Representation-Driven Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present a representation-driven framework for reinforcement learning. By representing policies as estimates of their expected values, we leverage techniques from contextual bandits to guide exploration and exploitation. Particularly, embedding a policy network into a linear feature space allows us to reframe the exploration-exploitation problem as a representation-exploitation problem, where good policy representations enable optimal exploration. We demonstrate the effectiveness of this framework through its application to evolutionary and policy gradient-based approaches, leading to significantly improved performance compared to traditional methods. Our framework provides a new perspective on reinforcement learning, highlighting the importance of policy representation in determining optimal exploration-exploitation strategies.

1 Introduction

Reinforcement learning (RL) is a field in machine learning in which an agent learns to maximize a reward through interactions with an environment. The agent maps its current state into action and receives a reward signal. Its goal is to maximize the cumulative sum of rewards over some predefined (possibly infinite) horizon [37]. This setting fits many real-world applications such as recommendation systems [20], board games [36], computer games [24], and robotics [29].

A large amount of contemporary research in RL focuses on gradient-based policy search methods [38, 35, 32, 33, 14]. Nevertheless, these methods optimize the policy **locally** at specific states and actions. Salimans et al. [31] have shown that such optimization methods may cause high variance updates in long horizon problems, while Tessler et al. [42] have shown possible convergence to suboptimal solutions in continuous regimes. Moreover, policy search methods are commonly sample inefficient, particularly in hard exploration problems, as policy gradient methods usually converge to areas of high reward, without sacrificing exploration resources to achieve a far-reaching sparse reward.

In this work, we present Representation-Driven Reinforcement Learning (RepRL) – a new framework for policy-search methods, which utilizes theoretically optimal exploration strategies in a learned latent space. Particularly, we reduce the policy search problem to a contextual bandit problem, using a mapping from policy space to a linear feature space. Our approach leverages the learned linear space to optimally tradeoff exploration and exploitation using well-established algorithms from the contextual bandit literature [1, 4]. By doing so, we reframe the exploration-exploitation problem to a representation-exploitation problem, for which good policy representations enable optimal exploration.

We demonstrate the effectiveness of our approach through its application to both evolutionary and policy gradient-based approaches – demonstrating significantly improved performance compared to traditional methods. Empirical experiments on the MuJoCo [43] and MinAtar [47] show the benefits of our approach, particularly in sparse reward settings. While our framework does not make the

Submitted to 16th European Workshop on Reinforcement Learning (EWRL 2023). Do not distribute.

exploration problem necessarily easier, it provides a new perspective on reinforcement learning, shifting the focus to policy representation in the search for optimal exploration-exploitation strategies.

2 Preliminaries

We consider the infinite-horizon discounted Markov Decision Process (MDP). An MDP is defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, T, \beta, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition kernel, $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the reward function, $\beta \in \Delta(\mathcal{S})$ is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. A stationary policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, maps states into a distribution over actions. We denote by Π the set of stationary stochastic policies, and the history of policies and trajectories up to episode k by $\mathcal{H}_k$. Finally, we denote $S = |\mathcal{S}|$ and $A = |\mathcal{A}|$.

The return of a policy is a random variable defined as the discounted sum of rewards

$$G(\pi) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t), \tag{1}$$

where $s_0 \sim \beta, a_t \sim \pi(s_t), s_{t+1} \sim T(s_t, a_t)$, and the policy's value is its mean, i.e., $v(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid \beta, \pi, T]$. An optimal policy maximizes the value, i.e., $\pi^* \in \arg\max_{\pi \in \Pi} v(\pi)$.

We similarly define the per-state value function, $v(\pi, s)$ as $v(\pi, s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, \pi, T]$, and note that $v(\pi) = \mathbb{E}_{s \sim \beta}[v(\pi, s)]$.

Finally, we denote the discounted state-action frequency distribution w.r.t. π by

$$\rho^\pi(s, a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t Pr\Big(s_t = s, a_t = a | \beta, \pi, T\Big),$$

and let $\mathcal{K} = \{\rho^\pi : \pi \in \Pi\}$.

2.1 Linear Bandits

In this work, we consider the linear bandit framework as defined in Abbasi-Yadkori et al. [1]. At each time t, the learner is given a decision set $D_t \subseteq \mathbb{R}^d$, which can be adversarially and adaptively chosen. The learner chooses an action $x_t \in D_t$ and receives a reward r_t, whose mean is linear w.r.t x_t, i.e., $\mathbb{E}[r_t \mid x_t] = \langle x_t, w \rangle$ for some unknown parameter vector $w \in \mathbb{R}^d$.

A general framework for solving the linear bandit problem is the "Optimism in the Face of Uncertainty Linear bandit algorithm" (OFUL, Abbasi-Yadkori et al. [1]). There, a linear regression estimator is constructed each round as follows:

$$\begin{aligned} \hat{w}_t &= V_t^{-1} b_t, \\ V_t &= V_{t-1} + x_t x_t^\top, \\ b_t &= b_{t-1} + x_t y_t, \end{aligned} \tag{2}$$

where y_t, x_t are the noisy reward signal and chosen action at time t, respectively, and $V_0 = \lambda I$ for some positive parameter $\lambda > 0$.

It can be shown that, under mild assumptions, and with high probability, the self-normalizing norm $\|\hat{w}_t - w\|_{V_t}$ can be bounded from above [1]. OFUL then proceeds by taking an optimistic action $(x_t, \bar{w}_t) \in \arg\max_{x \in D_t, \bar{w} \in \mathcal{C}_t} \langle x, \bar{w} \rangle$, where $\mathcal{C}_t$ is a confidence set induced by the aforementioned bound on $\|\hat{w}_t - w\|_{V_t}$. In practice, a softer version is used in Chu et al. [9], where an action is selected optimistically according to

$$x_t \in \arg\max_{x \in D_t} \langle x, \hat{w}_t \rangle + \alpha \sqrt{x^T V_t^{-1} x}, \tag{OFUL}$$

where $\alpha > 0$ controls the level of optimism.

