EXPLORATION BY MAXIMIZING THE ENTROPY OF RANDOM-NETWORK RESIDUALS

Anonymous authors
Paper under double-blind review

ABSTRACT

State-entropy maximization is a standard objective for reward-free exploration: it rewards states that are rare among those the current policy visits, so collecting diverse data is the objective itself. But the objective equates explored with visited. A rarely visited state stays rewarding after the agent's models have fit it, and a frequently visited one loses its reward while they still fail there. We propose to maximize the entropy of prediction residuals instead of states. We use the residual vector of random network distillation (RND), the difference between the outputs of a fixed random network and of a predictor trained to match it on the agent's experience, and apply to it the nearest-neighbour entropy estimator of state-entropy methods. The residuals of states the predictor has fit are small and lie close together, so these states earn little reward. The new objective still rewards diversity, now measured in what the predictor has yet to fit. We show that the resulting reward keeps RND's preference for large residuals and adds a preference for residuals that are rare among those the current policy produces. We implement the method in a world-model agent and evaluate it in the reward-free setting on a range of continuous-control tasks. It is competitive with the state of the art and nearly always significantly improves over state-entropy maximization.

1 INTRODUCTION

Without a reward to follow, an exploring agent has to decide for itself where to go. This is the central problem of reward-free reinforcement learning, where agents explore to pretrain for downstream tasks (Laskin et al., 2021) or to collect data for learning a world model (Sekar et al., 2020). A principled approach is *state-entropy maximization* (Hazan et al., 2019; Lee et al., 2020; Liu & Abbeel, 2021; Seo et al., 2021), which asks the agent to spread its visits as evenly as possible over the states it can reach. A state's reward falls the more often the agent visits it, which keeps pushing the agent toward states it has rarely seen. In practice, this reward is estimated from nearest-neighbour distances in a learned or random representation of the state (Liu & Abbeel, 2021; Seo et al., 2021).

Spreading visits evenly, however, ignores what the agent can already predict. Consider an environment with a large open area that is easy to predict and a small region where objects can be pushed around. State entropy aims to split the agent's time between the two in proportion to their size, so it keeps rewarding visits to the open area long after there is nothing left to learn there. Epistemic methods such as Random Network Distillation (RND; Burda et al., 2019b) and ensemble disagreement (Pathak et al., 2019; Sekar et al., 2020) do take predictability into account by rewarding states on which a learned model is wrong or uncertain. But they use this signal as a bonus, which alone does not make the agent spread out. A state earns the same bonus however often the agent returns, until the model catches up, so the agent can keep repeating the few interactions it predicts worst.

We keep entropy maximization and change what the agent is spread over. State-entropy methods such as APT (Liu & Abbeel, 2021) and RE3 (Seo et al., 2021) measure entropy in a representation of the state, learned contrastively in APT and drawn at random in RE3, and this representation determines what counts as uniform. Neither choice takes into account what the agent can predict. We instead measure entropy in the space of the agent's prediction errors, where all states the agent predicts well collapse toward a single point. A well-predicted region then contributes little to the entropy regardless of its size, and a poorly predicted region contributes according to the variety of errors it produces. We call this objective *Random-Network Residual Entropy* (RRE). Maximizing

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.

it spreads the agent over what it has yet to learn; in our example, visits to the open area earn little once the agent predicts it well, and broad coverage of the small region remains rewarding. Since RRE is still an entropy objective, repeated visits to a state lower its reward, even before the agent's predictions improve.

To measure prediction errors, we borrow RND's construction. A predictor network is trained to imitate a fixed, randomly initialized target network on the states the agent visits. The difference between their outputs, a vector we call the *residual*, records what the predictor still gets wrong at a state. Where RND rewards how large this residual is, RRE rewards how rare it is, using the same nearest-neighbour distances as state-entropy methods, now between the residuals of states from recent rollouts of the policy (Figure 1). RRE needs no networks beyond RND's. Its reward is also insensitive to the overall scale of the errors, because rescaling all residuals only shifts all rewards by a constant. RND's bonus, by contrast, shrinks with the errors and is renormalized with running statistics.

Section 3.3 formalizes this picture. Compared with state entropy, RRE replaces the uniform target over states with one that weights each region by the residual space it covers, and it acts mainly by withdrawing reward from regions the predictor has fit. Its reward favours the same states as RND's bonus when large residuals are also rare. It differs in two respects, though: it drops for a region as the agent crowds into it, and it distinguishes between states with equally large errors according to which outputs the predictor gets wrong.

In summary, we propose RRE, an exploration objective that maximizes the entropy of the residuals of a predictor distilling a fixed random network (Section 3.1). It reuses RND's networks, and its reward is insensitive to the overall scale of the errors (Section 3.2). We show that RRE reweights the uniform target of state entropy by the residual space each region covers, and that its reward, unlike RND's bonus, falls with repeated visits and depends on which outputs the predictor gets wrong (Section 3.3). We then compare RRE with state-entropy, prediction-error, and ensemble-based exploration on pixel-based locomotion and manipulation environments built on OGBench (Park et al., 2025), including variants with a noisy TV, where all methods share one DreamerV2 agent (Hafner et al., 2021) and differ only in their intrinsic reward. RRE roughly doubles the aggregate coverage reached by APT and by RND, never ends below Plan2Explore (Sekar et al., 2020), and is competitive with CIG (Joseph et al., 2026) while training a single predictor (Section 4.2). Ablations show that these gains depend on measuring novelty among the residuals themselves, rather than weighting state novelty by the size of the error, and on comparing each residual with residuals from many rollouts rather than only with those from its own (Sections 4.3 and 4.4).

2 Preliminaries

2.1 Reward-Free Exploration as State-Entropy Maximization

We consider reward-free reinforcement learning (Sekar et al., 2020; Laskin et al., 2021) in a Markov decision process (MDP) without extrinsic rewards, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \rho_0, \gamma)$, with state space $\mathcal{S} \subseteq \mathbb{R}^n$, action space $\mathcal{A}$, transition kernel $\mathcal{P}(s_{t+1} \mid s_t, a_t)$, initial state distribution ρ_0, and discount factor $\gamma \in (0, 1)$. The agent interacts with $\mathcal{M}$ to populate a replay buffer and optimizes a policy $\pi(a_t \mid s_t)$ against an intrinsic reward r_t to maximize $\mathbb{E}_\pi\big[\sum_{t\geq 0} \gamma^t r_t\big]$. Since there is no extrinsic reward for the agent to optimize, the objective is purely to explore the environment.

A standard exploration objective in this setting is *state-entropy maximization* (Hazan et al., 2019; Lee et al., 2020; Seo et al., 2021). Let $d^\pi = (1-\gamma)\sum_{t\geq 0} \gamma^t d_t^\pi$ denote the discounted state occupancy under π, where d_t^π is the distribution of s_t when $s_0 \sim \rho_0$, and let p^π denote its density on $\mathcal{S}$ whenever one exists. The objective maximizes the differential entropy of the occupancy:

$$\max_\pi \mathcal{H}\big(d^\pi\big) \;=\; \max_\pi \mathbb{E}_{s\sim d^\pi}\big[-\log p^\pi(s)\big]. \tag{1}$$

Because differential entropy decomposes as $\mathcal{H}(d^\pi) = \log \mathrm{vol}(\mathcal{S}_R) - D_{\mathrm{KL}}(d^\pi \,\|\, U_{\mathcal{S}_R})$ whenever the states reachable under some policy form a set $\mathcal{S}_R$ of finite volume (Cover & Thomas, 2006), optimizing equation 1 is equivalent to minimizing the divergence from the uniform distribution $U_{\mathcal{S}_R}$ on $\mathcal{S}_R$. The entropy is therefore bounded by $\log \mathrm{vol}(\mathcal{S}_R)$, with equality if and only if $d^\pi = U_{\mathcal{S}_R}$, and the objective pulls the occupancy toward $U_{\mathcal{S}_R}$, although the dynamics and the initial-state term $(1-\gamma)\rho_0$ of d^π generally prevent any policy from attaining it exactly.

