Information-Based Exploration via Random Features

Anonymous Author(s)
Affiliation
Address
email

Abstract

Representation learning has enabled classical exploration strategies to be extended to deep Reinforcement Learning (RL), but often makes algorithms more complex and theoretical guarantees harder to establish. We introduce Random Feature Information Gain (RFIG), grounded in Bayesian kernel methods theory, which uses random Fourier features to scalably approximate information gain and compute exploration bonuses in non-countable spaces. We provide error bounds on information gain approximation and avoid the black-box aspects of deep-based uncertainty estimation, for optimism-based exploration. We present practical details that make RFIG scalable to deep RL scenarios, enabling smooth integration with classical deep RL algorithms. Experimental evaluation across control and navigation tasks demonstrates that RFIG achieves competitive performance with well-established deep exploration methods while offering superior theoretical interpretation.

1 Introduction

In Reinforcement Learning (RL), agents learn optimal decision-making strategies through trial-and-error interactions with an environment, receiving rewards or penalties that guide their learning process [Sutton et al., 1998]. A fundamental challenge is the exploration-exploitation tradeoff, where agents must balance between exploiting current knowledge to maximize immediate rewards and exploring new actions to potentially discover better long-term strategies. This dilemma becomes particularly important in environments with sparse rewards or large state spaces, where undirected exploration strategies, like ϵ-greedy and entropy maximization, can lead to suboptimal policies [Thrun, 1992]. The exploration problem can be formalized as an *active learning* problem where there is a pursuit of information gain, to actively seek states and actions that reduce uncertainty about the environment [Settles, 2009]. A simple but effective strategy is optimism in the face of uncertainty [Auer et al., 2002], which operates on the principle that when an agent lacks sufficient information about certain states or actions, it should assume they may yield high rewards, thereby encouraging exploration of these uncertain regions. This strategy is often implemented in the count-based exploration setting, where an intrinsic reward is given to the agent, generally based on $1/\sqrt{N(s)}$, where $N(s)$ is the number of times the learner has visited the state Strehl and Littman [2008]. This bonus can be seen as a proxy for information gain: poorly visited states are very uncertain and could lead to high information gain, making them attractive targets for exploration while naturally diminishing the bonus as states become well-explored and their uncertainty decreases Kolter and Ng [2009].

Research problem. In continuous or high-dimensional spaces, where counting is not meaningful, as the probability of visiting the same state twice can be zero, count-based exploration becomes tricky. This fundamental challenge has led to the development of deep learning-based exploration strategies, where neural networks (NNs) that learn feature representations are used to approximate a proxy of uncertainty or pseudo-counts. Traditional representation learning approaches, while empirically successful, have a black-box aspect that complicates theoretical analysis and leads to hyperparameter

Submitted to the 18th European Workshop on Reinforcement Learning (EWRL 2025). Do not distribute.

brittleness and domain-specific tuning requirements. Successful work like Bellemare et al. [2016], Pathak et al. [2017] and Badia et al. [2020] raises an interesting research question:

Is it possible to design exploration strategies for deep RL that are simultaneously theoretically grounded, computationally efficient, and free from the complexities of representation learning?

Among these methods, Random Network Distillation (RND) introduced by Burda et al. [2018] stands out for its simplicity and computational efficiency and was the first to achieve success on the difficult Montezuma's Revenge problem. RND works by training a NN to predict the outputs of a fixed, randomly initialized target network, where the prediction error serves as a novelty signal for exploration. By exploiting random feature spaces rather than carefully learned representations, RND demonstrates that feature learning is not a prerequisite for effective exploration. Despite its empirical success, RND lacks clear theoretical connections to established methods for uncertainty estimation, and is sensitive to hyperparameters related to NNs initialization and distillation procedure, making it difficult to understand the fundamental principles behind its effectiveness and how it relates to information-theoretic approaches to exploration. A promising direction seems to be Bayesian kernel methods that offer a compelling alternative to NN-based exploration, providing theoretically grounded uncertainty quantification without the need for parameter optimization, extensive training procedures, or complex hyperparameter tuning [Srinivas et al., 2009]. However, traditional kernel methods suffer from cubic scaling with data size, a limitation that can be addressed through random features [Rahimi and Recht, 2007], and is suitable for deep RL, where dozens of samples are often required to find good strategies.

Contributions and outline. In this paper, we tackle the problem of optimism-based exploration in uncountable spaces by introducing Random Feature Information Gain (RFIG), a novel exploration bonus for RL, that is directly derived from information gain quantification in Bayesian kernel methods alongside with random features [Rahimi and Recht, 2007] capable of capturing complex nonlinear spatial patterns in high-dimensional data and approximating kernels. RFIG scales efficiently with the number of dimensions in the feature space and eliminates the need for NN training, complex hyperparameter tuning, or storage requirements. We first establish theoretical foundations by deriving RFIG from Bayesian kernel methods and random features (Section 4.1) and providing approximation error bounds (Section 4.2), which we apply to random Fourier features (Section 4.3). We then present a scheme for seamless integration in classical deep RL (Section 5.1) and demonstrate effectiveness across diverse exploration tasks (Section 5.2).

2 Related Work

Exploration remains one of the fundamental challenges in RL, particularly in environments with sparse rewards or large state spaces. This section reviews existing approaches to exploration, progressing from general methods to those most directly related to our information-based exploration approach using random features.

Exploration foundations. The exploration-exploitation trade-off was first formalized in multi-armed bandit and discrete RL. Upper Confidence Bounds (UCB) algorithms provide theoretical guarantees by maintaining confidence intervals and selecting optimistic actions [Auer et al., 2002], while Thompson Sampling offers a Bayesian alternative sampling from posterior distributions [Thompson, 1933, Chapelle and Li, 2011]. These approaches minimize the uncertainty in their objective and implicitly maximize information gain, but more recent approaches like Information Directed Sampling [Russo and Van Roy, 2014] and Minimum Empirical Divergence [Honda and Takemura, 2010] directly formalize the information gain in their objectives.

Representation learning. Modern deep RL predominantly couples exploration with representation learning, where NN learn features for uncertainty estimation. Curiosity-driven approaches like the Intrinsic Curiosity Module learn forward and inverse dynamics models, generating intrinsic rewards from prediction errors Pathak et al. [2017]. Never Give Up combines episodic and life-long novelty signals using learned embeddings Badia et al. [2020], while count-based methods learn density models for visitation estimation where information gain can be approximated through prediction gain. Information-theoretic approaches include Variational Information Maximizing Exploration , which learns probabilistic dynamics models to maximize information gain Houthooft et al. [2016], and ensemble methods that use disagreement between multiple networks or random sampling strategies as uncertainty signals Osband et al. [2016], Azizzadenesheli et al. [2018], Pathak et al. [2019]. A smaller but growing line of work separates exploration from representation learning. Hash-based methods use locality-sensitive hashing for efficient state counting Tang et al. [2017]. RND uses prediction errors on fixed random targets as exploration bonuses Burda et al. [2018], demonstrating that feature learning is not always necessary for effective exploration.