Alternatively, linear Thompson sampling (TS, Abeille & Lazaric [2]) shows it is possible to converge to an optimal solution with sublinear regret, even with a constant probability of optimism. This is

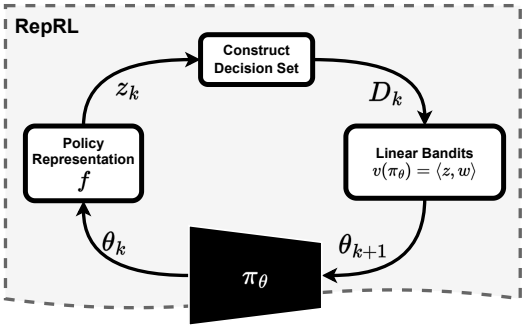

Figure 1: RepRL scheme. Composed of 4 stages: representation of the parameters, constructing a decision set, choosing the best arm using an off-the-shelf linear bandit algorithm, collect data with the chosen policy.

achieved through the sampling of a parameter vector from a normal distribution, which is determined by the confidence set $\mathcal{C}_t$. Specifically, linear TS selects an action according to

$$x_t \in \arg\max_{x \in D_t} \langle x, \tilde{w}_t \rangle, \ \tilde{w}_t \sim \mathcal{N}\big(\hat{w}_t, \sigma^2 V_t^{-1}\big), \tag{TS}$$

where $\sigma > 0$ controls the level of optimism. We note that for tight regret guarantees, both α and σ need to be chosen to respect the confidence set $\mathcal{C}_t$. Nevertheless, it has been shown that tuning these parameters can improve performance in real-world applications [9].

3 RL as a Linear Bandit Problem

Classical methods for solving the RL problem attempted to use bandit formulations [13]. There, the set of policies Π reflects the set of arms, and the value $v(\pi)$ is the expected bandit reward. Unfortunately, such a solution is usually intractable due to the exponential number of policies (i.e., bandit actions) in Π.

Alternatively, we consider a linear bandit formulation of the RL problem. Indeed, it is known that the value can be expressed in linear form as

$$v(\pi) = \mathbb{E}_{(s,a)\sim\rho^\pi}[r(s,a)] = \langle \rho^\pi, r \rangle. \tag{3}$$

Here, any $\rho^\pi \in \mathcal{K}$ represents a possible action in the linear bandit formulation [1]. Notice that $|\mathcal{K}| = |\Pi|$, as any policy $\pi \in \Pi$ can be written as $\pi(a|s) = \frac{\rho^\pi(s,a)}{\sum_{a'} \rho^\pi(s,a')}$, rendering the problem intractable. Nevertheless, this formulation can be relaxed using a lower dimensional embedding of ρ^π and r. As such, we make the following assumption.

Assumption 3.1 (Linear Embedding)**.** There exist a mapping $f : \Pi \to \mathbb{R}^d$ such that $v(\pi) = \langle f(\pi), w \rangle$ for all $\pi \in \Pi$ and some unknown $w \in \mathbb{R}^d$.

We note that Assumption 3.1 readily holds when $d = SA$ for $f(\pi) \equiv \rho^\pi$ and $w = r$. For efficient solutions, we consider environments for which the dimension d is relatively low, i.e., $d \ll SA$.

Note that neural bandit approaches also consider linear representations [30]. Nevertheless, these methods use **mappings from states** $\mathcal{S} \mapsto \mathbb{R}^d$, whereas we consider **mapping entire policies** $\Pi \mapsto \mathbb{R}^d$ (i.e., embedding the *function* π). Learning a mapping f can be viewed as trading the effort of finding good exploration strategies in deep RL problems to finding a good representation. We emphasize that we do not claim it to be an *easier* task, but rather a *different* viewpoint of the problem, for which possible new solutions can be derived. Similar to work on neural-bandits [30], finding such a mapping requires alternating between representation learning and exploration.

3.1 RepRL

We formalize a representation-driven framework for RL, inspired by linear bandits (Section 2.1) and Assumption 3.1. We parameterize the policy π and mapping f using neural networks, π_θ and f_ϕ, respectively. Here, a policy π_θ is represented in lower-dimensional space as $f_\phi(\pi_\theta)$. Therefore, searching in policy space is equivalent to searching in the parameter space. With slight abuse of notation, we will denote $f_\phi(\pi_\theta) = f_\phi(\theta)$.

Algorithm 1 RepRL

Init: $\mathcal{H}_0 \leftarrow \emptyset$, π_{θ_0}, f_{ϕ_0} randomly initialized
for $k = 1, 2, \ldots$ **do**
  Representation Stage:
  Map the policy network $\pi_{\theta_{k-1}}$ using representation network $f_{\phi_{k-1}}(\theta_{k-1})$.
  Decision Set Stage:
  $D_k \leftarrow \texttt{ConstructDecisonSet}(\theta_{k-1}, \mathcal{H}_{k-1})$.
  Bandit Stage:
  Use linear bandit algorithm to choose π_{θ_k} out of D_k.
  Exploitation Stage:
  Rollout policy π_{θ_k} and store the return G_k in $\mathcal{H}_k$.

  Update representation f_{ϕ_k}.
  Update bandit parameters $\hat{w}_t, V_t$ (Equation (2)) with the updated representation.
end for

Pseudo code for RepRL is presented in Algorithm 1. At every episode k, we map the policy's parameters θ_{k-1} to a latent space using $f_{\phi_{k-1}}(\theta_{k-1})$. We then use a construction algorithm, $\texttt{ConstructDecisonSet}(\theta_{k-1}, \mathcal{H}_{k-1})$, which takes into account the history $\mathcal{H}_{k-1}$, to generate a new decision set D_k. Then, to update the parameters θ_{k-1} of the policy, we select an optimistic policy $\pi_{\theta_k} \in D_k$ using a linear bandit method, such as TS or OFUL (see Section 2.1). Finally, we rollout the policy π_{θ_k} and update the representation network and the bandit parameters according to the procedure outlined in Equation (2), where x_k are the learned representations of f_{ϕ_k}. A visual schematic of our framework is depicted in Figure 1.

In the following sections, we present and discuss methods for representation learning, decision set construction, and propose two implementations of RepRL in the context of evolutionary strategies and policy gradient. We note that RepRL is a framework for addressing RL through representation, and as such, any representation learning technique or decision set algorithm can be incorporated as long as the basic structure is maintained.

3.2 Learning Representations for RepRL

We learn a linear representation of a policy using tools from variational inference. Specifically, we sample a representation from a posterior distribution $z \sim f_\phi(z|\theta)$, and train the representation by maximizing the Evidence Lower Bound (ELBO) [18] $\mathcal{L}(\phi, \kappa) = -\mathbb{E}_{z \sim f_\phi(z|\theta)}[\log p_\kappa(G|z)] + D_{KL}(f_\phi(z|\theta) \| p(z))$, where $f_\phi(z|\theta)$ acts as the encoder of the embedding, and $p_\kappa(G|z)$ is the return decoder or likelihood term.

