}$$

The agent's score is given by its position.

To reach position n from the initial position, the agent must produce the action sequence $(b_0, \ldots, b_{n-1})$. When the bit-string is initially unknown and has no predictable structure, the agent must learn these bits through interaction and retain them to reliably reproduce the corresponding action sequence. The process therefore offers an agent an opportunity for unbounded improvement in the positions it can reach while challenging the agent to retain what it has learned.

Interpreting leaderboards as environment-maintained counts. Because the score in BitWorld is its position, leaderboard placements encode state-visitation counts. Thus, through the leaderboard, the environment maintains state-visitation counts which may be communicated to the agent with a placement-based reward. This can be seen as externalized memory, as the state-space is countably infinite and so an agent with bounded memory cannot maintain exact counts for arbitrarily many positions.

Proposition 4.2 (Recovering visit counts from leaderboard placements)**.** *In BitWorld, consider a correct transition to position $i > 0$, and let $n_t(i)$ count previous visits to that position. Initialize the leaderboard with score zero. For every action sequence, the following identities hold, with L_t evaluated before recording the current arrival:*

1. *If scores are recorded after every transition, then*

$$n_t(i) = P_{L_t}(i) - P_{L_t}(i + 1).$$

2. *If scores are recorded at the end of play before returning to zero, then*

$$n_t(i) = P_{L_t}(i) - 1, \qquad \frac{1}{n_t(i) + 1} = \frac{1}{P_{L_t}(i)}.$$

The proof is given in Appendix D.4. In particular, on correct transitions, reciprocal-placement reward with end-of-period recording is both a leaderboard reward and a count-based exploration reward.

Thus, BitWorld provides a simple setting for studying the learning challenges created by leaderboard rewards. For the position score in BitWorld, record-breaking reward and top-score reward coincide because each new record exceeds the previous high score by exactly one. We thus investigate the challenges faced by an agent when it learns from the sparse top-score reward and the decaying reciprocal-placement reward.

4.1 Learning with Leaderboard Rewards in BitWorld

We refer to the top-score and reciprocal-placement settings as BitWorld-Sparse and BitWorld-Decaying, respectively. The leaderboard reward is the extrinsic reward in these experiments.

We can use rule-based agents as baselines to compare learning algorithms against in BitWorld. The *perfect-recall* agent stores every bit that it experiences. If it encounters a new bit, it chooses an action by flipping a fair coin. Its high score grows as $2\sqrt{T}$. Suppose the agent knows the first n bits and stands at position n, its high score. It guesses the bit. With probability $1/2$ the guess is right, and the high score is $n + 1$ after one step. With probability $1/2$ the guess is wrong; the miss reveals

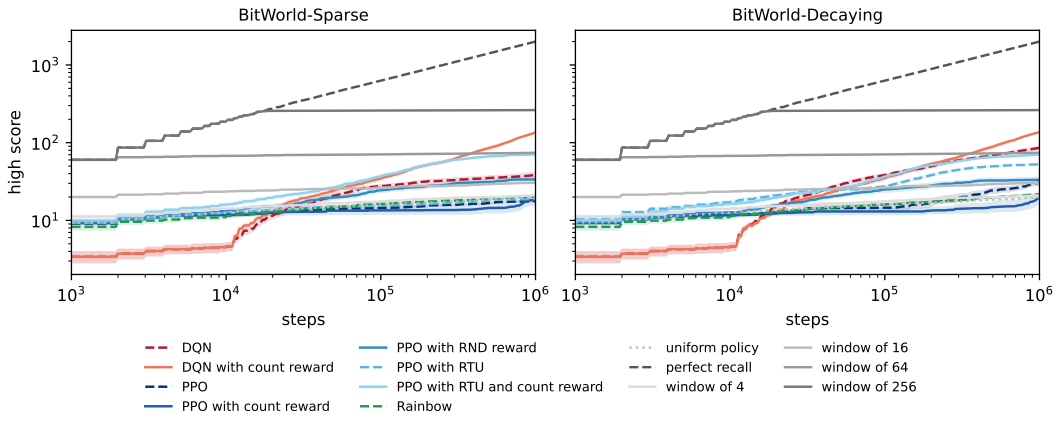

Figure 3: BitWorld-Sparse (left) and BitWorld-Decaying (right), $T = 10^6$, thirty seeds. Error bars are 95% percentile-bootstrap intervals. Dashed lines are agents learning from extrinsic reward. Solid lines from intrinsic reward.