Kernel methods. The closest related works employ kernel methods that provide theoretically principled exploration frameworks by measuring uncertainty in fixed feature spaces. In multi-armed bandits, GP-UCB maintains Gaussian Process models over reward functions, selecting actions that maximize upper confidence bounds with provable regret guarantees [Srinivas et al., 2009, Valko et al., 2013, Zenati et al., 2022]. In deep RL, several kernel-based exploration methods share similarities with our approach but have important limitations that we address. Domingues et al. exploits representation learning to learn a kernel function that is used to approximate a kernel density estimator. Ma et al. [2024] leverage random Fourier features with kernel density estimation to model Beta distributions on state, to approximate their rate to be in a successful trajectory, providing exploration bonuses in sparse reward environments. In contrast, we don't maximize the same objective; our method is directly grounded by active learning and information theory, and doesn't need density estimation nor storage, which scale linearly with the number of samples, as successful and failed trajectories are stored in replay buffers and normalization is needed. Morere and Ramos [2018] propose EMU-Q, an end-to-end Bayesian kernel RL approach where posterior variance of the value function drives exploration. While they use random Fourier features for scalability, they operate in a full Bayesian kernel setting, whereas our method provides a flexible information-based exploration bonus that can be integrated with any RL algorithm. Finally, Blau et al. [2019] proposes a Bayesian curiosity module, also based on the posterior variance of kernels learned through representation learning. They suggest using random features as future work, a contribution we realize here with a scalable online update procedure accompanied by concrete error bounds.

3 Background on Information Gain, RL and Scalable Kernels

This section establishes the theoretical foundations: information gain for exploration, Bayesian kernel methods for uncertainty quantification, and random Fourier features for computational scalability

Information gain. Consider learning an unknown function $f : \mathcal{X} \to \mathbb{R}$ from noisy observations. Given data $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$ where $y_i = f(x_i) + \eta_i$, we maintain a Bayesian posterior $p(f \mid \mathcal{D}_n)$ encoding uncertainty about f. The expected information gain from querying x_* is

$$\mathrm{IG}(x_* \mid \mathcal{D}_n) = H(f \mid \mathcal{D}_n) - \mathbb{E}_{Y_*}[H(f \mid \mathcal{D}_n \cup \{(x_*, Y_*)\})] \tag{1}$$

where $H(f \mid \mathcal{D}) = -\int p(f \mid \mathcal{D}) \log p(f \mid \mathcal{D})\, df$ is the differential entropy [Cover, 1999]. This criterion, central to active learning [Settles, 2009], provides a foundation for exploration in RL.

Reinforcement learning and exploration. We consider online RL where an agent interacts with an MDP $\mathbf{M} = (\mathcal{S}, \mathcal{A}, \mathbf{r}, \mathbf{p}, \gamma)$ to learn a policy $\pi : \mathcal{S} \to \Pr(\mathcal{A})$ maximizing expected cumulative reward $J(\pi) = \mathbb{E}_{\pi,\mathbf{p}}\left[\sum_{t=0}^{\infty} \gamma^t \mathbf{r}(s_t, a_t)\right]$ [Sutton et al., 1998]. A standard exploration approach augments the extrinsic reward with an exploration bonus [Strehl and Littman, 2008]

$$\mathbf{r}_{\text{total}}(s, a) = \mathbf{r}(s, a) + \beta \mathbf{r}^+(s, a), \tag{2}$$

where $\beta > 0$ controls exploration strength. The widely-used bonus $1/\sqrt{n(s)}$, where $n(s)$ is the visit count for state s, implicitly maximizes information gain: states with fewer visits have higher uncertainty and greater potential information gain [Bellemare et al., 2016].

Bayesian kernel methods. Kernel methods address nonlinearity by implicitly mapping inputs to reproducing kernel Hilbert spaces $\mathcal{H}_k$ [Aronszajn, 1950, Schölkopf et al., 2001]. A positive semi-definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ enables computations in high-dimensional spaces using only pairwise similarities. In Bayesian kernel ridge regression [Saunders et al., 1998, Jaakkola and Haussler, 1999], given observations $\mathcal{D}_n$ and regularization parameter $\lambda > 0$ that prevents overfitting and ensures numerical stability, the posterior mean and variance are

$$\mu_n(x) = \mathbf{k}_n(x)^T(\mathbf{K}_n + \lambda \mathbf{I}_n)^{-1}\mathbf{y}_n \qquad \sigma_n^2(x) = k(x, x) - \mathbf{k}_n(x)^T(\mathbf{K}_n + \lambda \mathbf{I}_n)^{-1}\mathbf{k}_n(x) \tag{3}$$

where $\mathbf{K}_n \in \mathbb{R}^{n\times n}$ with $[\mathbf{K}_n]_{ij} = k(x_i, x_j)$ and $\mathbf{k}_n(x) = [k(x_1, x), \ldots, k(x_n, x)]^T$. This formulation is equivalent to a Gaussian process view [Williams and Rasmussen, 1995] where $f \sim \mathcal{GP}(0, k(x, x'))$ with observation noise $\sigma^2 = \lambda$. However, inverting $(\mathbf{K}_n + \lambda \mathbf{I}_n)$ requires $\mathcal{O}(n^3)$ operations, becoming prohibitive for large datasets.

Random features. Random Fourier Features (RFFs) resolve this computational bottleneck by approximating kernels with explicit finite-dimensional mappings [Rahimi and Recht, 2007]. For shift-invariant kernels $k(x, x') = k(x - x')$, Bochner's theorem [Bochner et al., 1959] enables the approximation $k(x, x') \approx \boldsymbol{\phi}(x)^T \boldsymbol{\phi}(x')$ where

$$\boldsymbol{\phi}(x) = \sqrt{\frac{2}{D}} \begin{bmatrix} \cos(\boldsymbol{\omega}_1^T x + b_1) \\ \vdots \\ \cos(\boldsymbol{\omega}_D^T x + b_D) \end{bmatrix} \tag{4}$$

with $\boldsymbol{\omega}_i \sim p(\boldsymbol{\omega})$ from the kernel's spectral density and $b_i \sim \text{Uniform}[0, 2\pi]$. For the widely-used RBF kernel $k(x, x') = \exp(-\|x - x'\|^2 / 2\ell^2)$, the lengthscale ℓ controls function smoothness and determines the spectral density $p(\boldsymbol{\omega}) = \mathcal{N}(0, \ell^{-2}\mathbf{I})$, with smaller ℓ yielding higher-frequency components. Using the Woodbury matrix identity [Woodbury, 1950] with feature matrix $\boldsymbol{\Phi}_n \in \mathbb{R}^{n\times D}$, the posterior becomes

$$\mu_n(x) = \boldsymbol{\phi}(x)^T (\boldsymbol{\Phi}_n^T \boldsymbol{\Phi}_n + \lambda \mathbf{I}_D)^{-1} \boldsymbol{\Phi}_n^T \mathbf{y}_n \tag{5}$$

$$\sigma_n^2(x) = \boldsymbol{\phi}(x)^T \boldsymbol{\phi}(x) - \boldsymbol{\phi}(x)^T (\boldsymbol{\Phi}_n^T \boldsymbol{\Phi}_n + \lambda \mathbf{I}_D)^{-1} \boldsymbol{\phi}(x) \tag{6}$$

This transforms the computational complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(D^2)$, enabling efficient uncertainty quantification that scales with feature dimension D rather than dataset size n.

4 Random Feature Information Gain

Before looking at how information gain is implemented in a RL loop to promote exploration, let's derive our Random Feature Information Gain (RFIG). The derivation proceeds in three steps: (1) express GP information gain in terms of posterior variance, (2) approximate the kernel matrix using random features, (3) apply matrix identities to obtain the final $\mathcal{O}(D^2)$ form. All detailed proofs of this section can be found in Appendix A.