The latent representation prior $p(z)$ is typically chosen to be a zero-mean Gaussian distribution. In order to encourage linearity of the value (i.e the return's mean) with respect to the learned representation, we chose the likelihood to be a Gaussian distribution with a mean that is linear in the representation, i.e., $p_\kappa(G|z) = \mathcal{N}(\kappa^\top z, \sigma^2)$. When the encoder is also chosen to be a Gaussian distribution, the loss function has a closed form. The choice of the decoder to be linear is crucial, due to the fact that the value is supposed to be linear w.r.t learned embeddings. The parameters ϕ and κ are the learned parameters of the encoder and decoder, respectively. Note that a determinis-

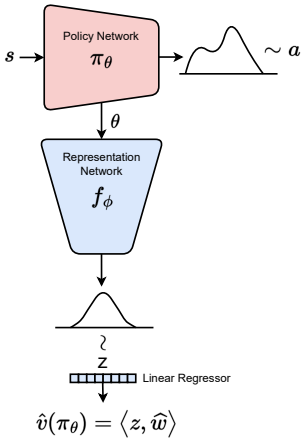

Figure 2: The diagram illustrates the structure of the networks in RepRL. The policy's parameters are fed into the representation network, which acts as a posterior distribution for the policy's latent representation. Sampling from this posterior, the latent representation is used by the bandits algorithm to evaluate the value that encapsulates the exploration-exploitation tradeoff.

tic mapping occurs when the function $f_\phi(z|\theta)$ takes the form of the Dirac delta function. A schematic of the architectural framework is presented in Figure 2.

3.3 Constructing a Decision Set

The choice of the decision set algorithm (line 4 of Algorithm 1) may have a great impact on the algorithm in terms of performance and computational complexity. Clearly, choosing $D_k = \Pi, \forall k$ will be unfeasible in terms of computational complexity. Moreover, it may be impractical to learn a linear representation for all policies at once. We present several possible choices of decision sets below.

(1) Policy Space Decision Set. One potential strategy is to sample a set of policies centered around the current policy: $D_k = \{\theta_k + \epsilon_i\}_{i=1}^N, \quad \epsilon_i \sim \mathcal{N}(0, \nu^2 I)$, where $\nu > 0$ controls how local policy search is. This approach is motivated by the assumption that the representation of policies in the vicinity of the current policy will exhibit linear behavior with respect to the value function due to their similarity to policies encountered by the learner thus far. **(2) Latent Space Decision Set.** An alternative approach involves sampling policies in their learned latent space, i.e., $D_k = \{z_k + \epsilon_i\}_{i=1}^N, \quad \epsilon_i \sim \mathcal{N}(0, \nu^2 I)$, where $z_k \sim f_\phi(z|\theta_k)$. The linearity of the latent space ensures that this decision set will improve the linear bandit target (UCB or the sampled value in TS), which will subsequently lead to an improvement in the actual value. This approach enables optimal exploration w.r.t. linear bandits, as it uniformly samples the eigen directions of the precision matrix V_t, rather than only sampling specific directions as may occur when sampling in the parameter space.

Unlike Section 3.3 constructing the set in Section 3.3 presents several challenges. First, in order to rollout the policy π_{θ_k}, one must construct an inverse mapping to extract the chosen policy from the selected latent representation. This can be done by training a decoder for the policy parameters $q(\theta|z)$. Alternatively, we propose to use a decoder-free approach. Given a target embedding $z^* \in \arg\max_{z \in D_t} \langle z, \hat{w} \rangle$, we search for a policy $\theta^* \in \arg\max_\theta f_\phi(z^*|\theta)$. This optimization problem can be solved using gradient descent-based optimization algorithms by varying the inputs to f_ϕ. A second challenge for latent-based decision sets involves the realizability of such policies. That is, there may exist representations $z \in D_k$, which are not mapped by any policy in Π. Lastly, even for realizable policies, the restored θ may be too far from the learned data manifold, leading to an overestimation of its value and a degradation of the overall optimization process. One way to address these issues is to use a small enough value of ν during the sampling process, reducing the probability of the set members being outside the data distribution. We leave more sophisticated methods of latent-based decision sets for future work. **(3) History-based Decision Set.** An additional approach uses the history of policies at time k to design a decision set. Specifically, at time episode k we sample around the set of policies observed so far, i.e., $D_k = \bigcup_{\ell \in [k]} \{\theta_\ell + \epsilon_{\ell,i}\}_{i=1}^N, \quad \epsilon_{\ell,i} \sim \mathcal{N}(0, \nu^2 I)$, resulting in a decision set of size Nk. After improving the representation over time, it may be possible to find a better policy near policies that have already been used and were missed due to poor representation or sampling mismatch. This method is quite general, as the history can be truncated only to consider a certain number of past time steps, rather than the complete set of policies observed so far. Truncating the history can help reduce the size of the decision set, making the search more computationally tractable.

In Section 5, we compare the various choices of decision sets. Nevertheless, we found that using policy space decisions is a good first choice, due to their simplicity, which leads to stable implementations. Further exploration of other decision sets is left as a topic for future research.

3.4 Inner trajectory sampling

Vanilla RepRL uses the return values of the entire trajectory. As a result, sampling the trajectories at their initial states is the natural solution for both the bandit update and representation learning. However, the discount factor diminishes learning signals beyond the $\frac{1}{1-\gamma}$ effective horizon, preventing the algorithm from utilizing these signals, which may be critical in environments with long-term dependencies. On the other hand, using a discount factor $\gamma = 1$ would result in returns with a large variance, leading to poor learning. Instead of sampling from the initial state, we propose to use the discount factor and sample trajectories at various states during learning, enabling the learner to observe data from different locations along the trajectory. Under this sampling scheme, the estimated

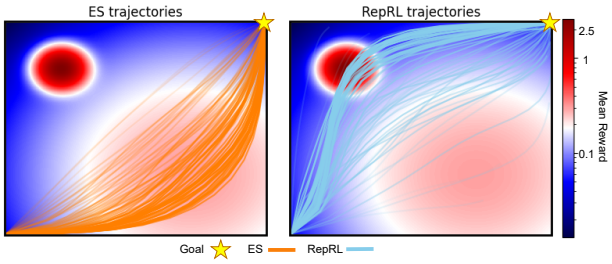

Figure 3: GridWorld visualization experiment. Trajectories were averaged across 100 seeds at various times during training, where more recent trajectories have greater opacity. Background colors indicate the level of mean reward.

value would be an estimate of the following quantity: $\tilde{v}(\pi) = \mathbb{E}_{s\sim\rho^\pi}[v(\pi, s)]$. In the following proposition we prove that optimizing $\tilde{v}(\pi)$ is equivalent to optimizing the real value.