the bit, the period of play ends, and the agent replays the $n + 1$ known bits in $n + 1$ steps to reach $n + 1$. Either way the agent then knows $n + 1$ bits and stands at $n + 1$, so the expected number of steps to raise the high score by one is $1 + \frac{1}{2}(n + 1) = (n + 3)/2$. Summing, the expected number of steps to reach N is $T(N) = \sum_{n<N}(n + 3)/2 = N(N + 5)/4$, so the high score after T steps is about $2\sqrt{T}$ and rises at about $2/(n + 3)$ positions per step.

The *uniform* agent guesses a fair coin at every step, whatever its history. Its high score grows as $\log_2 T$. A period of play is a run of correct guesses ended with a miss, so it reaches score s with probability 2^{-s}, and the number of periods until one reaches s is geometric with mean 2^s. A period of play lasts two steps on average, so reaching s takes about 2^{s+1} steps, and the high score after T steps is about $\log_2 T$.

The *sliding-window* agents are bounded-memory versions of the perfect recall agent. They can remember at most b bits and positions and guess at bits that are not in memory. For comparison, an agent that retains the first b bits and guesses uniformly thereafter advances beyond that prefix through consecutive correct guesses, suggesting the high score would grow approximately as $b + \log_2 T$. We ran sliding-window agents with $b \in \{4, 16, 64, 256\}$; they end near 22, 32, 78, and 270 at $T = 10^6$, each b plus a dozen or so.

We ran four reinforcement learning algorithms on both BitWorld-Sparse and BitWorld-Decaying. Three are non-recurrent, DQN, PPO, and Rainbow, using CleanRL's implementations and hyperparameters (Huang et al., 2022). The modifications for the continual-learning setting are described in Appendix B. For instance, no hyperparameter is annealed over the lifetime, since a continual learner has no end to anneal toward. The fourth agent is recurrent, PPO with a recurrent trace unit (RTU) (Elelimy et al., 2024). In these experiments, the agents observe both their position and the high score, encoded using 20 bits each. In both experiments the algorithms end within a small factor of the uniform agent and far below a window of 256 (the dashed lines in Figure 3).

We then compared against algorithms that learn from their own intrinsic reward, an inverse-square-root state-visitation count reward for DQN and PPO and random network distillation (RND) for PPO (Burda et al., 2019). These agents learn only from the intrinsic reward and never see the extrinsic one, yet under BitWorld-Sparse each outperforms its extrinsic counterpart, and under BitWorld-Decaying none falls far behind it.

In a second experiment, we tested whether learning difficulties persist when the correct action is directly available in the observation. We added an additional feature to the observation that is the correct bit for the agent to select. All the agent needs to learn in this problem is the identity function outputting the last feature. The observation about intrinsic and extrinsic rewards mostly holds from the previous experiment with a few exceptions. In the BitWorld-Sparse setting, the results of the PPO runs were bimodal. Twenty-four seeds ended with high scores between 16 and 54, around uniform random performance. There were, however, 6 runs that ended breaking out and obtaining