4.1 Derivation

We start by recalling the information gain in the Gaussian process framework using the entropy reduction formulation, as described in (1). Consider a Gaussian process, that we defined in Section 3, $f \sim \mathcal{GP}(0, k(\cdot, \cdot))$ with observation noise $\eta \sim \mathcal{N}(0, \sigma^2)$. Given current data $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$, the posterior entropy can be expressed using the kernel matrix $\mathbf{K}_n$ with $H(f \mid \mathcal{D}_n) = \frac{1}{2} \log \det(2\pi e(\mathbf{K}_n + \sigma^2 \mathbf{I}_n)^{-1})$. When we add a new observation (x_*, y_*) to our dataset, obtaining $\mathcal{D}_{n+1} = \mathcal{D}_n \cup \{(x_*, y_*)\}$, the posterior distribution changes.

Definition 4.1 (Information gain in GP [Lawrence et al., 2002][1]**).** Information gain in GP is

$$\begin{aligned} \text{IG}(x_* \mid \mathcal{D}_n) &= H(f \mid \mathcal{D}_n) - \mathbb{E}_{Y_*}[H(f \mid \mathcal{D}_{n+1})] \\ &= \frac{1}{2} \log \det(2\pi e(\mathbf{K}_n + \sigma^2 \mathbf{I}_n)^{-1}) - \mathbb{E}_{Y_*}\left[\frac{1}{2} \log \det(2\pi e(\mathbf{K}_{n+1} + \sigma^2 \mathbf{I}_{n+1})^{-1})\right] \\ &= \frac{1}{2} \log \det(\mathbf{K}_{n+1} + \sigma^2 \mathbf{I}_{n+1}) - \frac{1}{2} \log \det(\mathbf{K}_n + \sigma^2 \mathbf{I}_n) = \boxed{\frac{1}{2} \log\left(1 + \frac{\sigma_n^2(x_*)}{\sigma^2}\right)}, \end{aligned}$$

where the last equality follows from the matrix determinant lemma applied to the block structure of $\mathbf{K}_{n+1}$, yielding the correction term $\sigma_n^2(x_*) + \sigma^2$ that simplifies to the final logarithmic form.

Remark 4.2 (Posterior variance). This derivation establishes that information gain in Gaussian processes is directly determined by the posterior variance. For small u, $\log(1 + u) \approx u$, that explain why many work, directly use the posterior variance as criterion, as maximizing information gain is equivalent to querying points with maximum posterior variance. However in our work, we consider the full information gain: the logarithm provides diminishing returns for very uncertain regions. When $\sigma_n^2(x) \gg \sigma^2$, the log saturates while variance grows unboundedly.

[1]This formulation is equivalent to what they term the "differential entropy score".

Limitations. However, computing $\sigma_n^2(x_*)$ requires inverting the $n \times n$ matrix $(\mathbf{K}_n + \sigma^2 \mathbf{I}_n)$, which scales as $\mathcal{O}(n^3)$ and becomes prohibitive for huge datasets. To address this computational bottleneck, we next develop a random feature approximation that reduces complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(D^2)$.

Proposition 4.3 (Information Gain via Random Features)**.** *Consider a random Fourier feature transformation $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^D$ that approximates a shift-invariant kernel $k(x, x') \approx \boldsymbol{\phi}(x)^T \boldsymbol{\phi}(x')$ [Rahimi and Recht, 2007]. The information gain defined in Definition 4.1 can be approximated as*

$$\hat{\mathrm{IG}}(x_* \mid \mathcal{D}_n) = \frac{1}{2} \log \left(1 + \boldsymbol{\phi}(x_*)^T (\boldsymbol{\Phi}_n^T \boldsymbol{\Phi}_n + \lambda \mathbf{I}_D)^{-1} \boldsymbol{\phi}(x_*)\right) \tag{7}$$

where $\boldsymbol{\Phi}_n \in \mathbb{R}^{n \times D}$ is the feature matrix with rows $\boldsymbol{\phi}(x_i)^T$ for $i = 1, \ldots, n$, and $\lambda > 0$ is the regularization parameter.

Proof sketch. The result follows by substituting the approximation $\mathbf{K}_n \approx \boldsymbol{\Phi}_n \boldsymbol{\Phi}_n^T$ into the GP posterior variance formula, applying the Woodbury identity to transform the $n \times n$ matrix inversion into the desired $D \times D$ form, and reinterpreting the noise σ^2 as regularization parameter λ. □

Remark 4.4 (Neural network interpretation). Our approach connects to the deep learning literature through a fundamental equivalence: training with RFFs is equivalent to optimizing a single hidden layer neural network with frozen random first-layer parameters and cosine activations. The RFF mapping $\boldsymbol{\phi}(x) = \sqrt{2/D}[\cos(\boldsymbol{\omega}_1^T x + b_1), \ldots, \cos(\boldsymbol{\omega}_D^T x + b_D)]^T$ corresponds exactly to this architecture, where only the output layer weights are learned via regression. This perspective shows that our method provides principled uncertainty quantification *without requiring* backpropagation.

4.2 Error Bounds

In order to provide theoretical guarantees for our approach, we establish error bounds for RFIG under uniform kernel convergence assumptions. Our analysis serves two key purposes: (1) quantifying how errors in kernel approximation propagate to information gain estimates, and (2) determining the number of random features D required to achieve a desired approximation accuracy ε with high probability. We proceed by first bounding the error in posterior variance estimation, then using this result to establish guarantees for information gain approximation, and finally applying our general framework to the specific case of RFFs.

Assumptions. Our analysis relies on three standard assumptions commonly employed in the random features literature [Rahimi and Recht, 2007, Sutherland and Schneider, 2015].

Assumption 4.5 (Uniform kernel approximation)**.** *The random feature map $\boldsymbol{\phi}(x) : \mathcal{X} \to \mathbb{R}^D$ provides a uniform approximation to the kernel $k(x, x')$ over the domain:*

$$\mathbb{P}\Big[\sup_{x, x' \in \mathcal{X}} |\boldsymbol{\phi}(x)^\top \boldsymbol{\phi}(x') - k(x, x')| \geq \epsilon\Big] \leq \delta(\epsilon; d, D). \tag{8}$$

Assumption 4.6 (Regularization scaling)**.** *The regularization parameter scales linearly with sample size: $\lambda = n\lambda_0$ for some $\lambda_0 > 0$.*

Assumption 4.7 (Bounded kernel)**.** *The kernel is bounded: $|k(x, x')| \leq \kappa$ for all $x, x' \in \mathcal{X}$.*

Assumption 4.5 is the core requirement for random feature methods and holds for RFFs under mild conditions on the input domain [Rahimi and Recht, 2007]. Assumption 4.6 ensures that the regularization term remain properly balanced as sample size grows, preventing regularization from either dominating or vanishing asymptotically, which is useful for deriving clean convergence rates and consistency results. Assumption 4.7 is satisfied by most practical kernels including RBF and Matérn kernels.

Posterior variance error. Since information gain is fundamentally determined by posterior variance (Equation 1), we first establish how kernel approximation errors propagate to variance estimates.

Proposition 4.8 (Posterior variance error bound)**.** *Under Assumptions 4.5, 4.6, and 4.7, the error in posterior variance estimation when using random features is bounded by:*

$$|\hat{\sigma}_n^2(x) - \sigma_n^2(x)| \leq \epsilon \left(1 + \frac{\kappa^2}{\lambda_0^2} + \frac{2\kappa}{\lambda_0} + \frac{\epsilon}{\lambda_0}\right), \tag{9}$$

where $\epsilon = \sup_{x, x' \in \mathcal{X}} |\boldsymbol{\phi}(x)^\top \boldsymbol{\phi}(x') - k(x, x')|$.