Proposition 3.2. *For a policy* $\pi \in \Pi$, $\tilde{v}(\pi) = \frac{v(\pi)}{1-\gamma}$.

The proof can be found in the Appendix C. That is, sampling along the trajectory from ρ^π approximates the scaled value, which, like $v(\pi)$, exhibits linear behavior with respect to the reward function. Thus, instead of sampling the return defined in Equation (1), we sample $\tilde{G}(\pi) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$, where $s_0 \sim \rho^\pi, a_t \sim \pi(s_t), s_{t+1} \sim T(s_t, a_t)$, both during representation learning and bandit updates. Empirical evidence suggests that uniformly sampling from the stored trajectory produces satisfactory results in practice.

4 RepRL Algorithms

In this section we describe two possible approaches for applying the RepRL framework; namely, in Evolution Strategy [45] and Policy Gradients [38].

4.1 Representation Driven Evolution Strategy

Evolutionary Strategies (ES) are used to train agents by searching through the parameter space of their policy and sampling their return. In contrast to traditional gradient-based methods, ES uses a population of candidates evolving over time through genetic operators to find the optimal parameters for the agent. Such methods have been shown to be effective in training deep RL agents in high-dimensional environments [31, 23].

At each round, the decision set is chosen over the policy space with Gaussian sampling around the current policy as described in Section 3.3. Algorithm 3 considers an ES implementation of RepRL. To improve the stability of the optimization process, we employ soft-weighted updates across the decision set. This type of update rule is similar to that used in ES algorithms [31, 23], and allows for an optimal exploration-exploitation trade-off, replacing the true sampled returns with the bandit's value. Moreover, instead of sampling the chosen policy, we evaluate it by also sampling around it as done in ES-based algorithms. Each evaluation is used for the bandit parameters update and representation learning process. Sampling the evaluated policies around the chosen policy helps the representation avoid overfitting to a specific policy and generalize better for unseen policies - an important property when selecting the next policy.

Unlike traditional ES, optimizing the UCB in the case of OFUL or sampling using TS can encourage the algorithm to explore unseen policies in the parameter space. This exploration is further stabilized by averaging over the sampled directions, rather than assigning the best policy in the decision set. This is particularly useful when the representation is still noisy, reducing the risk of instability caused by hard assignments. An alternative approach uses a subset of D_t with the highest bandit scores, as suggested in Mania et al. [23], which biases the numerical gradient towards the direction with the highest potential return.

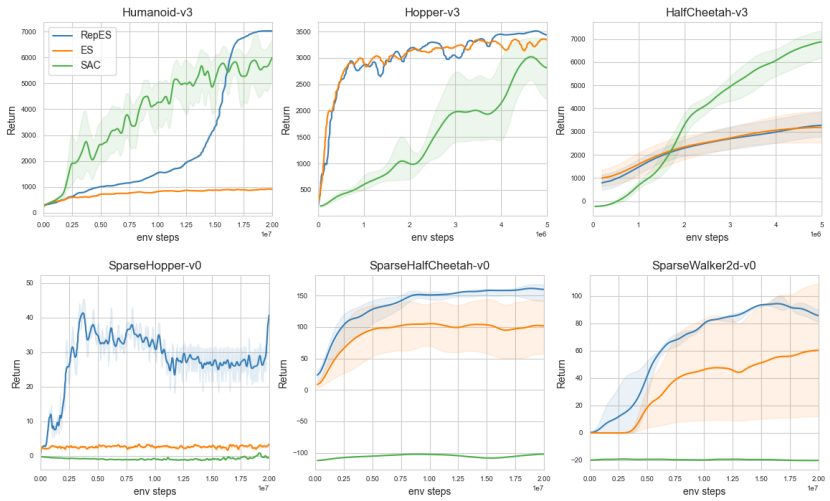

Figure 4: MuJoCo experiments during training. The results are for the MuJoCo suitcase (top) and the modified sparse MuJoCo (bottom).

4.2 Representation Driven Policy Gradient

RepRL can also be utilized as a regularizer for policy gradient algorithms. Pseudo code for using RepRL in policy gradients is shown in Algorithm 4. At each gradient step, a weighted regularization term $d(\theta, \tilde{\theta})$ is added, where $\tilde{\theta}$ are the parameters output by RepRL with respect to the current parameters for a chosen metric (e.g., ℓ_2): $\mathcal{L}_{\text{reg}}(\theta) = \mathcal{L}_{\text{PG}}(\theta) + \zeta d(\theta, \tilde{\theta})$.

After collecting data with the chosen policy and updating the representation and bandit parameters, the regularization term is added to the loss of the policy gradient at each gradient step. The policy gradient algorithm can be either on-policy or off-policy while in our work we experiment with an on-policy algorithm. Similar to the soft update rule in ES, using RepRL as a regularizer can significantly stabilize the representation process. Applying the regularization term biases the policy toward an optimal exploration strategy in policy space. This can be particularly useful when the representation is still weak and the optimization process is unstable, as it helps guide the update toward more promising areas of the parameter space. In our experiments, we found that using RepRL as a regularizer for policy gradients improved the stability and convergence of the optimization process.

5 Experiments

In order to evaluate the performance of RepRL, we conducted experiments on various tasks on the MuJoCo [43] and MinAtar [47] domains. We also used a sparse version of the MuJoCo environments, where exploration is crucial. We used linear TS as our linear bandits algorithm as it exhibited good performance during evaluation. The detailed network architecture and hyperparameters utilized in the experiments are provided in Appendix G. According to the decision set comparison, presented at Appendix F, RepRL demonstrated similar performance for the varying decision sets on the tested domains. In what follows, we focus on policy space decision sets.