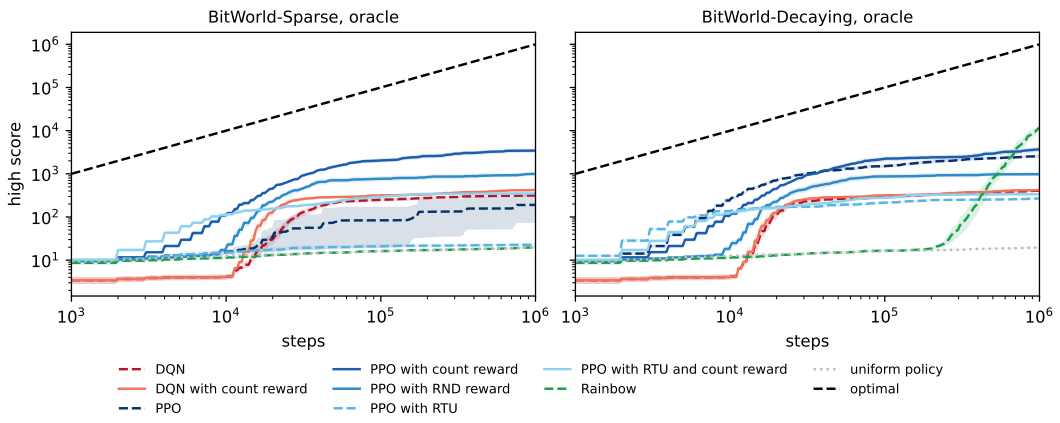

Figure 4: BitWorld-Sparse (left) and BitWorld-Decaying (right) with the oracle observation, $T = 10^6$, thirty seeds. The correct bit is in the observation, so the ideal policy copies it and its high score is the step count. Error bars are 95% percentile-bootstrap intervals. Dashed lines are agents learning from extrinsic reward. Solid lines from intrinsic reward.

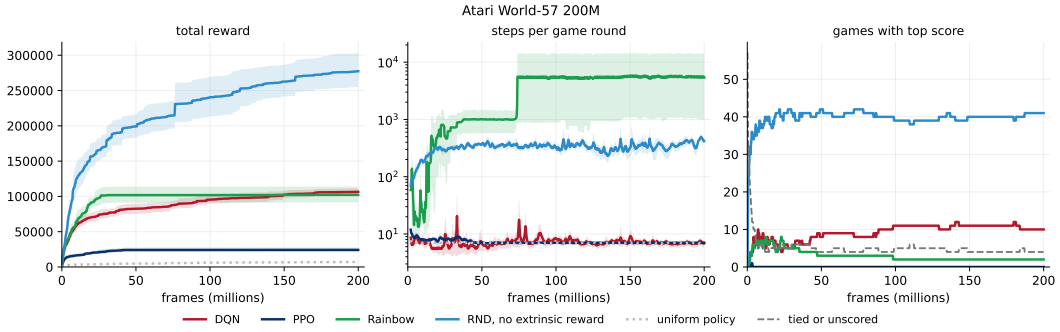

Figure 5: AtariWorld, 57 games, top-score reward, 200 million frames. Left: cumulative reward. Middle: steps per period of play (log axis). Right: the number of games in which an agent's high score is alone at the top; the dashed grey line counts games tied or not yet scored. Lines are means over seeds; shaded regions are 95% percentile-bootstrap confidence intervals.

high scores between 379 and 1429. In BitWorld-Decaying, Rainbow was able to learn the identity policy from extrinsic rewards, which was surprising but then, on the other hand, the correct action was in the observation.

5 AtariWorld

AtariWorld is an ArcadeWorld built from the 57 games of the Arcade Learning Environment (Bellemare et al., 2013). Each game keeps a leaderboard of the agent's scores, and the reward is the top-score improvement reward of Section 3. The agent's lifetime cumulative reward in a game is thus the rise in its high score. The agent sees the frames of the Arcade Learning Environment and nothing of the leaderboard. Appendix A gives additional details.

We ran DQN (Mnih et al., 2015), PPO (Schulman et al., 2017), and Rainbow (Hessel et al., 2018) on the leaderboard reward, and RND (Burda et al., 2019) on the novelty of its observations alone. Each algorithm was run for a lifetime of 200 million frames (roughly 38 days playing 24/7 at 60 FPS), a considerable amount of time for any agent, human or artificial.

RND improves in more games (Figure 5, right) compared to the other agents despite having no access to the extrinsic reward. Despite having higher reward and game scores than the other algorithms, this is not an impressive result for RND. Table 1 reports the five games where RND scores

Table 1: RND's five best games. Each seed's final policy and the uniform policy were evaluated for 30 periods of play in each game. The five games are those where RND is highest by human-normalised score. The Rainbow column is the published single-game result (Hessel et al., 2018, Table 6). Each entry is the mean over seeds, its 95% percentile-bootstrap interval in brackets, and the human-normalised score in parentheses.