This result shows that variance estimation error scales linearly with kernel approximation quality ϵ and exhibits the expected dependence on regularization strength.

Information gain error. The connection between posterior variance and information gain enables us to translate variance errors into information gain guarantees (Lemma A.1). We now establish our main theoretical result.

Proposition 4.9 (RFIG error bound)**.** *Under Assumptions 4.5, 4.6, and 4.7, the error in RFIG approximation is bounded by:*

$$|\,\mathrm{IG}(x|\mathcal{D}_n) - \hat{\mathrm{IG}}(x|\mathcal{D}_n)| \leq \frac{\epsilon(\lambda_0 + \kappa)^2 + \epsilon^2 \lambda_0}{2n\lambda_0^3}, \tag{10}$$

where $\epsilon = \sup_{x,x' \in \mathcal{X}} |\boldsymbol{\phi}(x)^\top \boldsymbol{\phi}(x') - k(x,x')|$.

Our bound exhibits desirable theoretical properties: the error decreases with sample size n (consistency), scales with kernel approximation quality ϵ (approximation dependence), and reveals a regularization trade-off where stronger λ_0 tightens the bound but may over-smooth posteriors.

4.3 Application to Random Fourier Features

We apply our general bound to RFFs by using existing uniform convergence results.

Proposition 4.10 (RFF uniform convergence Rahimi and Recht [2007])**.** *Let* $\mathcal{X} \subset \mathbb{R}^d$ *be compact with diameter* $\mathrm{diam}(\mathcal{X})$ *and* k *a shift-invariant kernel with unit maximum and Fourier transform* $P(\omega)$*. Let* $\sigma_p^2 = \mathbb{E}_P[\|\omega\|^2]$*. For RFF mapping* $\boldsymbol{\phi}$ *and any* $\epsilon > 0$*:*

$$\Pr\left[\|\boldsymbol{\phi}^\top \boldsymbol{\phi} - k\|_\infty \geq \epsilon\right] \leq c \left(\frac{\sigma_p \,\mathrm{diam}(\mathcal{X})}{\epsilon}\right)^2 \exp\left(-\frac{D\epsilon^2}{8(d+2)}\right), \tag{11}$$

where originally $c = 256$ *in Rahimi and Recht [2007], and then refined to 66 in Sutherland and Schneider [2015].*

Combining our information gain bound with RFF convergence rates yields our main practical result:

Corollary 4.11 (Feature dimension requirement)**.** *To achieve information gain approximation error* $|\,\mathrm{IG}(x|\mathcal{D}_n) - \hat{\mathrm{IG}}(x|\mathcal{D}_n)| \leq \varepsilon$ *with probability at least* $1-\delta$*, it suffices to choose:*

$$D = \mathcal{O}\left(\frac{d}{\epsilon_k^2} \log \frac{\sigma_p \,\mathrm{diam}(\mathcal{X})}{\epsilon_k \delta}\right), \tag{12}$$

where $\epsilon_k = \dfrac{2n\lambda_0^3 \varepsilon}{(\lambda_0 + \kappa)^2}$ *when* ε *is sufficiently small.*

This result provides practical guidance for hyperparameter selection: the required feature dimension D scales linearly with problem dimension d and logarithmically with desired accuracy. Importantly, D decreases with sample size n through ϵ_k, reflecting that larger datasets permit coarser kernel approximations while maintaining the same information gain accuracy. This theoretical foundation justifies our approach and enables confident deployment in practical exploration scenarios.

5 RFIG for Efficient Exploration in RL

This paper aims to apply RFIG for improving optimism-based exploration in deep RL. This section outlines the key algorithmic components and implementation considerations that enable efficient and scalable integration with existing deep RL agents.

5.1 The Details that Matter

Algorithm 1 outlines the core RFIG integration with deep RL, successful implementation requires careful attention to numerous practical details. This subsection presents the key considerations and hyperparameter choices that determine RFIG's effectiveness in practice.

Algorithm 1: RFIG for exploration

Input: RFF Feature map $\boldsymbol{\phi}_\ell : \mathcal{X} \to \mathbb{R}^D$ with lengthscale $\ell > 0$, regularization $\lambda > 0$, subsample ratio $\rho \in (0, 1]$, environment $\mathbf{M}$, policy π, exploration scale $\beta > 0$.

Initialize RFIG matrices $\boldsymbol{\Sigma}_0 \leftarrow \lambda \mathbf{I}_D$ and $\boldsymbol{\Lambda}_0 \leftarrow \lambda^{-1}\mathbf{I}_D$;

Initialize state normalization parameters (μ_s, σ_s^2) ;

for $t \leftarrow 1, 2, \cdots$ **do**

  Collect N transitions $\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^N$ with policy π in environment $\mathbf{M}$;

  Update state normalization parameters with $\{s_i\}_{i=1}^N$, obtain normalized states $\{\bar{s}_i\}_{i=1}^N$;

  Compute information gain bonuses $\mathcal{R}^+ = \left\{ r_i^+ = \frac{1}{2} \log\left(1 + \boldsymbol{\phi}_\ell(\bar{s}_i)^\top \boldsymbol{\Lambda}_{t-1} \boldsymbol{\phi}_\ell(\bar{s}_i)\right)\right\}_{i=1}^N$;

  Subsample $\lfloor N\rho \rfloor$ states uniformly from $\{\bar{s}_i\}_{i=1}^N$ to form matrix $\boldsymbol{\Phi}_t$ with rows $\boldsymbol{\phi}_\ell(\bar{s}_j)^\top$;

  Update $\boldsymbol{\Sigma}_t \leftarrow \boldsymbol{\Sigma}_{t-1} + \boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t$, then $\boldsymbol{\Lambda}_t \leftarrow \boldsymbol{\Sigma}_t^{-1}$ via Newton-Schulz iteration (13);

  Update policy π using RL algorithm (PPO, DQN, SAC, etc.) with augmented rewards $r_i + \beta r_i^+$ from $\mathcal{D}$ and $\mathcal{R}^+$;

end

State normalization[2]**.** We maintain running statistics μ_s and σ_s^2 to normalize states as $\bar{s} = (s - \mu_s)/\sigma_s$. This prevents scale differences across dimensions from dominating kernel computations and is critical for RFF effectiveness.

Lengthscale selection. The lengthscale ℓ controls the smoothness of the uncertainty estimates and should account for the curse of dimensionality. In high-dimensional spaces, typical distances between points scale as $\sqrt{d}$ where d is the input dimension [Hvarfner et al., 2024]. Therefore, we recommend initializing $\ell \propto \sqrt{d}$.

Newton-Schulz matrix inversion. A key computational challenge in RFIG is efficiently maintaining the matrix $(\boldsymbol{\Phi}_n^T \boldsymbol{\Phi}_n + \lambda \mathbf{I}_D)^{-1}$ as new observations arrive. We employ the Newton-Schulz iteration introduced in Schulz [1933], which iteratively computes matrix inverses using

$$\mathbf{X}_{k+1} = \mathbf{X}_k(2\mathbf{I} - \mathbf{A}\mathbf{X}_k). \tag{13}$$

This method converges quadratically to $\mathbf{A}^{-1}$ when $\|\mathbf{I} - \mathbf{A}\mathbf{X}_0\|_2 < 1$ and crucially allows using the previous iteration's result as a warm start for $\mathbf{X}_0$. Compared to Sherman-Morrison or Woodbury updates, more commonly considered, Newton-Schulz offers superior numerical stability by avoiding explicit small-number divisions and provides dramatic computational savings, for $D = 512$ features and batch size $B = 128$, Newton-Schulz requires only $\approx 786K$ operations, if we consider 5 iterations, versus $\approx 33.6M$ for repeated Sherman-Morrison updates. Combined with its embarrassingly parallel structure that maps naturally to GPU architectures, Newton-Schulz is ideally suited for the frequent matrix updates required in online deep RL applications. Further details are in Appendix B.1.