Grid-World Visualization. Before presenting our results, we demonstrate the RepRL framework on a toy example. Specifically, we constructed a GridWorld environment (depicted in Figure 3) which consists of spatially changing, noisy rewards. The agent, initialized at the bottom left state $(x, y) = (1, 1)$, can choose to take one of four actions: up, down, left, or right. To focus on exploration, the rewards were distributed unevenly across the grid. Particularly, the reward for every (x, y) was defined by the Normal random variable $r(x, y) \sim \mathcal{N}\big(\mu(x, y), \sigma^2\big)$, where $\sigma > 0$ and $\mu(x, y) \propto R_1 \exp\Big\{-\frac{(x-x_1)^2+(y-y_1)^2}{a_1}\Big\} + R_2 \exp\Big\{-\frac{(x-x_2)^2+(y-y_2)^2}{a_2}\Big\} + R_3 \mathbb{1}_{\{(x,y)=\text{goal}\}}$. That is, the reward consisted of Normally distributed noise, with mean defined by two spatial Gaussians, as

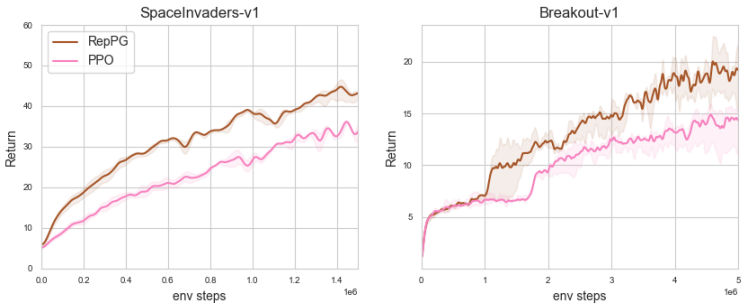

Figure 5: MinAtar experiments during training.

shown in Figure 3, with $R_1 > R_2$, $a_1 < a_2$ and a goal state (depicted as a star), with $R_3 \gg R_1, R_2$. Importantly, the values of R_1, R_2, R_3, a_1, a_2 were chosen such that an optimal policy would take the upper root in Figure 3.

Comparing the behavior of RepRL and ES on the GridWorld environment, we found that RepRL explored the environment more efficiently, locating the optimal path to the goal. This emphasizes the varying characteristics of state-space-driven exploration vs. policy-space-driven exploration, which, in our framework, coincides with representation-driven exploration. Figure 7 illustrates a two-dimensional t-SNE plot comparing the learned latent representation of the policy with the direct representation of the policy weights.

MuJoCo. We conducted experiments on the MuJoCo suitcase task using RepRL. Our approach followed the setting of Mania et al. [23], in which a linear policy was used and demonstrated excellent performance on MuJoCo tasks. We utilized the ES variant of our algorithm (Algorithm 3). We incorporated a weighted update between the gradients using the bandit value and the zero-order gradient of the sampled returns, taking advantage of sampled information and ensuring stable updates in areas where the representation is weak.

We first evaluated RepES on the standard MuJoCo baseline (see Figure 4). RepES either significantly outperformed or performed on-par with ES. We also tested a modified, sparse variant of MuJoCo. In the sparse environment, a reward was given for reaching a goal each distance interval, denoted as d, where the reward function was defined as:

$$r(s,a) = \begin{cases} 10 - c(a), & |x_{\text{agent}}| \bmod d = 0 \\ -c(a), & \text{o.w.} \end{cases}$$

Here, $c(a)$ is the control cost associated with utilizing action a, and x_{agent} denotes the location of the agent along the x-axis. The presence of a control cost function incentivized the agent to maintain its position rather than actively exploring the environment. The results of this experiment, as depicted in Figure 4, indicate that the RepRL algorithm outperformed both the ES and SAC algorithms in terms of achieving distant goals. However, it should be noted that the random search component of the ES algorithm occasionally resulted in successful goal attainment, albeit at a significantly lower rate in comparison to the RepRL algorithm.

MinAtar. We compared the performance of RepRL on MinAtar [47] with the widely used policy gradient algorithm PPO [33]. Specifically, we compared PPO against its regularized version with RepRL, as described in Algorithm 4, and refer to it as RepPG. We parametrized the policy by a neural network. Although PPO collects chunks of rollouts (i.e., uses subtrajectories), RepPG adjusted naturally due to the inner trajectory sampling (see Section 3.4). That is, the critic was used to estimate the value of the rest of the trajectory in cases where the rollouts were truncated by the algorithm.

Results are shown in Figure 5. Overall, RepRL outperforms PPO on all tasks, suggesting that RepRL is effective at solving challenging tasks with sparse rewards, such as those found in MinAtar.

6 Related Work

Policy Optimization: Policy gradient methods [38] have shown great success at various challenging tasks, with numerous improvements over the years; most notable are policy gradient methods for deterministic policies [35, 22], trust region based algorithms [32, 33], and maximum entropy algorithms [14]. Despite its popularity, traditional policy gradient methods are limited in continuous action spaces. Therefore, Tessler et al. [42] suggest optimizing the policy over the policy distribution space rather than the action space. In recent years, finite difference gradient methods have been rediscovered by the RL community. This class of algorithms uses numerical gradient estimation by sampling random directions [28]. A closely related family of optimization methods is Evolution Strategies (ES) a class of black-box optimization algorithms that heuristic search by perturbing and evaluating the set members, choosing only the mutations with the highest scores until convergence. Salimans et al. [31] used ES for RL as a zero-order gradient estimator for the policy, parameterized as a neural network. ES is robust to the choice of the reward function or the horizon length and it also does not need value function approximation as most state-of-art algorithms. Nevertheless, it suffers from low sample efficiency due to the potentially noisy returns and the usage of the final return value as the sole learning signal. Moreover, it is not effective in hard exploration tasks. Mania et al. [23] improves ES by using only the most promising directions for gradient estimation.

Policy Search with Bandits. Fox & Rolph [13] was one of the first works to utilize multi-arm bandits for policy search over a countable stationary policy set – a core approach for follow-up work [7, 3]. Nevertheless, the concept was left aside due to its difficulty to scale up with large environments. As an alternative, Neural linear bandits [30, 46, 25] simultaneously train a neural network policy, while interacting with the environment, using a chosen linear bandit method and are closely related to the neural-bandits literature [48, 16]. In contrast to this line of work, our work maps entire policy functions into linear space, where linear bandit approaches can take effect. This induces an exploration strategy in policy space, as opposed to locally, in action space.