2.2 Nonparametric Estimation of Surprisal

Unlike standard RL objectives that are linear in the state occupancy measure d^π, the differential entropy objective equation 1 is strictly concave. As a result, it cannot be expressed as the expected return of any fixed reward function. Following the Frank–Wolfe framework of Hazan et al. (2019), one can optimize it via sequential linearization: taking a first-order Taylor expansion around the current occupancy d^π yields an intrinsic reward equal to the state surprisal $-\log p^\pi(s_t)$ (up to an additive constant), where p^π is held fixed during the policy update. While theoretical convergence in Hazan et al. (2019) requires maintaining an explicit mixture over iterations, practical methods such as Active Pre-Training (APT; Liu & Abbeel, 2021) train a single policy directly on this evolving reward stream.

Prior work estimates surprisal with k-nearest-neighbour (k-NN) statistics over an evaluation pool of N states drawn from the replay buffer or from short rollouts of the policy, computed on embeddings $\phi(s)$ from a learned contrastive encoder (Liu & Abbeel, 2021) or a fixed random network (Seo et al., 2021).

$$\bar{\rho}_k(s) \;=\; \frac{1}{k}\sum_{j=1}^{k} \big\|s - s^{(j)}\big\|, \qquad r_t \;=\; \log \bar{\rho}_k(s_t). \tag{2}$$

By the analysis underlying the Kozachenko–Leonenko differential entropy estimator (Kozachenko & Leonenko, 1987; Singh et al., 2003), applied to each of the k neighbour distances, the log mean distance satisfies

$$\log \bar{\rho}_k(s) \;\approx\; \tfrac{1}{n}\big(-\log p(s)\big) + c(N, n, k), \tag{3}$$

where p is the density of the states in the pool (which coincides with p^π only when the pool is drawn from d^π), and $c(N, n, k)$ depends only on the pool size, the state dimension, and the neighbour count. The log mean distance is thus a rescaled and shifted surprisal, and since neither a positive rescaling nor a constant shift of the reward changes the optimal policy in the setting of Section 2.1, $\log \bar{\rho}_k(s)$ serves directly as the intrinsic reward.

Relation equation 3 holds only up to a per-state error whose mean vanishes as N grows but whose variance shrinks only with k, so each reward is a noisy estimate of its surprisal, and k trades this noise against the locality of the density estimate.

3 Method: Random-Network Residual Entropy

The state-entropy objective introduced in Section 2.1 provides a principled mechanism for exploration: its concavity in the occupancy distribution lowers the reward wherever the agent concentrates, pulling the occupancy toward the uniform distribution on the reachable states. However, this dispersion is fundamentally *epistemic-blind*. Because the objective operates directly on states (or a representation $\phi(s)$ of them), the uniform distribution it pulls toward weights each reachable region by its volume alone, spending exploration budget equally on states a predictive model already predicts well and on states it still gets wrong.

Epistemic exploration methods address this limitation by conditioning rewards on predictive error, as in Random Network Distillation (RND; Burda et al., 2019b) or ensemble disagreement (Pathak et al., 2019; Sekar et al., 2020). Yet these approaches reduce predictive failure to a scalar bonus optimized via expected return, and this reduction introduces its own trade-off: it casts exploration into a standard RL objective that is linear in the state occupancy, forfeiting the dispersive properties of entropy maximization.

To unite the benefits of both paradigms, we propose *Random-Network Residual Entropy* (RRE). Rather than maximizing the entropy of visited states or the expected value of scalar errors, RRE maximizes the differential entropy of the *residual error vectors* of a predictor distilling a random network. By applying a concave entropy functional directly to the model's error distribution, RRE naturally filters out states the predictor has already mastered while compelling the policy toward residuals that are rare in the current occupancy, in magnitude or in direction.

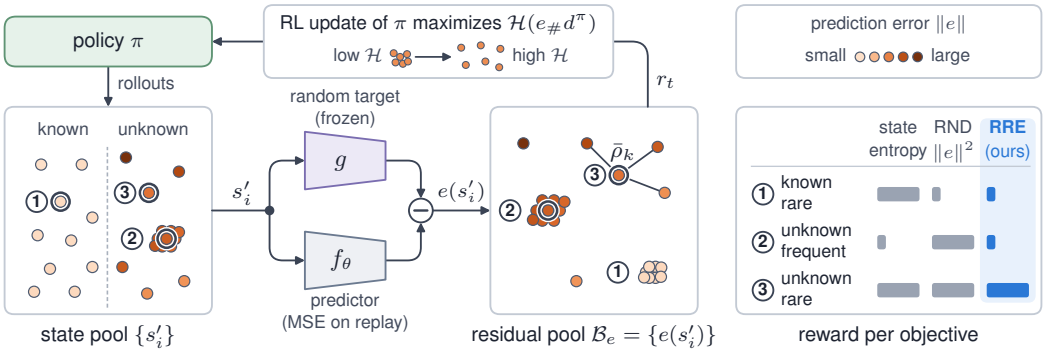


Figure 1: **RRE rewards states that are both unknown to the predictor and rare under the policy.** The residuals $e(s) = f_\theta(s) - g(s)$ of rollout states form the pool $\mathcal{B}_e$, and each transition is rewarded as in equation 6 with the log mean distance $\bar{\rho}_k$ from its residual to its k nearest neighbours in $\mathcal{B}_e$. RL then spreads the residuals out, maximizing the objective in equation 5. States the predictor already fits, such as ①, collapse onto $e \approx 0$, and frequently visited ones such as ② stay crowded, so only ③ earns a high reward; state entropy also rewards ①, and RND also rewards ② (right; schematic).

3.1 Residual Entropy Maximization

We define the exploration objective directly over model prediction errors. Following RND (Burda et al., 2019b), we fix a randomly initialized target network $g\colon \mathcal{S} \to \mathbb{R}^L$ and train an online predictor $f_\theta\colon \mathcal{S} \to \mathbb{R}^L$ on replay states via mean squared error. For a state $s \in \mathcal{S}$, the residual error vector is:

$$e(s) \;=\; f_\theta(s) - g(s) \;\in\; \mathbb{R}^L. \tag{4}$$

Because f_θ is trained throughout exploration, e changes over time; we define the objective below for the current predictor, which makes it non-stationary. The RRE objective maximizes the differential entropy of the pushforward residual distribution $e_{\#}d^{\pi}$:

$$\max_{\pi}\; \mathcal{H}\big(e_{\#}d^{\pi}\big) \;=\; \max_{\pi}\; \mathbb{E}_{s\sim d^{\pi}}\big[-\log p_e^{\pi}(e(s))\big], \tag{5}$$

where p_e^{π} denotes the density of the residual distribution $e_{\#}d^{\pi}$.[1] Because the pushforward $d^{\pi} \mapsto e_{\#}d^{\pi}$ is linear and differential entropy is concave in the distribution, equation 5 is concave in d^{π}, though not strictly if e is not injective. Sequentially linearizing equation 5 via Frank–Wolfe updates, as in Section 2.2, therefore yields the residual surprisal $-\log p_e^{\pi}(e(s_t))$ as the policy's reward.