Game	RND	Uniform policy	Rainbow (published)
Atlantis	20,144 [18,080, 22,095] (0.45)	21,143 [20,467, 21,918] (0.51)	826,660 (50.30)
Krull	2,051 [1,797, 2,271] (0.42)	1,777 [1,686, 1,857] (0.17)	8,742 (6.69)
Assault	319 [308, 329] (0.19)	306 [300, 311] (0.16)	14,198 (26.90)
Solaris	2,446 [2,009, 3,100] (0.11)	2,217 [2,082, 2,350] (0.09)	3,560 (0.21)
Robotank	3 [2, 4] (0.10)	2 [2, 3] (0.02)	61 (6.10)

highest and compares its final policy with the uniform policy and with the results published in the Rainbow paper (Hessel et al., 2018) when trained on that game alone in the standard episodic setting. We took RND's final policy after 200M frames and evaluated it independently on each game with the switch-game actions removed so the agent is forced to keep playing each of the five games. As we can see, RND's performance is effectively the same as that of the random policy.

Where are these agents spending their time? Mining one game for new top scores or hopelessly skipping around? Sadly, the latter. Figure 6 shows the fraction of the lifetime each agent spent on each game. DQN and PPO divide their lifetime evenly across the 57 games—spending hardly any time in each. Rainbow spends almost all its time in a handful of games, but as we saw in Figure 5, Rainbow is not accruing significant total reward, or achieving top scores. Unfortunately, the deep RL algorithms known to master ALE games one at a time do not master them in one world.

6 Conclusion

ArcadeWorld turns any suite of episodic environments into a single continuing world in which the agent's own history shapes its future reward. A leaderboard of the agent's past scores becomes part of the world's state, so the world is non-ergodic, its reward is nonstationary even when every game is stationary, and the best agent never finishes learning. These properties are the stated goals of continual reinforcement learning, and ArcadeWorld delivers them without designing a new environment: it is a construction that applies to suites the community already uses. Central to this construction are leaderboards that reward performance improvement while diminishing the reward for previously attained performance. Leaderboard rewards order agents by top score exactly when they take a simple difference form, they never favour abandoning a possible record, and placement is precisely a visitation count over score thresholds maintained by the world rather than the agent.

Our experiments show that today's deep RL algorithms are not built for such a world. In BitWorld, where what is achievable can be computed exactly, DQN, PPO, Rainbow and a recurrent PPO agent all end within a small factor of a uniform random policy, far below a rule-based agent that remembers the last 256 steps. Giving them the correct bit in the observation, reducing the problem to learning the identity function, does not reliably fix this, so the failure is not one of network capacity but potentially one of exploration. In AtariWorld, given 200 million frames, roughly 38 days of continuous play, the same algorithms that master Atari games one at a time fail to master them as one world: DQN and PPO spread their lifetime thinly across all 57 games, Rainbow settles on a handful without setting new records, and even RND, which raises its high score in the most games, ends with policies no better than random. These results point to a gap in the algorithms, not the benchmark. Methods that assume stationary targets, revisitable states and a lifetime that ends have little to hold on to when the world remembers.

ArcadeWorld opens several directions. Agents must decide not only how to act but where to spend their experience, which invites methods that pursue learning progress and choose their own curricula. The leaderboard is a form of external memory, which raises the question of what an agent should remember itself and what it can leave to the world. BitWorld and AtariWorld are complementary: BitWorld supports rapid iteration against exact reference agents, while AtariWorld, like the Arcade Learning Environment before it, is a challenge problem beyond the algorithms of the day. We hope both help the community build agents that keep improving for as long as they live.

AI USE STATEMENT

We have not used generative AI tools in generating: data, proofs or other theoretical results, algorithmic ideas, or experiments to run.

We used generative AI to help generate code, but not run our experiments. We also used it to verify the correctness of proofs.