Representation Learning. Learning a compact and useful representation of states [19, 34, 40], actions [41, 8], rewards [5, 26, 44], and policies [15, 11], has been at the core of a vast array of research. Such representations can be used to improve agents' performance by utilizing the structure of an environment more efficiently. Tang et al. [39] utilizes policy representation to learn a generalized value function. They demonstrate that the generalized value function can generalize across policies and improve value estimation for actor-critic algorithms. In another study, Li et al. [21] enhance the stability and efficiency of Evolutionary Reinforcement Learning (ERL) [17] by adopting a linear policy representation with a shared state representation. In our research, we view the representation problem as an alternative solution to the exploration-exploitation problem in RL. Although this shift does not necessarily simplify the problem, it transfers the challenge to a different domain, offering opportunities for the development of new methods.

7 Discussion and Future Work

We presented RepRL, a novel representation-driven framework for reinforcement learning. By optimizing the policy over a learned representation, we leveraged techniques from the contextual bandit literature to guide exploration and exploitation. We demonstrated the effectiveness of this framework through its application to evolutionary and policy gradient-based approaches, leading to significantly improved performance compared to traditional methods. In this work, we suggested reframing the exploration-exploitation problem as a representation-exploitation problem. By embedding the policy network into a linear feature space, good policy representations enable optimal exploration. This framework provides a new perspective on reinforcement learning, highlighting the importance of policy representation in determining optimal exploration-exploitation strategies.

As future work, one can incorporate RepRL into more involved representation methods, including pretrained large Transformers [10, 6], which have shown great promise recently in various areas of machine learning. Another avenue for future research is the use of RepRL in scenarios where the policy is optimized in latent space using an inverse mapping (i.e., decoder), as well as more involved decision sets. Finally, while this work focused on linear bandit algorithms, future work may explore the use of general contextual bandit algorithms, (e.g., SquareCB Foster & Rakhlin [12]), which are not restricted to linear representations.

References

[1] Yasin Abbasi-Yadkori, David Pal, and Csaba Szepesvari. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pp. 2312–2320, 2011.

[2] Marc Abeille and Alessandro Lazaric. Linear thompson sampling revisited. In *Artificial Intelligence and Statistics*, pp. 176–184. PMLR, 2017.

[3] Rajeev Agrawal, Demosthenis Teneketzis, and Venkatachalam Anantharam. Asymptotically efficient adaptive allocation schemes for controlled markov chains: Finite parameter space. Technical report, MICHIGAN UNIV ANN ARBOR COMMUNICATIONS AND SIGNAL PROCESSING LAB, 1988.

[4] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pp. 127–135, 2013.

[5] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Apostolos N Burnetas and Michael N Katehakis. Optimal adaptive policies for markov decision processes. *Mathematics of Operations Research*, 22(1):222–255, 1997.

[8] Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip Thomas. Learning action representations for reinforcement learning. In *International conference on machine learning*, pp. 941–950. PMLR, 2019.

[9] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 208–214. JMLR Workshop and Conference Proceedings, 2011.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

[12] Dylan Foster and Alexander Rakhlin. Beyond ucb: Optimal and efficient contextual bandits with regression oracles. In *International Conference on Machine Learning*, pp. 3199–3210. PMLR, 2020.

[13] Bennett L Fox and John E Rolph. Adaptive policies for markov renewal programs. *The Annals of Statistics*, 1(2):334–341, 1973.

[14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International conference on machine learning*, pp. 1861–1870, 2018.

[15] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

[16] Parnian Kassraie and Andreas Krause. Neural contextual bandits without regret. In *International Conference on Artificial Intelligence and Statistics*, pp. 240–278. PMLR, 2022.

[17] Shauharda Khadka and Kagan Tumer. Evolution-guided policy gradient in reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[19] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650. PMLR, 2020.

[20] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 661–670, 2010.

[21] Pengyi Li, Hongyao Tang, Jianye Hao, Yan Zheng, Xian Fu, and Zhaopeng Meng. Erl-re: Efficient evolutionary reinforcement learning with shared state representation and individual policy representation. *arXiv preprint arXiv:2210.17375*, 2022.

[22] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[23] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[25] Ofir Nabati, Tom Zahavy, and Shie Mannor. Online limited memory neural-linear bandits with likelihood matching. *arXiv preprint arXiv:2102.03799*, 2021.

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

[27] Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant architectures for learning in deep weight spaces. *arXiv preprint arXiv:2301.12780*, 2023.

[28] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

[29] Athanasios S Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

[30] Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018.

[31] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[32] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. *International conference on machine learning*, pp. 1889–1897, 2015.

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

[34] Erez Schwartz, Guy Tennenholtz, Chen Tessler, and Shie Mannor. Language is power: Representing states using natural language in reinforcement learning. *arXiv preprint arXiv:1910.02789*, 2019.

[35] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. *International conference on machine learning*, pp. 387–395, 2014.

[36] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[37] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[38] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[39] Hongyao Tang, Zhaopeng Meng, Jianye Hao, Chen Chen, Daniel Graves, Dong Li, Changmin Yu, Hangyu Mao, Wulong Liu, Yaodong Yang, et al. What about inputting policy in value function: Policy representation and policy-extended value function approximator. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8441–8449, 2022.

[40] Guy Tennenholtz and Shie Mannor. Uncertainty estimation using riemannian model dynamics for offline reinforcement learning. In *Advances in Neural Information Processing Systems*.

[41] Guy Tennenholtz and Shie Mannor. The natural language of actions. In *International Conference on Machine Learning*, pp. 6196–6205. PMLR, 2019.

[42] Chen Tessler, Guy Tennenholtz, and Shie Mannor. Distributional policy optimization: An alternative approach for continuous control. *Advances in Neural Information Processing Systems*, 32, 2019.

[43] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

[44] Rodrigo Toro Icarte, Ethan Waldie, Toryn Klassen, Rick Valenzano, Margarita Castro, and Sheila McIlraith. Learning reward machines for partially observable reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

[45] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

[46] Pan Xu, Zheng Wen, Handong Zhao, and Quanquan Gu. Neural contextual bandits with deep representation and shallow exploration. *arXiv preprint arXiv:2012.01780*, 2020.

[47] Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

[48] Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, pp. 11492–11502. PMLR, 2020.