3.2 Nonparametric Reward Formulation

As in Section 2.2, we estimate the residual surprisal with k-NN statistics, here over the residuals $\mathcal{B}_e = \{e(s'_i)\}_{i=1}^{N}$ of N next states sampled from recent policy rollouts:

$$r_t \;=\; \log \bar{\rho}_k\big(e(s_{t+1})\big), \tag{6}$$

where $\bar{\rho}_k$ is the mean distance to the k nearest neighbours as in equation 2, taken among the other residuals in $\mathcal{B}_e$. By equation 3, r_t estimates the residual surprisal up to a positive scaling and an additive shift, which leave the optimal policy unchanged (Section 2.2). As in RND, the reward for transition (s_t, a_t, s_{t+1}) is evaluated at the next state.

Unlike RND's bonus $\|e\|^2$, whose magnitude decays as f_θ fits g and which RND renormalizes with running statistics, r_t is unaffected by the overall residual scale: scaling every residual by $\lambda > 0$ only shifts every reward by $\log \lambda$. What is rewarded is a residual's position relative to the pool, not its absolute size. To keep the reward finite when a residual coincides with its k nearest neighbours, we add a small constant $\varepsilon > 0$ to $\bar{\rho}_k$ inside the logarithm, which preserves this invariance as long as $\bar{\rho}_k \gg \varepsilon$. Computing the pairwise distances within the pool costs $\mathcal{O}(N^2 L)$ per update, quadratic in the pool size like the k-NN rewards of Section 2.2.

[1] For $L > n$, the residuals lie on an n-dimensional surface in $\mathbb{R}^L$; there, densities and volumes of residuals are taken with respect to n-dimensional surface measure.

3.3 Analysis: Two Views of the Residual Surprisal

We now make precise how RRE keeps the dispersion of state-entropy maximization while directing it by predictive error. We read the residual surprisal $-\log p_e^\pi(e(s))$ from two sides: from state space, which shows what it adds to the state surprisal of Section 2.2, and from residual space, which shows what it adds to the squared error rewarded by RND. Throughout, the predictor f_θ is held fixed, so that e is a fixed map.

View from state space: what RRE adds to state entropy. Picture a small ball of states around s and its image under the residual map $s \mapsto e(s)$. Where the predictor has not fit the target, f_θ generally does not follow g, so the error varies with s about as fast as the networks themselves and the ball maps onto a patch of ordinary size. Where it has fit, f_θ tracks g as s varies, so their difference barely changes and the ball is squeezed into a tiny patch. Write $\sigma_e(s)$ for this stretch factor: the volume of the image divided by the volume of the ball, in the limit of a small ball.[2] An approximate fit makes σ_e small, not zero, so what follows applies to fitted regions too; an exact fit is the limiting case. Stretching spreads a region's probability mass over more residual volume and squeezing concentrates it, so the residual surprisal at s is the state surprisal plus the log-stretch:

$$-\log p_e^\pi\big(e(s)\big) \;=\; -\log p^\pi(s) \;+\; \log \sigma_e(s). \tag{7}$$

The first term is the reward of state-entropy maximization (Section 2.2). The second is a correction set by the predictor: moderate where the predictor has not fit the target, and very low where it has, because there the error barely changes as the state moves. It therefore acts mainly by withdrawing reward from fitted regions rather than by adding reward to novel ones.

Averaging equation 7 over d^π turns this pointwise statement into one about the objective. Let $Z = \int_{\mathcal{S}_R} \sigma_e(s)\,\mathrm{d}s$ be the (n-dimensional) volume of the reachable residuals $e(\mathcal{S}_R)$, and let $U^e_{\mathcal{S}_R}$ be the distribution on $\mathcal{S}_R$ with density σ_e/Z, under which the residuals are uniform on $e(\mathcal{S}_R)$. Then

$$\mathcal{H}\big(e_{\#}d^\pi\big) \;=\; \mathcal{H}\big(d^\pi\big) + \mathbb{E}_{s\sim d^\pi}\big[\log \sigma_e(s)\big] \;=\; \log Z \;-\; D_{\mathrm{KL}}\big(d^\pi \,\|\, U^e_{\mathcal{S}_R}\big). \tag{8}$$

This mirrors the decomposition in Section 2.1 with one change. State entropy pulls the occupancy toward $U_{\mathcal{S}_R}$, which weights each region of state space by its volume in $\mathcal{S}$; RRE pulls it toward $U^e_{\mathcal{S}_R}$, which weights each region by the volume of residual space it covers. A region the predictor has fit covers little residual volume, however large it is as a set of states, and therefore receives little weight; in the limit of an exact fit, where its residuals coincide, it receives none. This is the sense in which RRE filters out what the predictor already knows.

View from residual space: what RRE adds to RND. RND rewards the squared distance from $e(s)$ to the origin, where the residual of a perfectly fitted state lies; RRE rewards the log-distance from $e(s)$ to its nearest neighbours among the current residuals equation 6. RND thus asks how wrong the predictor is at s, RRE how unusual this error is among the errors the policy currently produces. When the current residuals are densest at the origin and thin out with distance from it, equally in every direction, far from the origin means far from other residuals, so RRE's reward, like RND's, grows with the residual's length. This agreement breaks in two ways. First, it holds only for the current occupancy. With the predictor fixed, RND pays a state the same bonus however often the policy visits it, whereas RRE's reward there falls as the pool fills with that region's residuals, although the errors are unchanged. RND lowers such a bonus only indirectly, as training shrinks the errors. RRE thus keeps the dispersion of state-entropy maximization (Section 2.1) that RND's linear objective forfeits. Second, residuals need not spread evenly across directions. RND reduces a residual to its length, so all states where the predictor is equally wrong earn the same bonus; RRE also sees the direction, which records which target outputs the predictor gets wrong and with which sign, and rewards residuals pointing where few other current residuals point.

In summary, relative to state entropy, RRE reweights states by the residual volume they cover; relative to RND, it scores a residual by how crowded its neighbourhood is rather than by its distance from the origin.

[2] Formally, $\sigma_e = \sqrt{\det(J_e^\top J_e)}$ with $J_e = J_{f_\theta} - J_g \in \mathbb{R}^{L\times n}$. Equations equation 7 and equation 8 assume $L \geq n$, a full-rank J_e, and e injective on $\mathcal{S}_R$; without injectivity, equation 7 and the first equality in equation 8 become upper bounds).