Subsampling Strategy[2]**.** The subsample ratio ρ serves multiple purposes. The primary goal is to prevent information gain from shrinking too rapidly to zero as the number of samples grows, which would lead to premature exploration termination. Additionally, subsampling helps Newton-Schulz iterations converge faster since the covariance matrix $\boldsymbol{\Sigma}_t$ changes more slowly between updates, making warm starts more effective. This approach mirrors techniques in sparse Gaussian processes, where a subset of inducing points can represent the uncertainty structure of the entire dataset.

5.2 Numerical Experiments

We evaluate RFIG by integrating it with Proximal Policy Optimization (PPO) [Schulman et al., 2017], following the non-episodic exploration framework described in Burda et al. [2018] for Random Network Distillation (RND). This allows for direct comparison while leveraging proven implementation practices for intrinsic motivation in deep RL. Following the PPO+RND architecture, we augment the standard PPO objective with RFIG-based intrinsic rewards, as described in Algorithm 1. We maintain separate value networks for extrinsic and intrinsic rewards, enabling independent learning dynamics and reward normalization.

[2] These details have shown beneficial for many deep exploration strategies in Yuan et al. [2024].

Setup. We adopt global hyperparameter settings proven effective in PPO (detailed in Appendix B.2). For RFIG-specific parameters, we use $D = 1000$ random features, regularization $\lambda = 10^{-3}$, subsample ratio $\rho = 3.13\%$, and lengthscale $\ell = \sqrt{d}$. The exploration coefficient β is set to 0.5, without conducting any hyperparameter optimization. The most sensitive parameters are lengthscale ℓ and exploration scale β. Regularization λ and feature dimension D are less critical once set in reasonable ranges. We evaluate RFIG across three domains designed to test exploration capabilities. Classic control tasks (Acrobot, MountainCar) provide baseline comparisons in low-dimensional settings [Lange, 2022]. For more challenging continuous control, we use sparse reward variants of locomotion tasks in Brax environments [Freeman et al., 2021], where agents receive milestone rewards only upon reaching specific distance thresholds, creating challenging exploration scenarios (Appendix B.3). Additionally, we test on the PointMaze environments suite [Park et al., 2024], which are navigation tasks. All experiments are evaluated using 32 random seeds and 32 parallel environments, with an unroll length of 128 steps.

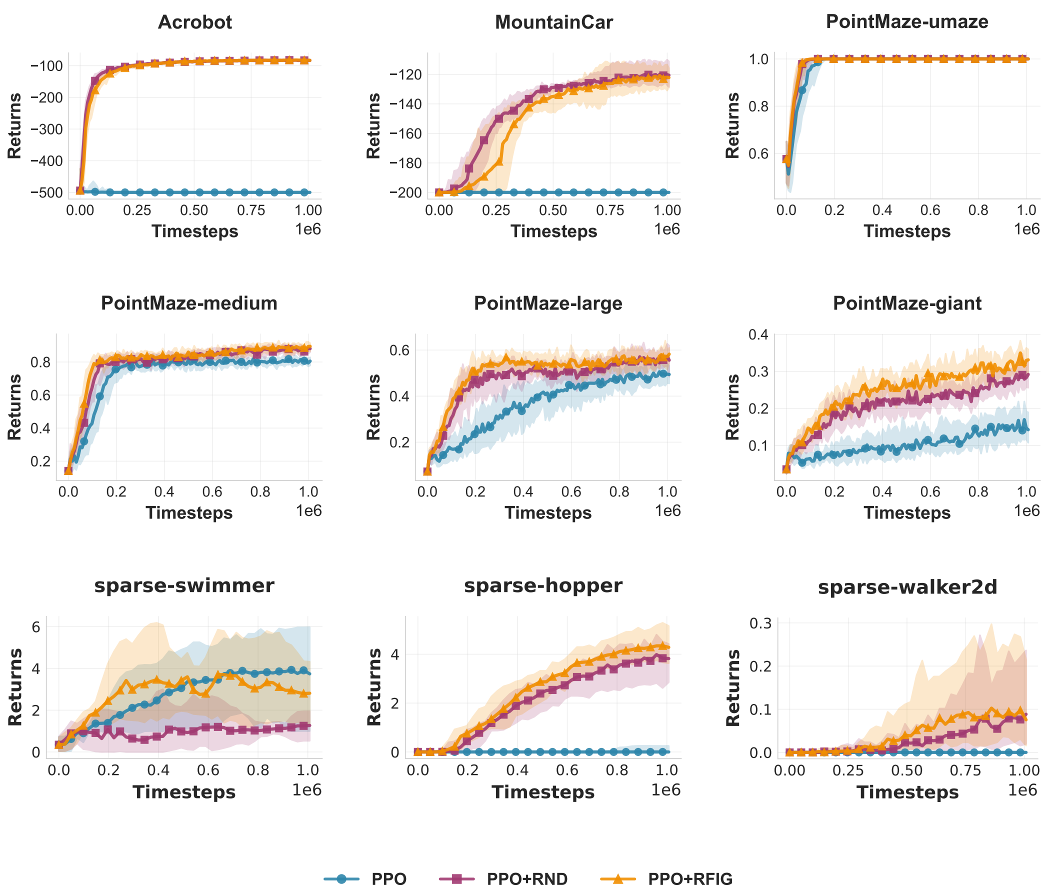

Figure 1: Comparing PPO, PPO+RND, and PPO+RFIG. Solid lines represent the interquartile mean, with shaded areas indicating the 25th-75th percentiles across 32 random seeds.

Discussion. Our experimental evaluation (Figure 1) demonstrates that PPO+RFIG achieves competitive performance with PPO+RND across diverse exploration challenges while offering superior theoretical foundations. In classic control tasks (Acrobot, MountainCar), both exploration methods significantly outperform vanilla PPO with simple entropy coefficient, confirming that RFIG provides effective exploration bonuses in low-dimensional settings. The advantages of RFIG become more pronounced in complex navigation tasks, particularly in PointMaze-giant, where RFIG demonstrates superior sample efficiency compared to RND. In sparse Brax environments, RFIG exhibits modest improvements over RND. These results highlight RFIG's key advantage: delivering exploration performance comparable to state-of-the-art methods while providing rigorous theoretical guarantees rooted in information theory, unlike RND, which relies on a more heuristic approach. However, our evaluation has limitations, we focus primarily on navigation and locomotion tasks. Future work should explore RFIG's applicability in manipulation tasks, partial observability settings, as well as image-domain environments, and investigate how hyperparameters, like lengthscale, improve performance. Notably, RFIG introduces only approximately 10% computational overhead over vanilla PPO due to matrix inversion and bonus computation, making it viable for large-scale applications.

6 Conclusion

We introduced Random Feature Information Gain (RFIG), a theoretically grounded exploration method that achieves competitive performance without complex representation learning. By leveraging Bayesian kernel methods and random Fourier features, RFIG provides a theoretically grounded alternative that maintains computational efficiency while avoiding the black-box aspects of neural network-based uncertainty estimation.