We used generative AI to help find related work and basic writing support (grammar checking and rewording suggestions). Gen AI wrote no parts of the paper directly.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work focuses on foundational aspects of reinforcement learning and the main contributions are methodological and empirical on simulated worlds.

REPRODUCIBILITY STATEMENT

Theory & Formal Conditions: Complete proofs for all formal statements (including Proposition 3.1 on score-ordering rewards, Proposition 3.2 on top-score improvement incentives, and Proposition 3.3 / Proposition 4.2 on leaderboard count equivalences) are provided in Appendix D. All underlying environment assumptions and mathematical definitions for ArcadeWorld and BitWorld are detailed in Section 2 and Section 4.

Algorithmic Baseline Specifications: Analytical expressions and detailed derivations for all rule-based baselines—including the uniform random policy, the perfect-recall agent ($2\sqrt{T}$ progression), and bounded sliding-window agents—are provided in Section 4.1.

Experimental Setup & Hyperparameters: Hyperparameter choices, neural network architectures, optimization settings, and lifetime adaptation rules (such as unannealed schedules and non-clipped rewards) for DQN, PPO, Rainbow, PPO-RTU, and RND are fully specified in Appendix B.

Implementations & Code: All learning agents are CleanRL implementations at their published hyperparameters, with the few departures listed in Appendix B; each result uses Bitworld experiments were over thirty seeds and AtariWorld over 10 seeds. Code will be released upon publication.

REFERENCES

David Abel, André Barreto, Benjamin Van Roy, Doina Precup, Hado van Hasselt, and Satinder Singh. A definition of continual reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning (ICML)*, 2020.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research (JAIR)*, 2013.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. In *Neural Information Processing Systems*, 2016.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations (ICLR)*, 2019.

Esraa Elelimy, Adam White, Martha White, and Michael Bowling. Real-time recurrent learning using trace units in reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Esraa Elelimy, David Szepesvari, Martha White, and Michael Bowling. Rethinking the foundations for continual reinforcement learning. *Reinforcement Learning Journal (RLJ)*, 2025.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35: 18343–18362, 2022.

William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*, 2019.

Danijar Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado Van Hasselt. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3796–3803, 2019.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G. M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research (JMLR)*, 2022.

Khurram Javed and Richard S Sutton. The big world hypothesis and its ramifications for artificial intelligence. In *Finding the Frame: An RLC Workshop for Examining Conceptual Frameworks*, 2024.

J. Zico Kolter and Andrew Y. Ng. Near-bayesian exploration in polynomial time. In *International Conference on Machine Learning*, 2009.

Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. *Advances in Neural Information Processing Systems*, 33:7671–7684, 2020.

Alex Lewandowski, Aditya A. Ramesh, Edan Meyer, Dale Schuurmans, and Marlos C. Machado. The world is bigger! a computationally-embedded perspective on the big world hypothesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research (JAIR)*, 2018.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. *arXiv preprint arXiv:2402.16801*, 2024.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

Mohamed Ayman Mohamed, Kateryna Nekhomiazh, Vedant Vyas, Marcos Menon Jose, Andrew Patterson, and Marlos C. Machado. The cell must go on: Agar.io for continual reinforcement learning. *Reinforcement Learning Journal*, 7, 2026.

Andrew Patterson, Samuel Neumann, Martha White, and Adam White. Empirical design in reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 2024.

Emmanouil Antonios Platanios, Abulhair Saparov, and Tom Mitchell. Jelly bean world: A testbed for never-ending learning. *arXiv preprint arXiv:2002.06306*, 2020.

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

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

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

Steven Tang, Xinze Xiong, Anna Hakhverdyan, Andrew Patterson, Jacob Adkins, Jiamin He, Esraa Elelimy, Parham Mohammad Panahi, Martha White, and Adam White. Forager: a lightweight testbed for continual learning with partial observability in rl. *Reinforcement Learning Journal*, 7, 2026.