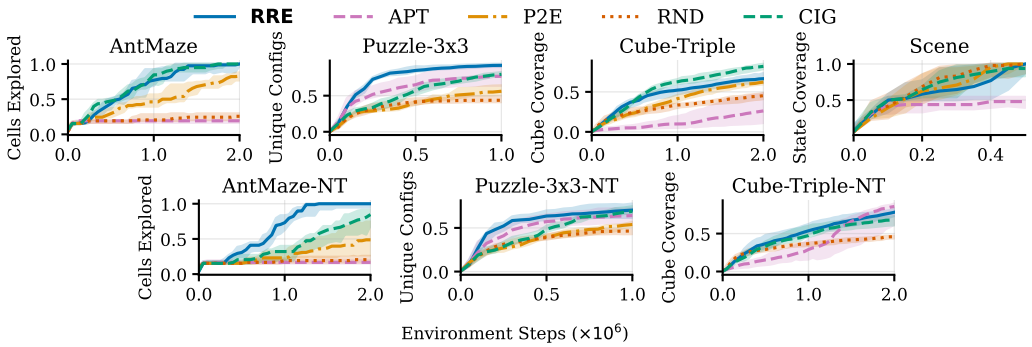


Figure 2: **RRE keeps extending its coverage where APT and RND stall.** Coverage of the task-relevant state over training on the four OGBench environments (top) and the three noisy-TV variants (bottom); 1 means fully explored. IQM over five seeds with 95% bootstrap CIs.

4 Experiments

Section 3.3 argues that RRE keeps the dispersion of state-entropy maximization while directing it by prediction error. Our experiments test whether this translates into more effective exploration, and which of RRE's ingredients the improvement depends on. Specifically, we ask:

RQ1 **Exploration.** Does directing state-entropy maximization by prediction error let an agent explore faster and more broadly than maximizing state entropy or prediction error alone, and how does it compare with ensemble-based exploration?

RQ2 **Residual geometry.** Can RRE's gains be recovered by weighting state novelty with the magnitude of the prediction error, or do they depend on the geometry of learned residuals?

RQ3 **Pool composition.** Does RRE rely on estimating the residual entropy of the occupancy, as its objective prescribes, or on residual diversity within individual trajectories?

To isolate the effect of the exploration objective, all methods share the same DreamerV2 world model and actor-critic and differ only in their intrinsic reward. Section 4.1 describes the environments, metrics, and baselines, and Sections 4.2–4.4 answer the questions in turn.

4.1 Experimental Setup

Environments and evaluation. We evaluate on four pixel-based continuous-control environments built on the OGBench simulators (Park et al., 2025). In *AntMaze*, a quadruped must learn to walk before distant cells become reachable; in *Puzzle* 3×3, *Cube-Triple*, and *Scene*, most object configurations require long manipulation sequences within one episode, such as specific Lights-Out-style press sequences, displacing all three cubes, or unlocking a drawer or window before opening it. Noisy-TV variants (Burda et al., 2019a;b) of AntMaze, Puzzle 3×3, and Cube-Triple (suffix -NT) recolor part of the scene whenever an action dimension exceeds a threshold, offering visual novelty that by itself adds nothing to coverage. We measure coverage, the fraction of discretized task-relevant configurations reached so far, from ground-truth simulator states, independently of the latent space of the intrinsic rewards. We train for 0.5M–2.0M steps and report interquartile means (IQM) over five seeds with 95% stratified bootstrap confidence intervals (CIs) (Agarwal et al., 2021), per environment and aggregated (Appendix A).

Agent and baselines. All methods share a DreamerV2 agent (Hafner et al., 2021) whose actor-critic is trained on imagined rollouts of $H = 15$ latent steps, so that, as in Plan2Explore (Sekar et al., 2020), intrinsic rewards are computed on imagined model states (Appendix A.4). We compare RRE with *APT* (Liu & Abbeel, 2021), whose k-NN reward (Eq. equation 2) it extends; with *RND* (Burda et al., 2019b), whose target and predictor it shares, so that the two differ only in how the residual becomes a reward; and with two ensemble-disagreement methods, *Plan2Explore* (P2E) and *CIG* (Joseph et al., 2026), which discounts disagreement already explained by earlier steps of the same imagined rollout.

4.2 EXPLORATION

RRE explores faster and more broadly than the two methods it builds on. Neither APT nor RND is reliably better than the other: RND ends ahead on four environments, APT on three (Figure 3). RRE, however, ends above the better of the two on four environments (AntMaze, Puzzle 3×3, Cube-Triple, and AntMaze-NT), with non-overlapping CIs on all but Puzzle 3×3, where they barely overlap, and within overlapping CIs on the other three. Its aggregate final coverage is 0.88, against 0.46 for APT and 0.43 for RND. RRE thus does more than select the better of the two, most plainly on AntMaze, with or without the noisy TV: APT and RND end at 0.26 or less, little beyond the 0.15 that every method covers within 0.05M steps, whereas RRE covers the whole maze. RRE also explores faster (Figure 2): the area under its coverage curve exceeds that of both methods everywhere except on Scene, and on five environments it reaches the better one's final coverage within 15 to 56% of the training budget. After the first 0.25M steps, it falls behind only on Scene, where RND leads mid-training (0.92 vs. 0.69 at 0.35M), and on Cube-Triple-NT, where APT overtakes it after about 1.6M steps and ends 0.08 higher, both times within overlapping CIs.

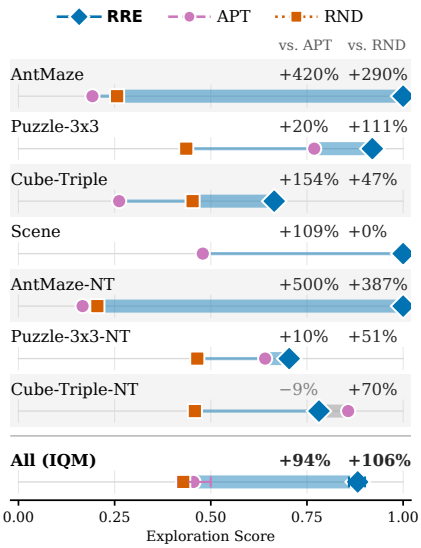


Figure 3: **RRE improves on both methods it builds on.** Final coverage (IQM, five seeds) and RRE's relative gain over APT (state entropy) and RND (prediction error); gray: APT or RND ahead. RND is hidden behind RRE on Scene.

Without an ensemble, RRE keeps pace with the stronger ensemble method. RRE's final coverage is never below that of P2E, and its CI lies entirely above P2E's on five environments (Figure 2). Without a distractor, RRE and the stronger ensemble method, CIG, each lead on one environment with non-overlapping CIs, RRE on Puzzle 3×3 from 0.1M steps on and CIG on Cube-Triple from 0.7M steps on (final coverage 0.82 for CIG vs. 0.66 for RRE); on AntMaze and Scene, their final CIs overlap. The noisy TV lowers CIG's final coverage on all three variants, by 0.11 to 0.15, and P2E's on AntMaze from 0.83 to 0.49. RRE, in contrast, is the only method to cover the whole of AntMaze-NT, by 1.4M steps, and it ends above CIG there and on Cube-Triple-NT (non-overlapping CIs). Yet on Puzzle 3×3-NT, its final coverage drops by 0.22, the largest drop of any method on this variant, to the level of APT and CIG. On Cube-Triple-NT, the distractor even helps the two methods that maximize a k-NN entropy estimate, APT (0.26 to 0.86) and RRE (0.66 to 0.78), while it hurts CIG and leaves P2E and RND unaffected: although the recoloring itself adds no coverage, whether it hinders exploration depends on the objective as much as on the environment.