A Algorithms

Algorithm 2 Random Search / Evolution Strategy

Input: initial policy $\pi_0 = \pi_{\theta_0}$, noise ν, step size α, set size K.
for $t = 1, 2, \ldots, T$ **do**
  Sample a decision set $D_t = \{\theta_{t-1} \pm \delta_i\}_{i=1}^K$, $\delta_i \sim \mathcal{N}(0, \nu^2 I)$.
  Collect the returns $\{G(\theta_{t-1} \pm \delta_i)\}_{i=1}^K$ of each policy in D_t.
  Update policy

$$\theta_t = \theta_{t-1} + \frac{\alpha}{\sigma_R K} \sum_{i=1}^{K} \Big[G(\theta_{t-1} + \delta_i) - G(\theta_{t-1} - \delta_i) \Big] \delta_i$$

end for

Algorithm 3 Representation Driven Evolution Strategy

Input: initial policy $\pi_0 = \pi_{\theta_0}$, noise ν, step size α, set size K, decision set size N, history $\mathcal{H}$.
for $t = 1, 2, \ldots, T$ **do**
  Sample an evaluation set $\{\theta_{t-1} \pm \delta_i\}_{i=1}^K$, $\delta_i \sim \mathcal{N}(0, \nu^2 I)$.
  Collect the returns $\{G(\theta_{t-1} \pm \delta_i)\}_{i=1}^K$ from the environment and store them in replay buffer.
  Update representation f_t and bandit parameters $(\hat{w}_t, V_t)$ using history.
  Construct a decision set $D_t = \{\theta_{t-1} \pm \delta_i\}_{i=1}^N$, $\delta_i \sim \mathcal{N}(0, \nu^2 I)$.
  Use linear bandit algorithm to evaluate each policy in D_t: $\{\hat{v}(\theta_{t-1} \pm \delta_i)\}_{i=1}^N$.
  Update policy

$$\begin{aligned} g_t =& \frac{1}{N} \sum_{i=1}^{N} \Big[\hat{v}(\theta_{t-1} + \delta_i) - \hat{v}(\theta_{t-1} - \delta_i) \Big] \delta_i, \\ \theta_t =& \theta_{t-1} + \alpha g_t \end{aligned}$$

end for

Algorithm 4 Representation Driven Policy Gradient

Input: initial policy π_θ, noise ν, step size α, decision set size N, ζ, history $\mathcal{H}$.
for $t = 1, 2, \ldots, T$ **do**
  for $1, 2, \ldots, K$ **do**
    Collect trajectory data using π_θ.
    Update representation f and bandit parameters $(\hat{w}, \Sigma)$ using history.
  end for
  for $1, 2, \ldots, M$ **do**
    Sample a decision set $D = \{\theta + \delta_i\}_{i=1}^N$, $\delta_i \sim \mathcal{N}(0, \nu^2 I)$.
    Use linear bandit algorithm to choose the best parameter $\tilde{\theta} \in \arg\max_{\theta \in D} \langle z, \hat{w} \rangle$ for $z \sim f(z|\theta)$.
    Compute

$$\begin{aligned} g =& \nabla_\theta \big[\mathcal{L}_{PG}(\theta) + \zeta \|\theta - \tilde{\theta}\|_2 \big], \\ \theta =& \theta - \alpha g \end{aligned}$$

  end for
end for

B Variational Interface

We present here proof of the ELBO loss for our variational interface, which was used to train the representation encoder.

Proof.

$$
\begin{aligned}
\log p(G;\phi,\kappa) &= \log \int_z p_\kappa(G|z)p(z)dz \\
&= \log \int_z p_\kappa(G|z)\frac{p(z)}{f_\phi(z|\pi)} f_\phi(z|\pi)dz \\
&= \log \mathbb{E}_{z\sim f_\phi(z|\pi)}\left[p_\kappa(G|z)\frac{p(z)}{f_\phi(z|\pi)}\right] \\
&\geq \mathbb{E}_{z\sim f_\phi(z|\pi)}\left[\log p_\kappa(G|z)\right] + \mathbb{E}_{z\sim f_\phi(z|\pi)}\left[\log \frac{p(z)}{f_\phi(z|\pi)}\right] \\
&= \mathbb{E}_{z\sim f_\phi(z|\pi)}\left[\log p_\kappa(G|z)\right] - D_{KL}(f_\phi(z|\pi)\|p(z)),
\end{aligned}
$$

where the inequality is due to Jensen's inequality.

C Proof for Proposition 3.2

By definition:

$$
\begin{aligned}
\tilde{v}(\pi) &= \sum_s \rho^\pi(s)v(\pi,s) \\
&= \sum_s \rho^\pi(s)\sum_a \pi(a|s)\left\{r(s,a) + \gamma\sum_{s'} T(s'|s,a)v(\pi,s')\right\} \\
&= v(\pi) + \gamma\sum_s \rho^\pi(s)\sum_a \pi(a|s)\sum_{s'} T(s'|s,a)v(\pi,s') \\
&= v(\pi) + \gamma\sum_{s'} \rho^\pi(s')v(\pi,s') \\
&= v(\pi) + \gamma\tilde{v}(\pi),
\end{aligned}
$$

where the second equality is due to the Bellman equation and the third is from the definition. Therefore,

$$
\tilde{v}(\pi) = v(\pi) + \gamma\tilde{v}(\pi) \Longrightarrow \tilde{v}(\pi) = \frac{v(\pi)}{1-\gamma}
$$

□

D Full RepRL Scheme

The diagram presented below illustrates the networks employed in RepRL. The policy's parameters are inputted into the representation network, which serves as a posterior distribution capturing the latent representation of the policy. Subsequently, a sampling procedure is performed from the representation posterior, followed by the utilization of a linear return encoder, acting as the likelihood, to forecast the return distribution with a linear mean (i.e. the policy's value). This framework is employed to maximize the Evidence Lower Bound (ELBO).

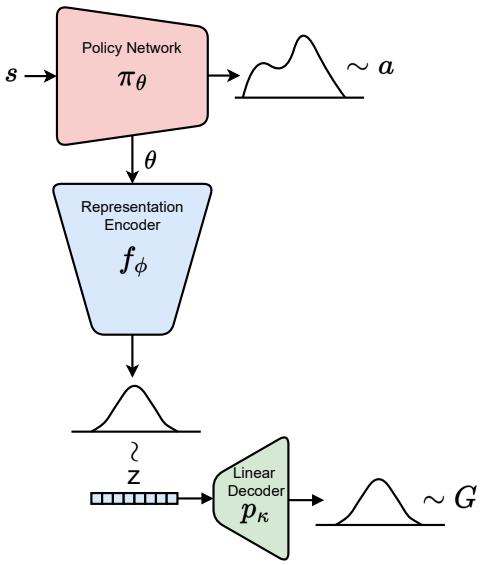

Figure 6: The full diagram illustrates the networks in RepRL.