Key contributions. Our work demonstrates that rigorous kernel methods can achieve competitive empirical performance with state-of-the-art exploration algorithms while providing superior theoretical interpretability. RFIG's success across classic control, navigation, and sparse locomotion tasks, combined with reasonable computational overhead, suggests that the field's trend toward increasingly complex representation learning may not be necessary for effective exploration. The method's theoretical foundations offer mathematical rigor often lacking in modern deep RL exploration strategies.

Broader impact. The information gain estimation framework developed for RFIG extends beyond EL applications. The same principled approach to uncertainty quantification and information-theoretic bonuses can be applied to active learning, Bayesian optimization, and other sequential decision-making problems where exploration-exploitation trade-offs are crucial.

Future directions. Several promising research directions emerge from this work. First, see how RFIG extend to high-dimensional observation spaces, particularly image-based environments like Atari games (e.g., Montezuma's Revenge), would test whether our approach can achieve state-of-the-art results in challenging visual domains. Second, developing adaptive kernel selection mechanisms that learn optimal lengthscales during training could further improve performance and avoid hyperparameter search. Finally, the same information-theoretic framework could be adapted for offline RL, where conservative "anti-exploration" strategies that avoid out-of-distribution states are preferred over optimistic exploration.

References

Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient exploration through bayesian deep q-networks. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–9. IEEE, 2018.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andew Bolt, et al. Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038*, 2020.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and hashing. In *Advances in neural information processing systems*, pages 2611–2619, 2016.

Tom Blau, Lionel Ott, and Fabio Ramos. Bayesian curiosity for efficient exploration in reinforcement learning. *arXiv preprint arXiv:1911.08701*, 2019.

Salomon Bochner et al. *Lectures on Fourier integrals*, volume 42. Princeton University Press, 1959.

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

Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. *Advances in neural information processing systems*, 24, 2011.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Omar Darwiche Domingues, Corentin Tallec, Remi Munos, and Michal Valko. Density-based bonuses on learned representations for reward-free exploration in deep reinforcement learning. In *ICML 2021 Workshop on Unsupervised Reinforcement Learning*.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL `http://github.com/google/brax`.

Junya Honda and Akimichi Takemura. An asymptotically optimal bandit algorithm for bounded support models. In *COLT*, pages 67–79. Citeseer, 2010.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. *Advances in neural information processing systems*, 29, 2016.

Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla bayesian optimization performs great in high dimensions. *arXiv preprint arXiv:2402.02229*, 2024.

Tommi S Jaakkola and David Haussler. Probabilistic kernel regression models. In *Seventh International Workshop on Artificial Intelligence and Statistics*. PMLR, 1999.

J Zico Kolter and Andrew Y Ng. Near-bayesian exploration in polynomial time. In *Proceedings of the 26th annual international conference on machine learning*, pages 513–520, 2009.

Robert Tjarko Lange. gymnax: A JAX-based reinforcement learning environment library, 2022. URL `http://github.com/RobertTLange/gymnax`.

Neil Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse gaussian process methods: The informative vector machine. *Advances in neural information processing systems*, 15, 2002.

Haozhe Ma, Zhengding Luo, Thanh Vinh Vo, Kuankuan Sima, and Tze-Yun Leong. Highly efficient self-adaptive reward shaping for reinforcement learning. *arXiv preprint arXiv:2408.03029*, 2024.

Philippe Morere and Fabio Ramos. Bayesian RL for goal-only rewards. In *Conference on Robot Learning*, 2018.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. *arXiv preprint arXiv:2410.20092*, 2024.

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

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

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Daniel Russo and Benjamin Van Roy. Learning to optimize via information-directed sampling. *Advances in neural information processing systems*, 27, 2014.

Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. 1998.

Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001.

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

Günther Schulz. Iterative berechung der reziproken matrix. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 13(1):57–59, 1933.

Burr Settles. Active learning literature survey. 2009.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

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.

Danica J Sutherland and Jeff Schneider. On the error of random fourier features. *arXiv preprint arXiv:1506.02785*, 2015.

Richard S Sutton, Andrew G Barto, et al. Introduction to reinforcement learning, vol. 135, 1998.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Sebastian B Thrun. *Efficient exploration in reinforcement learning*. Carnegie Mellon University, 1992.

Michal Valko, Nathaniel Korda, Rémi Munos, Ilias Flaounas, and Nelo Cristianini. Finite-time analysis of kernelised contextual bandits. *arXiv preprint arXiv:1309.6869*, 2013.

Christopher Williams and Carl Rasmussen. Gaussian processes for regression. *Advances in neural information processing systems*, 8, 1995.

Max A Woodbury. *Inverting modified matrices*. Department of Statistics, Princeton University, 1950.

Mingqi Yuan, Roger Creus Castanyer, Bo Li, Xin Jin, Wenjun Zeng, and Glen Berseth. Rlexplore: Accelerating research in intrinsically-motivated reinforcement learning. *arXiv preprint arXiv:2405.19548*, 2024.

Houssam Zenati, Alberto Bietti, Eustache Diemert, Julien Mairal, Matthieu Martin, and Pierre Gaillard. Efficient kernelized ucb for contextual bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 5689–5720. PMLR, 2022.

A Full Proofs

A.1 Information Gain via Random Features

Poof of Proposition 4.3. Using the random feature approximation $\mathbf{K}_n \approx \mathbf{\Phi}_n \mathbf{\Phi}_n^T$, the posterior variance becomes

$$\sigma_n^2(x_*) = \boldsymbol{\phi}(x_*)^T \boldsymbol{\phi}(x_*) - \boldsymbol{\phi}(x_*)^T (\mathbf{\Phi}_n \mathbf{\Phi}_n^T + \sigma^2 \mathbf{I}_n)^{-1} \boldsymbol{\phi}(x_*)$$

Applying the Woodbury identity:

$$(\mathbf{\Phi}_n \mathbf{\Phi}_n^T + \sigma^2 \mathbf{I}_n)^{-1} = \frac{1}{\sigma^2} \mathbf{I}_n - \frac{1}{(\sigma^2)^2} \mathbf{\Phi}_n (\mathbf{\Phi}_n^T \mathbf{\Phi}_n + \sigma^2 \mathbf{I}_D)^{-1} \mathbf{\Phi}_n^T$$

Substituting and simplifying:

$$\sigma_n^2(x_*) = \phi(x_*)^T\phi(x_*) - \frac{1}{\sigma^2}\phi(x_*)^T\phi(x_*) + \frac{1}{\sigma^2}\phi(x_*)^T(\Phi_n^T\Phi_n + \sigma^2\mathbf{I}_D)^{-1}\phi(x_*) \quad (14)$$

$$= \sigma^2\phi(x_*)^T(\Phi_n^T\Phi_n + \sigma^2\mathbf{I}_D)^{-1}\phi(x_*) \quad (15)$$

$$\frac{\sigma_n^2(x_*)}{\sigma^2} = \phi(x_*)^T(\Phi_n^T\Phi_n + \sigma^2\mathbf{I}_D)^{-1}\phi(x_*) \quad (16)$$

Substituting back into the information gain formula of Gaussian Processes yields $\frac{1}{2}\log\Big(1 + \phi(x_*)^T(\Phi_n^T\Phi_n + \sigma^2\mathbf{I}_D)^{-1}\phi(x_*)\Big)$. We can finally reinterpret the observation noise variance σ^2 as a regularization parameter λ, giving the desired result. □