In short, directing state entropy by prediction error makes exploration reliable where maximizing either alone fails, and competitive with ensemble-based exploration while training a single predictor. RQ2 and RQ3 test what this depends on: the geometry of the residuals and a pool drawn from the occupancy rather than, as in CIG's redundancy term, from the prefix of a single rollout.

4.3 RESIDUAL GEOMETRY

Weighting state novelty by the magnitude of the prediction error does not recover RRE's gains. Equation equation 7 writes RRE's reward as the state surprisal plus a correction set by the predictor. The most direct way to build a reward of this kind from existing components is to weight state novelty by the size of the prediction error, much as NGU (Badia et al., 2020) scales an episodic novelty bonus by an RND-based factor. We therefore construct *APT×RND*, which multiplies APT's reward by the RND bonus. Like RRE, it draws on state novelty and prediction error and so discounts both the fitted and the crowded states of Figure 1; unlike RRE, it combines the two signals outside the entropy and sees only the magnitude of the error. We run this comparison on Puzzle 3×3, whose discrete configuration space makes coverage an exact count. RRE explores both faster and more completely (Figure 4, left): it reaches APT×RND's final coverage in less than half the budget, and from a quarter of the budget on, its confidence interval lies above APT×RND's until the end of

training. As both methods discount the same states, the gap points to the value of measuring novelty among the errors themselves, in their full geometry, rather than among states scaled by the error's magnitude.

4.4 Pool Composition

RRE relies on estimating residual entropy over the occupancy, not within individual trajectories. RRE compares each residual with neighbors drawn from many imagined rollouts. Two ablations on Puzzle 3×3 change which rollouts supply them: *Own-Rollout Pool* admits only the residual's own rollout, turning RRE into a short-horizon episodic bonus (Badia et al., 2020; Henaff et al., 2022), and *Other-Rollouts Pool* admits all rollouts but its own.

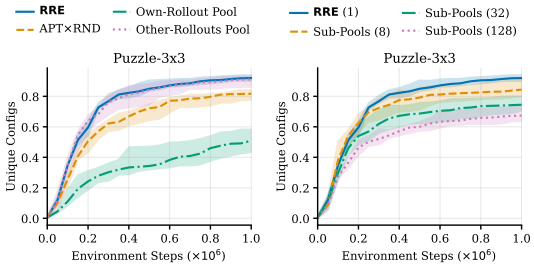


Figure 4: **Removing residual geometry or shrinking the pool slows exploration.** Fraction of the Puzzle 3×3 configurations visited (IQM over five seeds; shaded: 95% bootstrap CIs). Left: APT$\times$RND tests residual geometry (Section 4.3), the pool ablations the choice of neighbors (Section 4.4); Other-Rollouts Pool nearly coincides with RRE. Right: the pool split into 8, 32, or 128 disjoint sub-pools, each residual drawing its neighbors only from its own sub-pool; RRE (1) keeps a single pool (Section 4.4).

Restricting neighbors to a residual's own rollout severely limits exploration: Own-Rollout Pool ends at a coverage RRE reaches within 0.15M steps (Figure 4, left). Excluding it instead changes little: the confidence intervals of Other-Rollouts Pool and RRE overlap throughout training.

The two ablations thus point the same way: what matters is that a residual is compared with residuals from other rollouts, while its own rollout adds little. Because all rollouts come from the same policy, the other rollouts largely pass through the same regions and already supply similar neighbors. With only its own rollout, in contrast, a residual can be crowded only by residuals from that rollout. Each rollout is then rewarded for not repeating itself, but not for avoiding where the other rollouts go, so all rollouts can follow the same route and still collect the full reward. Only a pool shared across rollouts makes them compete for rarely visited regions and thereby spreads the occupancy (Section 3.3).

A graded ablation confirms this. *Sub-Pools (K)* partitions the pool into K disjoint sub-pools and draws each residual's neighbors only from its own sub-pool, which also divides the cost of the pairwise distances by K. Coverage declines steadily as the split becomes finer, yet even the finest split stays well above Own-Rollout Pool (Figure 4, left and right). RRE thus explores best when each residual is compared with as much of the occupancy as possible, as its objective prescribes, and saving computation by splitting the pool costs coverage.

5 Related Work

Maximum state entropy exploration. Data-based methods, in the terminology of Laskin et al. (2021), encourage exploration by maximizing the entropy of the state occupancy (Hazan et al., 2019) or by matching the occupancy to a target distribution (SMM; Lee et al., 2020). In practice, the entropy is usually estimated from k-NN distances between raw states, learned embeddings or random features (Mutti et al., 2020; Liu & Abbeel, 2021; Yarats et al., 2021; Seo et al., 2021). Count-based bonuses (Bellemare et al., 2016; Ostrovski et al., 2017; Tang et al., 2017; Machado et al., 2019) and episodic novelty bonuses (Savinov et al., 2019; Henaff et al., 2022; Jiang et al., 2025) build on the same intuition. All of these methods treat a state as novel regardless of whether the agent can already predict it. We keep the entropy objective and its k-NN estimate but apply them to the residuals $f_\theta(s) - g(s)$ of a predictor trained to match a random network. These residuals collapse wherever the predictor is accurate. Under the assumptions of Section 3.3, equation 8 shows that RRE is a form of SMM whose target distribution is defined implicitly by the predictor and changes as it learns.

Exploration via prediction error and uncertainty. Knowledge-based methods instead reward the agent for reaching states that a learned model predicts poorly or is uncertain about (Schmidhuber, 1991; Oudeyer & Kaplan, 2007). The bonus is typically the prediction error of a forward model (Pathak et al., 2017; Burda et al., 2019a), the information gain or surprise of a probabilistic dynamics model (Houthooft et al., 2016; Achiam & Sastry, 2017; Mazzaglia et al., 2022), the disagreement within an ensemble (Shyam et al., 2019; Pathak et al., 2019; Sekar et al., 2020), or a quantity derived from predictors trained on random targets (Burda et al., 2019b; Ciosek et al., 2020; Ash et al., 2022; Lobel et al., 2023; Yang et al., 2024). For a fixed model, the bonus of a state (or transition) is the same no matter how often the policy visits it. The objective is therefore linear in the occupancy, and visits spread out only as the model is updated, which Lee et al. (2020) interpret as an approximate form of SMM that lacks historical averaging. RRE uses the same residuals as RND but maximizes their entropy. Because this objective is concave in the occupancy, the reward in a region drops as the policy visits it more often, even while the predictor stays fixed. The reward also depends on where a state's residual lies relative to those of other recently visited states, both in direction and in length (Section 3.3).

Combining novelty and prediction error. Closest to our approach are methods that combine the two ideas. They pair a lifelong bonus based on prediction error or model uncertainty with a term that discounts states similar to ones already seen, either within the current episode (Badia et al., 2020; Zhang et al., 2021b; Henaff et al., 2023; Joseph et al., 2026) or according to a density model of recent visits (Zhang et al., 2021a). RRE obtains both effects from a single quantity, the density of the residuals over recent rollouts. Its reward is low both in regions the agent visits often and in regions the predictor already fits, because the residuals are dense in both. Computing it requires no networks beyond the two used by RND.