E GridWorld t-SNE

Below we present the two-dimensional t-SNE visualization depicts the policy representation in the GridWorld experiment. On the right, we observe the learned latent representation, while on the left, we see the direct representation of the policy's weights. Each point in the visualization corresponds to a distinct policy, and the color of each point corresponds to a sample of the policy's value.

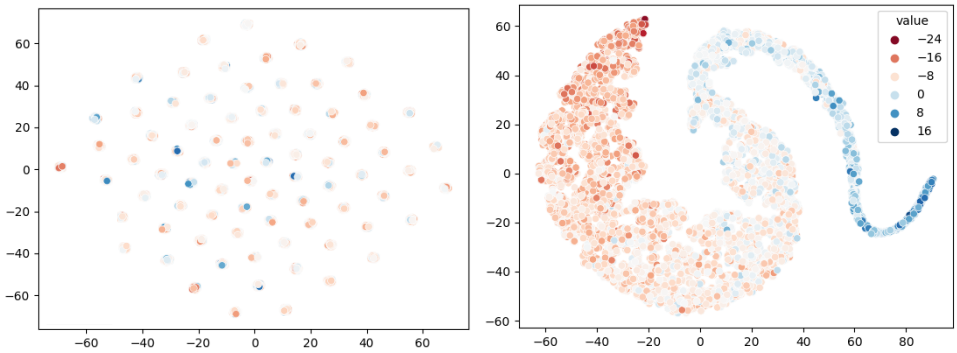

Figure 7: The two-dimensional t-SNE visualization during the GridWorld experiment.

F Decision Set Experiment

The impact of different decision sets on the performance of RepRL was assessed in our evaluation. We conducted tests using three specific decision sets as described in Section 3.3. The evaluation was carried out on a sparse HalfCheetah environment, utilizing the RepES variant. When evaluating the history-based decision set, we considered a history window consisting of 20 policies. In the latent-based setting, the parameters corresponding to the selected latent code were obtained using a gradient descent algorithm. The results showed that RepRL exhibited similar performance across the various decision sets tested in different domains.

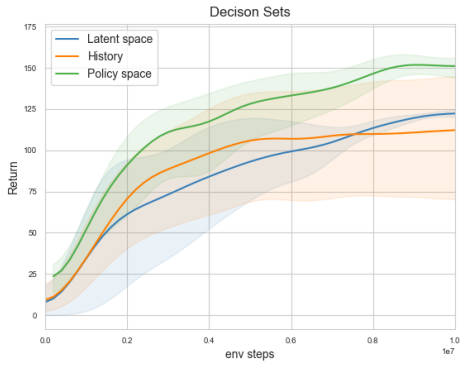

Figure 8: Plots depict experiments for three decision sets: policy space-based, latent space-based, and history-based. The experiment was conducted on the SparseHalfCheetah environment.

G Hyperparameters and Network Architecture

G.1 Grid-World

In the GridWorld environment, a 8×8 grid is utilized with a horizon of 20, where the reward is determined by a stochastic function as outlined in the paper: $r(x, y) \sim \mathcal{N}\big(\mu(x, y), \sigma^2\big)$, where $\sigma > 0$ and $\mu(x, y) \propto R_1 \exp\Big\{-\frac{(x-x_1)^2+(y-y_1)^2}{a_1}\Big\} + R_2 \exp\Big\{-\frac{(x-x_2)^2+(y-y_2)^2}{a_2}\Big\} + R_3 \mathbb{1}_{\{(x,y)=\text{goal}\}}$. The parameters of the environment are set as $R_1 = 2.5, R_2 = 0.3, R_3 = 13, \sigma = 3, a_1 = 0.125, a_2 = 8$.

The policy employed in this study is a fully-connected network with 3 layers, featuring the use of the $tanh$ non-linearity operator. The hidden layers' dimensions across the network are fixed at 32, followed by a $Softmax$ operation. The state is represented as a one-hot vector. please rephrase the next paragraph so it will sounds more professional: The representation encoder is built from Deep Weight-Space (DWS) layers [27], which are equivariant to the permutation symmetry of fully connected networks and enable much stronger representation capacity of deep neural networks compared to standard architectures. The DWS model (DWSNet) comprises four layers with a hidden dimension of 16. Batch normalization is applied between these layers, and a subsequent fully connected layer follows. Notably, the encoder is deterministic, meaning it represents a delta function. For more details, we refer the reader to the code provided in Navon et al. [27], which was used by us.

In the experimental phase, 300 rounds were executed, with 100 trajectories sampled at each round utilizing noisy sampling of the current policy, with a zero-mean Gaussian noise and a standard deviation of 0.1. The ES algorithm utilized a step size of 0.1, while the RepRL algorithm employed a decision set of size 2048 without a discount factor ($\gamma = 1$) and $\lambda = 0.1$.

G.2 MuJoCo

In the MuJoCo experiments, both ES and RepES employed a linear policy, in accordance with the recommendations outlined in [23]. For each environment, ES utilized the parameters specified by Mania et al. [23], while RepES employed the same sampling strategy in order to ensure a fair comparison.

RepES utilized a representation encoder consisting of 4 layers of a fully-connected network, with dimensions of 2048 across all layers, and utilizing the $ReLU$ non-linearity operator. This was followed by a fully-connected layer for the mean and variance. The latent dimension was also chosen to be 2048. After each sampling round, the representation framework (encoder and decoder) were trained for 3 iterations on each example, utilizing an $Adam$ optimizer and a learning rate of $3e-4$. When combining learning signals of the ES with RepES, a mixture gradient approach was employed, with 20% of the gradient taken from the ES gradient and 80% taken from the RepES gradient. Across all experiments, a discount factor of $\gamma = 0.995$ and $\lambda = 0.1$ were used.

G.3 MinAtar

In the MinAtar experiments, we employed a policy model consisting of a fully-connected neural network similar to the one utilized in the GridWorld experiment, featuring a hidden dimension of 64. The value function was also of a similar structure, with a scalar output. The algorithms collected five rollout chunks of 512 between each training phase.

The regulation coefficient chosen for RepRL was 1, while the discount factor and the mixing factor were set as $\gamma = 0.995$ and $\lambda = 0.1$. The representation encoder used was similar to the one employed in the GridWorld experiments with two layers, followed by a symmetry invariant layer and two fully connected layers.