A.2 Posterior Variance Error Bound

Proof of Proposition 4.8. Let consider the true posterior variance, $\sigma_n^2(x) = k(x,x) - \mathbf{k}^\top\mathbf{K}^{-1}\mathbf{k}$, with $\mathbf{k} = \mathbf{k}_n(x)$ and $\mathbf{K} = \mathbf{K}_n + \lambda I_n$, considering the approximation $k(x,x') \approx \phi(x)^T\phi(x')$, we can consider our approximated posterior variance $\hat{\sigma}_n^2(x)$ as a perturbation of the true one and define $\hat{\mathbf{k}} = \mathbf{k} + \mathbf{\Delta_k}$ and $\hat{\mathbf{K}} = \mathbf{K} + \mathbf{\Delta_K}$. Let's expand the following difference

$$\hat{\mathbf{k}}^\top\hat{\mathbf{K}}^{-1}\hat{\mathbf{k}} - \mathbf{k}^\top\mathbf{K}^{-1}\mathbf{k} = (\mathbf{k}+\mathbf{\Delta_k})^\top(\mathbf{K}+\mathbf{\Delta_K})^{-1}(\mathbf{k}+\mathbf{\Delta_k}) - \mathbf{k}^\top\mathbf{K}^{-1}\mathbf{k} \quad (17)$$

$$= \mathbf{k}^\top\hat{\mathbf{K}}^{-1}\mathbf{k} + \mathbf{k}^\top\hat{\mathbf{K}}^{-1}\mathbf{\Delta_k} + \mathbf{\Delta_k}^\top\hat{\mathbf{K}}^{-1}\mathbf{k} + \mathbf{\Delta_k}^\top\hat{\mathbf{K}}^{-1}\mathbf{\Delta_k} - \mathbf{k}^\top\mathbf{K}^{-1}\mathbf{k} \quad (18)$$

$$= \mathbf{k}^\top(\hat{\mathbf{K}}^{-1} - \mathbf{K}^{-1})\mathbf{k} + 2\mathbf{k}^\top\hat{\mathbf{K}}^{-1}\mathbf{\Delta_k} + \mathbf{\Delta_k}^\top\hat{\mathbf{K}}^{-1}\mathbf{\Delta_k} \quad (19)$$

$$\leq |\underbrace{\mathbf{k}^\top(\hat{\mathbf{K}}^{-1} - \mathbf{K}^{-1})\mathbf{k}}_{\text{Matrix perturbation } t_1}| + |\underbrace{2\mathbf{k}^\top\hat{\mathbf{K}}^{-1}\mathbf{\Delta_k}}_{\text{Cross term } t_2}| + |\underbrace{\mathbf{\Delta_k}^\top\hat{\mathbf{K}}^{-1}\mathbf{\Delta_k}}_{\text{Vector term } t_3}|. \quad (20)$$

where we used the symmetry property $\mathbf{k}^\top\hat{\mathbf{K}}^{-1}\mathbf{\Delta_k} = \mathbf{\Delta_k}^\top\hat{\mathbf{K}}^{-1}\mathbf{k}$ and triangle inequality.

Bounding t_1: Since the smallest eigenvalue of $\mathbf{K}^{-1}$ $\hat{\mathbf{K}}^{-1}$ is λ, $\|\mathbf{k}\|_2 \leq \sqrt{n}\kappa$, and using the inverse matrix perurbation bound for $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{E}$, $\|\hat{\mathbf{A}}^{-1} - \mathbf{A}^{-1}\|_2 \leq \|\mathbf{A}^{-1}\|_2 \cdot \|\hat{\mathbf{A}}^{-1}\|_2 \cdot \|\mathbf{E}\|_2$, we have

$$|\mathbf{k}^\top(\hat{\mathbf{K}}^{-1} - \mathbf{K}^{-1})\mathbf{k}| \leq \|\hat{\mathbf{K}}^{-1}\|_2 \cdot \|\mathbf{\Delta_K}\|_2 \cdot \|\mathbf{K}^{-1}\|_2 \cdot \|\mathbf{k}\|_2^2 \leq \frac{\epsilon\kappa^2 n^2}{\lambda^2}. \quad (21)$$

Bounding t_2 and t_3: Similarly to t_1, we can bound the two other terms with

$$|2\mathbf{k}^\top\hat{\mathbf{K}}^{-1}\mathbf{\Delta_k}| \leq \frac{2\epsilon n\kappa}{\lambda}, \qquad |\mathbf{\Delta_k}^\top\hat{\mathbf{K}}^{-1}\mathbf{\Delta_k}| \leq \frac{\epsilon^2 n}{\lambda}. \quad (22)$$

Finally, $|\hat{\sigma}_n^2(x) - \sigma_n^2(x)| \leq \epsilon + \frac{\epsilon\kappa^2 n^2}{\lambda^2} + \frac{2\epsilon n\kappa}{\lambda} + \frac{\epsilon^2 n}{\lambda} = \epsilon + \frac{\epsilon\kappa^2}{\lambda_0^2} + \frac{2\epsilon\kappa}{\lambda_0} + \frac{\epsilon^2}{\lambda_0}$, using Assumption 4.6. □

A.3 RFIG Error Bound

To bound RFIG, we can directly use the established bound for posterior variance, by using the following lemma:

Lemma A.1 (Shifted Logarithmic Difference bound)**.** *For any $a, b > 0$, we have*

$$|\log(1+a) - \log(1+b)| \leq |a - b| \quad (23)$$

Proof. On the interval between a and b, there exists c between a and b such that $\log(1+a) - \log(1+b) = f'(c)(a-b) = \frac{a-b}{1+c}$. Since $\min(a,b) \leq c \leq \max(a,b)$, we have $\frac{1}{1+\max(a,b)} \leq \frac{1}{1+c} \leq \frac{1}{1+\min(a,b)}$. Therefore, $\left|\frac{a-b}{1+c}\right| \leq \frac{|a-b|}{1+\min(a,b)}$, we have $a, b > 0$, which completes the proof. □

Now, we have all the elements to obtain deterministic upper bound bound on RFIG.

Proof of Proposition 4.9. By applying the Lemma X, we have

$$|\hat{\mathrm{IG}}(x|\mathcal{D}_n) - \mathrm{IG}(x|\mathcal{D}_n)| \leq \frac{\mathbf{\Delta}_{\sigma_n^2}}{2\lambda} = \frac{\epsilon + \frac{\epsilon\kappa^2}{\lambda_0^2} + \frac{2\epsilon\kappa}{\lambda_0} + \frac{\epsilon^2}{\lambda_0}}{2\lambda} \tag{24}$$

$$= \frac{\epsilon\left[\left(1 + \frac{\kappa}{\lambda_0}\right)^2 + \frac{\epsilon}{\lambda_0}\right]}{2\lambda} \tag{25}$$

$$= \frac{\epsilon\left[\frac{(\lambda_0+\kappa)^2+\epsilon\lambda_0}{\lambda_0^2}\right]}{2\lambda} \tag{26}$$

$$= \frac{\epsilon(\lambda_0 + \kappa)^2 + \epsilon^2\lambda_0}{2n\lambda_0^3} \tag{27}$$

That ends the proof. □

A.4 High Probability RFIG Bound

Proof of Proposition 4.11. We aim to find when the information gain error is at least ε:

$$\frac{\epsilon(\lambda_0 + \kappa)^2 + \epsilon^2\lambda_0}{2n\lambda_0^3} \geq \varepsilon \tag{28}$$

$$\epsilon(\lambda_0 + \kappa)^2 + \epsilon^2\lambda_0 \geq \varepsilon 2n\lambda_0^3 \tag{29}$$

$$\epsilon(\lambda_0 + \kappa)^2 + \epsilon^2\lambda_0 - \varepsilon 2n\lambda_0^3 \geq 0 \tag{30}$$