6 Conclusion

We introduced RRE, which maximizes the entropy of the residuals of a predictor distilling a fixed random network rather than the entropy of states. Its k-NN reward reuses RND's networks and is insensitive to the overall error scale. RRE pulls the occupancy toward a distribution that weights each region by the residual volume it covers, which withdraws reward from regions the predictor has fit; unlike RND's bonus, its reward also falls with repeated visits. On pixel-based OGBench environments, RRE roughly doubles the aggregate coverage of APT and RND and is competitive with ensemble-based exploration while training a single predictor. More broadly, RRE instantiates a general principle: maximizing the entropy of a pushforward $h_{\#}d^{\pi}$ of the occupancy. State entropy corresponds to the identity map, APT and RE3 to a representation of the state, and RRE to the residual map. The choice of map thus implicitly sets which states exploration should cover. RRE chooses a map whose target withdraws weight from whatever the predictor has learned.

Limitations. Our approach inherits some of the susceptibility to noisy TVs from RND. A residual records what the predictor has not yet fit, not whether it is worth learning. A distractor that stays hard to fit therefore keeps covering residual volume, which may partly explain why RRE loses more coverage on Puzzle 3×3-NT. The pairwise distances cost $\mathcal{O}(N^2L)$ per update, and splitting the pool to reduce this cost lowers coverage (Section 4.4). Finally, we evaluate coverage with a single model-based agent and run our ablations only on Puzzle 3×3.

Future work. The pushforward view suggests maps beyond RND residuals, such as the errors of a dynamics model or the deviations of ensemble members from their mean. Maps that discount aleatoric variation (Mavor-Parker et al., 2022) could make the target robust to distractors. Approximate nearest-neighbour search or parametric density estimates could remove the quadratic cost without splitting the pool.

AI use statement

We used generative AI tools to check the correctness of our mathematical claims and proofs. We also used them to find relevant prior work and relate our results to it, to suggest a structure for the paper, and to improve the readability of author-written text. Furthermore, AI has assisted in creating the figures of this paper. We have reviewed all AI-assisted work, and the authors verified every issue

the tools raised and every reference they suggested. 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 is fundamental research in reinforcement learning. All experiments are conducted in simulated environments using publicly available benchmarks. The study involves no human subjects, collects no data from people, and uses no personally identifiable information. We do not release new datasets. As with most advances in reinforcement learning, our methods could eventually be incorporated into downstream systems whose applications carry their own risks. However, our contribution introduces no risks beyond those common to general-purpose machine learning research, and we do not foresee any direct negative societal impact. The authors declare no conflicts of interest. All authors have read and adhere to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

We have taken several steps to ensure the reproducibility of our results. The complete source code needed to reproduce all experiments, including training and evaluation scripts and the configuration files for each reported setting, is provided in anonymized form as supplementary material. Section 3 describes the proposed method in full, and Appendix A.4 lists all hyperparameters and model architectures.

REFERENCES

Joshua Achiam and Shankar Sastry. Surprise-Based Intrinsic Motivation for Deep Reinforcement Learning. *CoRR*, abs/1703.01732, 2017.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 29304–29320. Curran Associates, Inc., 2021.

Jordan T. Ash, Cyril Zhang, Surbhi Goel, Akshay Krishnamurthy, and Sham M. Kakade. Anti-Concentrated Confidence Bonuses For Scalable Exploration. In *International Conference on Learning Representations*, 2022.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andew Bolt, and Charles Blundell. Never Give Up: Learning Directed Exploration Strategies, February 2020.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying Count-Based Exploration and Intrinsic Motivation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. Large-Scale Study of Curiosity-Driven Learning. In *International Conference on Learning Representations*, 2019a.

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

Kamil Ciosek, Vincent Fortuin, Ryota Tomioka, Katja Hofmann, and Richard Turner. Conservative uncertainty estimation by fitting prior networks. In *International Conference on Learning Representations*, 2020.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, July 2006. ISBN 0-471-24195-4.

Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with Discrete World Models. In *International Conference on Learning Representations*, 2021.

Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2681–2691. PMLR, June 2019.

Mikael Henaff, Roberta Raileanu, Minqi Jiang, and Tim Rocktäschel. Exploration via Elliptical Episodic Bonuses. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

Mikael Henaff, Minqi Jiang, and Roberta Raileanu. A Study of Global and Episodic Bonuses for Exploration in Contextual MDPs. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 12972–12999. PMLR, July 2023.

Rein Houthooft, Xi Chen, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. VIME: Variational Information Maximizing Exploration. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Yuhua Jiang, Qihan Liu, Yiqin Yang, Xiaoteng Ma, Dianyu Zhong, Hao Hu, Jun Yang, Bin Liang, Bo XU, Chongjie Zhang, and Qianchuan Zhao. Episodic Novelty Through Temporal Distance. In *The Thirteenth International Conference on Learning Representations*, 2025.

Tim Joseph, Marcus Fechner, Philipp Stegmaier, Karam Daaboul, and J. Marius Zöllner. CIG: Exploration via conditional information gain, 2026.

L. F. Kozachenko and N. N. Leonenko. Sample estimate of the entropy of a random vector. *Problems of Information Transmission*, 23(2):95–101, 1987.

Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. URLB: Unsupervised Reinforcement Learning Benchmark. In *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Lisa Lee, Benjain Eysenbach, Emilio Parisotto, Erix Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient Exploration via State Marginal Matching, 2020.

Hao Liu and Pieter Abbeel. Behavior From the Void: Unsupervised Active Pre-Training. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021.

Sam Lobel, Akhil Bagaria, and George Konidaris. Flipping Coins to Estimate Pseudocounts for Exploration in Reinforcement Learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22594–22613. PMLR, July 2023.

Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. Count-Based Exploration with the Successor Representation, November 2019.

Augustine Mavor-Parker, Kimberly Young, Caswell Barry, and Lewis Griffin. How to Stay Curious while avoiding Noisy TVs using Aleatoric Uncertainty Estimation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15220–15240. PMLR, July 2022.

Pietro Mazzaglia, Ozan Catal, Tim Verbelen, and Bart Dhoedt. Curiosity-Driven Exploration via Latent Bayesian Surprise. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7): 7752–7760, June 2022. doi: 10.1609/aaai.v36i7.20743.

Mirco Mutti, Lorenzo Pratissoli, and Marcello Restelli. A policy gradient method for task-agnostic exploration. *CoRR*, abs/2007.04640, 2020.

Georg Ostrovski, Marc G. Bellemare, Aäron van den Oord, and Rémi Munos. Count-Based Exploration with Neural Density Models. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2721–2730. PMLR, August 2017.

Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? A typology of computational approaches. *Frontiers in Neurorobotics*, Volume 1 - 2007, 2007. ISSN 1662-5218. doi: 10.3389/neuro.12.006.2007.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking Offline Goal-Conditioned RL. In *International Conference on Learning Representations (ICLR)*, 2025.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-Driven Exploration by Self-Supervised Prediction. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 2778–2787, Sydney, NSW, Australia, 2017. JMLR.org.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-Supervised Exploration via Disagreement. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5062–5071. PMLR, June 2019.

Sai Rajeswar, Pietro Mazzaglia, Tim Verbelen, Alexandre Piché, Bart Dhoedt, Aaron Courville, and Alexandre Lacoste. Mastering the Unsupervised Reinforcement Learning Benchmark from Pixels. In *40th International Conference on Machine Learning*, 2023.