A AtariWorld details

The games are arranged alphabetically in a ring. `next` and `previous` move the agent to the neighbouring games, and `reset` restarts the current one. Every game takes the same 18 joystick actions, so the agent has 21 actions in all. The agent observes the standard frames of the Arcade Learning Environment, four stacked 84×84 grayscale frames, with a frame skip of four and sticky actions with probability 0.25 (Machado et al., 2018). A period of play lasts from the start of a game to its end. No termination or truncation signal marks the end; the agent sees a transition to the next period's first frame.

A lifetime is 200 million frames. Each algorithm runs at CleanRL's hyperparameters (Appendix B), but with no schedule over the lifetime and no reward clipping. Each algorithm was run for ten seeds. Confidence intervals are 95% percentile bootstrap over seeds. In Figure 5, an agent holds a game's top score when its high score, the mean over its seeds, exceeds every other learning agent's, and the dashed line counts the games tied or not yet scored.

For Table 1, each seed's final policy was evaluated for 30 periods of play in each game, and the uniform policy likewise. The human-normalised score is 0 at the published random score and 1 at the human tester, both from Badia et al. (2020, Appendix H.4). The Rainbow column is the published single-game result after 200 million frames (Hessel et al., 2018, Table 6).

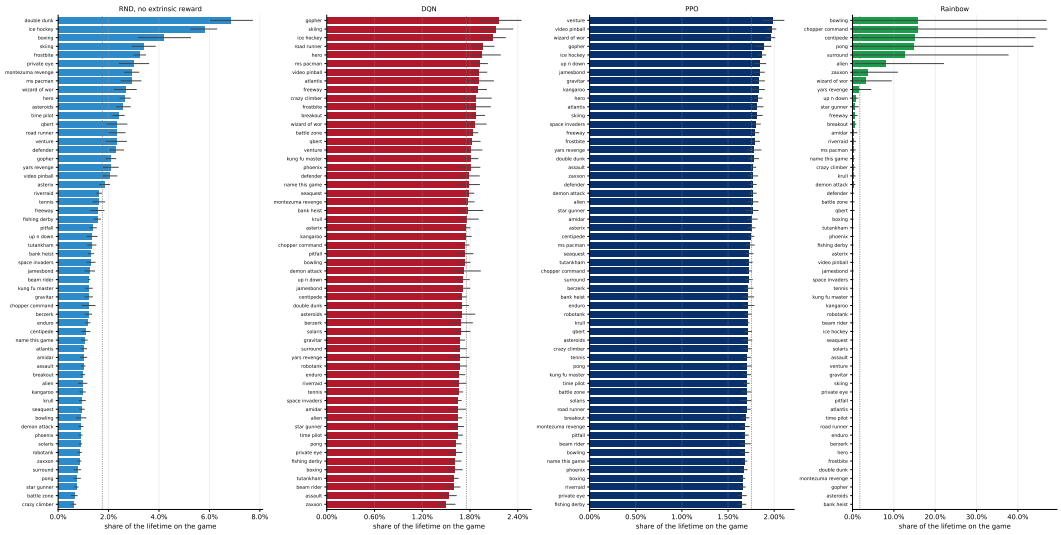

Figure 6: The fraction of the lifetime each agent spent on each of the 57 games under the top-score reward, games ordered by share. Bars are means over the seeds.

B HYPERPARAMETERS

Every learning agent is CleanRL's implementation at CleanRL's published hyperparameters for its setting (Huang et al., 2022). DQN follows `dqn` in BitWorld and `dqn_atari` in AtariWorld, PPO follows `ppo` and `ppo_atari_envpool_xla_jax`, Rainbow follows `rainbow_atari`, and RND follows `ppo_rnd_envpool`. We departed from CleanRL where the continuing setting demanded it and in a few values.

No schedule reads the horizon, since a continuing agent has no end to anneal toward. DQN's exploration rate is a constant, 0.05 in BitWorld and 0.01 in AtariWorld, the values CleanRL anneals to. PPO's and RND's learning rates stay at CleanRL's initial values. Rainbow's importance-sampling exponent β is a constant 0.5, where CleanRL anneals it from 0.4 to 1. No reward is clipped. No episode ends. Value targets and advantage estimates run through the end of a period of play, replay stores the stream once, and the recurrent agent's state is never reset.