This is a quadratic inequality in ϵ. The quadratic $f(\epsilon) = \lambda_0\epsilon^2 + (\lambda_0 + \kappa)^2\epsilon - 2n\lambda_0^3\varepsilon$ has for root:

$$\epsilon \geq \frac{-(\lambda_0 + \kappa)^2 + \sqrt{(\lambda_0 + \kappa)^4 + 8n\lambda_0^4\varepsilon}}{2\lambda_0}, \tag{31}$$

since $\lambda_0 > 0$, the parabola opens upward. Setting $\kappa = 1$ (Proposition 4.10), ends the proof. □

B Numerical Experiments

B.1 Newton-Schulz iterations

The Newton-Schulz method provides an iterative approach to matrix inversion that is particularly well-suited for our kernel matrix updates. Due to JAX's compilation and parallelization constraints,

Algorithm 2: Newton-Schulz Matrix Inversion Update

Input: Previous inverse $\mathbf{X}_{old}$, matrix update $\mathbf{\Phi}_t$, regularization λ
$\mathbf{A} \leftarrow \mathbf{\Phi}_t^T\mathbf{\Phi}_t + \lambda\mathbf{I}$;
$\mathbf{X}_0 \leftarrow \mathbf{X}_{old}$ (warm start);
for $k = 1, 2, \ldots, K$ **do**
| $\mathbf{X}_k \leftarrow \mathbf{X}_{k-1}(2\mathbf{I} - \mathbf{A}\mathbf{X}_{k-1})$;
end
return $\mathbf{X}_K$

we implement a fixed number of Newton-Schulz iterations ($K = 20$) rather than iterating until convergence. In practice, we observe that 20 iterations provide sufficient accuracy for information gain estimation while maintaining computational efficiency across all experimental environments.

B.2 Hyperparameter Configuration

Table 1 presents the complete hyperparameter configuration used for PPO experiments across all environments. For RND baseline comparisons, we use an embedding size of 256, hidden layer sizes of (256, 256), a bonus learning rate of 1e-4, with ReLU activations, following standard RND implementation practices.

Table 1: PPO hyperparameters used in all experiments.

Parameter	**Value**
Training Configuration	
Total timesteps	1,000,000
Number of environments	32
Steps per environment	128
Evaluation frequency	24,576
Anneal learning rate	True
PPO Algorithm	
Learning rate	0.0003
Number of epochs	4
Number of minibatches	32
Clip ratio (ϵ)	0.2
Value function coefficient	0.5
Entropy coefficient	0.01
Maximum gradient norm	0.5
GAE & Discounting	
Discount factor (γ)	0.99
GAE lambda (λ)	0.95
Normalization	
Normalize observations	True
Normalize intrinsic rewards	True
Network Architecture	
Activation function	Swish
Hidden layer sizes	(64, 64)

B.3 Milestone Reward Wrapper

To create sparse reward variants of Brax locomotion environments, we implement a `MilestoneRewardWrapper` that transforms dense reward signals into sparse, milestone-based rewards. This wrapper provides rewards only when the agent reaches specific distance milestones during locomotion, creating challenging exploration scenarios where traditional dense rewards are unavailable. The wrapper operates by tracking the agent's forward displacement from its initial position and providing rewards at fixed distance intervals. Specifically, it:

1. Records the agent's initial position at environment reset
2. Monitors the agent's current position throughout the episode
3. Calculates the total distance traveled as the difference between current and initial positions
4. Awards rewards when the agent crosses predefined distance milestones

The milestone reward r_t at timestep t is computed as:

$$r_t = \begin{cases} \alpha \cdot (m_t - m_{t-1}) & \text{if } m_t > m_{t-1} \\ 0 & \text{otherwise} \end{cases} \tag{32}$$

where $m_t = \lfloor d_t/\delta \rfloor$ represents the current milestone, d_t is the distance traveled, δ is the milestone distance interval, and α is the reward scale factor. The wrapper accepts three key parameters:

- **milestone_distance** ($\delta = 1.0$): Distance interval between consecutive milestones
- **reward_scale** ($\alpha = 1.0$): Scale factor applied to milestone rewards
- **position_fn**: Function extracting agent position from environment state (defaults to x-coordinate of the first body)

This design creates environments where agents receive no immediate feedback for small movements but are rewarded for achieving meaningful locomotion progress, making these tasks particularly challenging for exploration strategies. For reproducibility, we provide the complete implementation of the `MilestoneRewardWrapper`:

```python
from typing import Callable, Optional
from brax.envs import PipelineEnv, State, Wrapper
import jax
from jax import numpy as jp

class MilestoneRewardWrapper(Wrapper):
    """Wrapper that adds milestone-based rewards to any Brax environment.
    This wrapper gives a reward whenever the agent reaches specified distance
    milestones (e.g., every 1.0 unit of forward movement).
    """
    def __init__(
            self,
            env: PipelineEnv,
            milestone_distance: float = 1.0,
            reward_scale: float = 1.0,
            position_fn: Optional[Callable[[State], jp.ndarray]] = lambda state: state.pipeline_state.x.pos[0, 0],
    ):
        """Initializes the milestone reward wrapper.
        Args:
          env: The environment to wrap.
          milestone_distance: Distance between reward milestones.
          reward_scale: Scale factor for milestone rewards.
          position_fn: Function that extracts position from state.
                       Default extracts x position from first body.
        """
        super().__init__(env)
        self._milestone_distance = milestone_distance
        self._reward_scale = reward_scale
        self._position_fn = position_fn

    def reset(self, rng: jax.Array) -> State:
        """Resets the environment and initializes milestone reward tracking."""
        state = self.env.reset(rng)
        # Get initial position
        initial_position = self._position_fn(state)
        # Add milestone reward tracking info
        info = state.info.copy()
        info.update({
            'initial_position': initial_position,
            'last_milestone': 0.0,
            'total_milestones': 0,
            'distance_traveled': 0.0,
            'current_milestone': 0.0,
        })
        return state.replace(info=info)

    def step(self, state: State, action: jax.Array) -> State:
        """Steps the environment and adds milestone rewards."""
        # Get tracking info
        initial_position = state.info.get('initial_position')
        last_milestone = state.info.get('last_milestone', 0.0)
        total_milestones = state.info.get('total_milestones', 0)

        # Step the environment
        next_state = self.env.step(state, action)

        # Get current position and calculate distance traveled
        current_position = self._position_fn(next_state)
```

```python
        distance_traveled = current_position - initial_position

        # Calculate the current milestone
        current_milestone = jp.floor(distance_traveled / self._milestone_distance)

        # Check if we've reached a new milestone
        new_milestone_reached = current_milestone > last_milestone

        # Calculate milestone reward
        reward = jp.where(
            new_milestone_reached,
            self._reward_scale * (current_milestone - last_milestone),
            0.0
        )

        # Update the total milestones count
        total_milestones = jp.where(
            new_milestone_reached,
            total_milestones + jp.int32(current_milestone - last_milestone),
            total_milestones
        )

        # Update the last milestone
        last_milestone = jp.where(new_milestone_reached, current_milestone, last_milestone)

        # Update info
        info = next_state.info.copy()
        info.update({
            'initial_position': initial_position,
            'last_milestone': last_milestone,
            'total_milestones': total_milestones,
            'distance_traveled': distance_traveled,
            'current_milestone': current_milestone,
        })

        return next_state.replace(reward=reward, info=info)
```

Listing 1: MilestoneRewardWrapper Implementation