Nikolay Savinov, Anton Raichuk, Damien Vincent, Raphael Marinier, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic Curiosity through Reachability. In *International Conference on Learning Representations*, 2019.

Jürgen Schmidhuber. Curious model-building control systems. In *Proceedings of the 1991 IEEE International Joint Conference on Neural Networks (IJCNN '91), Singapore*, volume 2, pp. 1458–1463. IEEE, 1991. doi: 10.1109/IJCNN.1991.170605.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to Explore via Self-Supervised World Models. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8583–8592. PMLR, July 2020.

Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State Entropy Maximization with Random Encoders for Efficient Exploration. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9443–9454. PMLR, July 2021.

Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-Based Active Exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5779–5788. PMLR, June 2019.

Harshinder Singh, V. Hnizdo, Adam Demchuk, and Neeraj Misra. Nearest Neighbor Estimates of Entropy. *American Journal of Mathematical and Management Sciences*, 23, February 2003. doi: 10.1080/01966324.2003.10737616.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Chen Xi, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. #Exploration: A study of count-based exploration for deep reinforcement learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Kai Yang, Jian Tao, Jiafei Lyu, and Xiu Li. Exploration and Anti-Exploration with Distributional Random Network Distillation. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 56397–56421. PMLR, July 2024.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11920–11931. PMLR, July 2021.

Tianjun Zhang, Paria Rashidinejad, Jiantao Jiao, Yuandong Tian, Joseph E. Gonzalez, and Stuart Russell. MADE: Exploration via Maximizing Deviation from Explored Regions. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021a.

Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E. Gonzalez, and Yuandong Tian. NovelD: A Simple yet Effective Exploration Criterion. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021b.

A Environments

We evaluate on four environments built on the visual versions of the OGBench simulators (Park et al., 2025) and on Noisy-TV variants of three of them. We use neither OGBench's datasets nor its goals and task rewards, so no goal is ever shown to the agent and all training is reward-free. Appendix A.1 covers the setup that all environments share and the coverage metric, Appendix A.2 the four base environments, and Appendix A.3 the Noisy-TV variants

A.1 Common Setup and Coverage

The agent acts at every control step of the simulator, without action repeat, with continuous actions normalized to $[-1, 1]$, and it perceives each step as an RGB image of 64×64 pixels. Episodes always last 1000 steps. Coverage should grow only through the agent's own behavior and not through randomness at reset. In the three manipulation environments, we therefore reseed the simulator with the same value at every reset, so every episode starts from a bit-identical scene. In AntMaze, the ant starts every episode in the same cell, up to OGBench's default perturbation of its initial position by at most a quarter of the cell width. Table 1 summarizes the environments.

Table 1: Environments. $\dim \mathcal{A}$ is the action dimension, Steps the reward-free training budget, and $|\mathcal{C}|$ the number of configurations over which coverage is measured. Every Noisy-TV variant (Table 2) shares its base environment's simulator, budget, and coverage metric.

Environment	Robot	$\dim \mathcal{A}$	Steps	Coverage over	$\lvert\mathcal{C}\rvert$
AntMaze	Ant (quadruped)	8	2.0M	maze cells	26
Puzzle 3×3	UR5e + gripper	5	1.0M	button configurations	512
Cube-Triple	UR5e + gripper	5	2.0M	joint cube bins	729
Scene	UR5e + gripper	5	0.5M	lock, drawer, and window states	16

Coverage. For each environment, a map ϕ sends the ground-truth simulator state to one element of a finite set $\mathcal{C}$ of task-relevant configurations. Depending on the environment, ϕ bins a position or reads off discrete button and lock states. After t environment steps, the coverage is

$$\mathrm{Cov}(t) = \frac{\left|\{\phi(s_\tau) : \tau \le t\}\right|}{|\mathcal{C}|},$$

the fraction of configurations reached by any real transition collected so far. The set of reached configurations is shared by all episodes of a run, so coverage can only grow. Each Noisy-TV variant uses the same ϕ and $\mathcal{C}$ as its base environment.

A.2 Base Environments

AntMaze. In `visual-antmaze-medium-navigate-v0`, an eight-joint quadruped (the ant) is controlled by joint torques. Its maze is an 8×8 grid of square cells with a side length of 4 m, of

which 26 are free. The ant starts every episode in a corner of the maze, inside a 2×2 block of free cells that opens onto the remaining maze only through a single cell, and the most distant free cell is 11 cells away along the corridors. The camera follows the ant from behind and above, and, as in OGBench, the floor carries a position-dependent color gradient from which the agent can infer its location. Getting far beyond the initial block requires the agent to discover, from images, a gait that it can sustain along the corridors. ϕ assigns the position of the ant's torso to the cell it lies in, so $\mathcal{C}$ consists of the 26 free cells.

Puzzle 3×3**.** In `visual-puzzle-3x3-v0`, a UR5e arm with a Robotiq 2F-85 gripper is controlled by relative end-effector position (3 dimensions), relative yaw (1), and relative gripper opening (1). In front of the arm lies a 3×3 grid of binary buttons, all initially off. A press registers once the arm pushes a button below a fixed depth, and it flips the pressed button together with the buttons directly above, below, left, and right of it, following the rules of the game Lights Out. ϕ returns the vector of button states, so $\mathcal{C} = \{0,1\}^9$ with 512 configurations. The toggle matrix has full rank over $\mathrm{GF}(2)$, so each configuration corresponds to exactly one set of buttons pressed an odd number of times, and every configuration can be reached. Only 46 configurations have such a set of at most two buttons, and only 256 one of at most four. Coverage above $46/512 \approx 9\%$ therefore requires pressing at least three distinct buttons within one episode, and coverage above 50% at least five.

Cube-Triple. In `visual-cube-triple-play-v0`, the same arm faces three cubes, which start at $(x, y) = (0.36, -0.15)$, $(0.36, 0)$, and $(0.36, 0.15)$ m, in the same order in every episode. ϕ assigns each cube's (x, y) position to a 3×3 grid over $x \in [0.25, 0.60]$ m and $y \in [-0.35, 0.35]$ m, clipping positions outside this region to the nearest border cell, and returns the triple of the three cubes' cells, so $|\mathcal{C}| = 9^3 = 729$. The cubes are distinguished, so exchanging two cubes yields a new configuration, while their heights are ignored, so a stack counts by its footprint. The cubes start in three different cells of the row with the smallest x. Moving a single cube therefore reaches at most 9 configurations, whereas in 512 of the 729 configurations each cube lies outside its initial cell, which requires displacing all three cubes within one episode.

Scene. In `visual-scene-v0`, the same arm faces a cube, a drawer, a sliding window, and two buttons that toggle a lock on the drawer and on the window, respectively. Every episode starts with the cube at $(0.375, 0.055)$ m, the drawer and the window closed, and both locked. ϕ returns the two lock states together with whether the drawer and the window are open, where each counts as open once it has moved more than half of its travel (0.16 m for the drawer, 0.2 m for the window). Thus $\mathcal{C} = \{0,1\}^4$ with 16 configurations; the cube's position does not enter ϕ. Only the four configurations in which both are closed are reachable by pressing buttons alone, which caps coverage at 25% for an agent that never slides the drawer or the window. The other twelve require unlocking the drawer or the window and then opening it, and seven of them additionally require re-locking one of the two while it is open. Full coverage thus includes a configuration in which the agent has unlocked, opened, and re-locked both the drawer and the window within one episode.