A few values differ from CleanRL's. Every agent runs one environment, so a PPO batch is one 128-step rollout. PPO's entropy coefficient is 0.05. A lifetime is 10^6 steps in BitWorld and 5×10^7 steps, 200 million frames, in AtariWorld. The agents that learn from intrinsic reward never see the leaderboard reward; RND's intrinsic coefficient is 1 and its extrinsic coefficient 0.

PPO with RTU uses a linear recurrent trace unit of width 64.

C ALTERNATIVE RECORDING CONVENTIONS

The recording convention, logging scores at every transition, ensures that even unchanged performance accumulates leaderboard entries. These placements are not an issue for top-score improvement reward, because the reward is non-zero only when the score changes. However, for placement-based rewards, duplicates from unchanging scores affect subsequent placements. Thus, omitting those duplicates can allow an unchanged score to earn a positive reward indefinitely, so the vanishing-reward argument in Proposition 3.2 no longer applies in general. Recording only the final score at the end of play requires further care as an intermediate score can earn reward, but subsequent

decreases in score can be pursued before the end of play, allowing the same intermediate score to be rewarding again. When scores are nondecreasing within each period of play, recording the final score can be used, and does preserve top-score improvement reward. Thus, alternative recording conventions can preserve particular rewards or results, but do not generally preserve the entire class of leaderboard rewards and its guarantees.

D PROOFS

D.1 CHARACTERIZATION OF SCORE-ORDERING REWARDS

Proposition 3.1 (Characterization of score-ordering rewards)**.** *Let $r(z_t, L_t, z_{t+1})$ be a leaderboard reward satisfying Definition 2.1. Fix a common initial score z_0 and leaderboard L_0 with $H(L_0) = z_0$. Across all finite score sequences starting from this configuration, cumulative undiscounted reward and final top score induce the same ordering, including ties, if and only if*

$$r(z_t, L_t, z_{t+1}) = F(\max\{H(L_t), z_{t+1}\}) - F(H(L_t)) \tag{1}$$

for some strictly increasing function F. In this case, cumulative reward after T transitions is

$$F(H(L_T)) - F(z_0), \tag{2}$$

where L_T is the leaderboard at time T.

Proof. Since every resulting score is recorded, $H(L_{t+1}) = \max\{H(L_t), z_{t+1}\}$. For necessity, agreement on ties makes cumulative reward a function F of the final top score alone. Agreement on strict comparisons makes F strictly increasing. The reward for a transition is the difference between the cumulative rewards of a sequence and its one-step extension. Their top scores are $H(L_t)$ and $\max\{H(L_t), z_{t+1}\}$, respectively, giving equation 1.

For sufficiency, summing the reward differences telescopes to equation 2. Since F is strictly increasing and $F(z_0)$ is common to all sequences, cumulative reward and final top score induce the same ordering, including ties.

Ordinary top-score reward corresponds to $F(z) = z$.

$$\max\{H(L_t), z_{t+1}\} - H(L_t) = \max\{0, z_{t+1} - H(L_t)\}.$$

□

D.2 INCENTIVE FOR TOP-SCORE IMPROVEMENT

Proposition 3.2 (Incentive for top-score improvement)**.** *Let $r(z_t, L_t, z_{t+1})$ be a bounded leaderboard reward satisfying Definition 2.1 and fix $0 < \gamma < 1$. Suppose possible scores are bounded above and distinct scores are separated by at least $\delta > 0$.*

Starting from a leaderboard with high score h and a current score at most h, suppose there exists a possible greater high score $h' > h$. Then every stochastic score sequence that almost surely never exceeds h has strictly smaller expected discounted cumulative reward than some score sequence that exceeds h almost surely.

Proof. Choose a possible score $h' > h$. Consider any stochastic score sequence that almost surely never exceeds h, and denote $r_t = r(z_t, L_t, z_{t+1})$.