A.3 Noisy-TV Variants

AntMaze, Puzzle 3×3, and Cube-Triple each come in a Noisy-TV version (Burda et al., 2019b), marked with the suffix *-NT* in figures. It resamples the color of one part of the scene whenever one action dimension is above 0.9 in the normalized action range $[-1, 1]$, and additionally at every reset. Colors are drawn in HSV space, with hue from $\mathcal{U}[0, 1]$ and saturation and value from $\mathcal{U}[0.7, 1]$, independently of the simulator state. The threshold is checked on every step, so holding the action above it resamples the colors on every step, and the new colors appear from the next observation on. Each such step yields an image unlike any seen before, although the task-relevant state has not changed. Any exploration bonus computed on observations, or on a representation that encodes the colors, therefore stays high at such steps, while coverage, which reads the same ϕ and $\mathcal{C}$ as in the base environment, does not change. Table 2 lists the three variants; Scene has none.

AntMaze Noisy-TV. All maze walls receive one common new color whenever the first torque dimension exceeds the threshold. Only the walls change; in particular, the floor gradient that encodes position is untouched, leaving the agent able to localize itself from the image. The trigger works only while the ant is in a cell (i, j) with $i + j$ even, a checkerboard of 13 of the 26 free cells. The

Table 2: Noisy-TV variants. The listed surface is recolored whenever the trigger dimension exceeds 0.9 while the agent is in the listed region. Apart from these colors, each variant is identical to its base environment in Table 1.

Base environment	Recolored surface	Trigger dimension	Active in
AntMaze	maze walls (one color)	first joint torque	13 of 26 cells
Puzzle 3×3	arm and gripper (9 colors)	gripper	everywhere
Cube-Triple	floor	gripper	everywhere

start cell is one of them, so the distractor can be used from the first step of every episode, before the ant has learned to move. Since the trigger dimension is one of the ant's joint torques, walking through these cells can also recolor the walls as a side effect.

Puzzle 3×3 Noisy-TV. Each of the nine materials of the arm and gripper (four of the UR5e and five of the Robotiq gripper) receives its own new color whenever the gripper dimension exceeds the threshold, and nothing else in the scene is recolored. Pressing requires no grasping, which leaves the gripper dimension free to serve purely as a switch for the distractor.

Cube-Triple Noisy-TV. The floor on which the cubes rest is tinted with a new color whenever the gripper dimension exceeds the threshold; the cubes, the arm, the walls, and the lighting keep their colors. Unlike in Puzzle 3×3, the trigger is entangled with the task, since the agent also needs the gripper dimension to grasp the cubes. The agent therefore cannot simply leave the trigger dimension alone, and has to keep moving cubes while strong gripper commands also flood its observations with unpredictable floor colors.

A.4 Hyperparameters

All methods share one agent: a DreamerV2 world model (Hafner et al., 2021) trained on pixels of the real transitions collected so far, and a Dreamer actor–critic trained entirely on imagined rollouts of that model. As in Plan2Explore (Sekar et al., 2020), intrinsic rewards are therefore computed on imagined latent states, so the state s of Sections 2 and 3 is the model state. World model, actor–critic, network sizes, and optimizers are identical across methods, and no method trains a representation of its own, so the methods differ only in the intrinsic reward the actor–critic maximizes. APT computes the k-NN reward of Eq. equation 2 on latent states as in Rajeswar et al. (2023). RND rewards the squared residual $\|e(s)\|^2$, and RRE uses the same target network g, predictor f_θ, and predictor training. Plan2Explore rewards the disagreement of an ensemble of one-step predictors, and CIG, derived from trajectory-level information gain, discounts this disagreement by the part already explained by earlier steps of the same imagined rollout. Table 3 lists the reward-specific hyperparameters of RRE and of the four baselines, and Table 4 lists the backbone settings shared by all five methods. Every value is read from the resolved configuration of the reported runs, and every method uses the same values on every environment.

Table 3: Intrinsic-reward hyperparameters. RND and RRE use the same predictor and target networks, and CIG uses the P2E ensemble. All reward heads use the MLP and optimizer in the last block.

Hyperparameter	**Symbol**	**Value**
RRE (equation 4, equation 6)		
Predictor, target, and training	f_θ, g	as RND (below)
Neighbours, pool size, distance	k, N	as APT (below)
Log floor	ε	10^{-12}
APT (Liu & Abbeel, 2021)		
Particle	—	deterministic GRU state (600-d)
Neighbours	k	12
Pool size	N	37,500 (15 imagined steps $\times$ 2,500 starts)
Distance	—	Euclidean
RND (Burda et al., 2019b)		
Residual dimensionality	L	512
Predictor / target input	—	next latent state (stochastic + deterministic)
Target network	g	random initialization, frozen
Predictor training	f_θ	MSE to g on every replay batch
Plan2Explore (Sekar et al., 2020)		
Ensemble size	M	5
Target dimensionality	D	1536
CIG (Joseph et al., 2026)		
Ensemble	—	P2E ensemble (above)
Ridge scale multiplier	—	1.0
$\hat{\sigma}^2$ EMA momentum	—	0.99
$\hat{\sigma}^2$ floor	—	10^{-8}
Reward-head MLP and optimizer (RRE, RND, P2E, CIG)		
Hidden layers $\times$ units	—	4×400
Activation / initialization	—	ELU / Glorot
Optimizer	—	Adam, lr 10^{-4}, $\epsilon = 10^{-5}$, wd 10^{-6}, grad-clip 100

Table 4: Backbone hyperparameters shared by RRE and all baselines. Settings not listed follow the DreamerV2 defaults of Hafner et al. (2021).

Hyperparameter	**Symbol**	**Value**
DreamerV2 world model		
Observation	—	64×64 RGB
Stochastic latent (categorical)	—	32×32
Deterministic GRU state	—	600 (layer-normalized GRU)
KL balancing	α	0.8
KL scale	—	0.1
Free nats	—	0.0
Loss scales (image / discount / reward)	—	1.0 / 5.0 / 0.0
CNN channel multiplier	—	48
Model optimizer	—	Adam, lr 2×10^{-4}, $\epsilon = 10^{-5}$, wd 10^{-6}, grad-clip 100
Actor–critic over imagined rollouts		
Imagination horizon	H	15
Discount	γ	0.99
λ-return	λ	0.95
Actor / critic MLP	—	4×400, LayerNorm, ELU
Actor distribution	—	truncated normal on $[-1, 1]$, std $\in [0.1, 2.1]$
Actor gradient	—	dynamics backpropagation
Entropy regularization	—	10^{-4}
Actor learning rate	—	8×10^{-5}
Critic learning rate	—	8×10^{-5}
Slow-critic update	—	hard, every 100 gradient steps
Replay buffer and training schedule		
Replay capacity	—	10^6 transitions
Sequence length	—	50
Batch size (trajectories)	—	50
Random prefill	—	5,000 env steps
Train every	—	100 env steps
Gradient updates per train call	—	10
Action repeat	—	1
Episode length	—	1,000 env steps
Intrinsic-reward normalization (all methods)		
Per-step reward normalization	—	z-score, EMA momentum 0.99
Return-level normalization	—	divide by Welford running std. of λ-returns
Warm-up (batch statistics blended in)	—	1,000 policy updates