We first show that the leaderboard reward tends to zero, as the placements tend to infinity. Since every resulting score is recorded, the leaderboard grows by one entry at each transition. Fix K. Once the leaderboard contains K entries, each of its top K scores is nondecreasing, bounded above, and increases by at least δ whenever it changes. These scores therefore eventually stop changing. Every new score with placement at most K changes at least one of them, so only finitely many such placements occur. By the second property of leaderboard rewards (vanishing reward for poor placement), we have that $r_t \to 0$ almost surely.

Because rewards are bounded, $\mathbb{E}[r_t] \to 0$. Thus the expected discounted return at time t satisfies

$$V_t := \mathbb{E}\left[\sum_{j=0}^{\infty} \gamma^j r_{t+j}\right] \leq \frac{\sup_{s \geq t} \mathbb{E}[r_s]}{1-\gamma} \longrightarrow 0.$$

With the third property, persistent reward for improvement, we have a constant $c_{h'-h} > 0$ such that attaining h' earns at least $c_{h'-h}$. This is because $H(L_t) = h$ throughout the original sequence. We can thus choose t sufficiently large that $V_t < c_{h'-h}$.

We can then construct another stochastic score sequence with the higher score. We do so by replacing z_{t+1} with h' and leaving all other scores unchanged. In this case, rewards before time t are unchanged. The replacement earns at least $c_{h'-h}$, hence all subsequent rewards remain zero or positive. The increase in expected discounted cumulative reward is therefore at least

$$\gamma^t \big(c_{h'-h} - V_t\big) > 0.$$

The modified sequence exceeds h, proving the claim. □

D.3 Reciprocal placement as a count-based bonus

Proposition 3.3 (Reciprocal-placement reward as a count-based bonus)**.** *Define reciprocal-placement reward by $r_m(z) = 1/P_m(z)$. This reward is an inverse-count exploration bonus over score thresholds, with an offset accounting for the initial leaderboard entry:*

$$r_m(z) = \frac{1}{N_m(z) + 1 + \mathbf{1}\{z_0 \geq z\}}. \tag{4}$$

Proof. Initially, $N_0^+(z) = \mathbf{1}\{z_0 \geq z\}$ and $P_0(z) = N_0^+(z) + 1$. After recording a score z', we can increment $N_m^+(z)$ for each threshold $z \leq z'$. The new entry precedes exactly these candidate scores. This has the effect of increasing their placements by one and leaving all other placements unchanged. Consequently, $P_m(z) = N_m^+(z) + 1$ after any number of recording events. Substituting $N_m^+(z) = N_m(z) + \mathbf{1}\{z_0 \geq z\}$ gives the stated expression and completes the proof. □

D.4 Recovering visit counts from leaderboard placements

Proposition 4.2 (Recovering visit counts from leaderboard placements)**.** *In BitWorld, consider a correct transition to position $i > 0$, and let $n_t(i)$ count previous visits to that position. Initialize the leaderboard with score zero. For every action sequence, the following identities hold, with L_t evaluated before recording the current arrival:*

1. *If scores are recorded after every transition, then*

$$n_t(i) = P_{L_t}(i) - P_{L_t}(i+1).$$

2. *If scores are recorded at the end of play before returning to zero, then*

$$n_t(i) = P_{L_t}(i) - 1, \qquad \frac{1}{n_t(i)+1} = \frac{1}{P_{L_t}(i)}.$$

Proof. For per-transition recording, $P_{L_t}(i) - 1$ counts leaderboard entries at least i, while $P_{L_t}(i+1) - 1$ counts entries at least $i+1$. Since positions are integers, their difference counts entries equal to i. Every arrival at i contributes exactly one such entry, and the initial entry at zero contributes none because $i > 0$. Consequently, $P_{L_t}(i) - P_{L_t}(i+1) = n_t(i)$.

For end-of-period recording: we have that, for each completed period of play, position i is visited exactly once if and only if its final position was at least i. Moreover, the current period of play has not previously visited i. This is because each correct action advances the agent by one position. Thus, exactly $n_t(i)$ final positions are recorded and are at least i. The initial entry at zero again contributes none. It follows that $P_{L_t}(i) = n_t(i) + 1$, giving both identities in the second case and completing the proof. □