Efficient Exploration via Epistemic-Risk-Seeking Policy Optimization

Brendan O'Donoghue [1]

Abstract

Exploration remains a key challenge in deep reinforcement learning (RL). Optimism in the face of uncertainty is a well-known heuristic with theoretical guarantees in the tabular setting, but how best to translate the principle to deep reinforcement learning, which involves online stochastic gradients and deep network function approximators, is not fully understood. In this paper we propose a new, differentiable optimistic objective that when optimized yields a policy that provably explores efficiently, with guarantees even under function approximation. Our new objective is a zero-sum two-player game derived from endowing the agent with an epistemic-risk-seeking utility function, which converts uncertainty into value and encourages the agent to explore uncertain states. We show that the solution to this game minimizes an upper bound on the regret, with the 'players' each attempting to minimize one component of a particular regret decomposition. We derive a new model-free algorithm which we call 'epistemic-risk-seeking actor-critic' (ERSAC), which is simply an application of simultaneous stochastic gradient ascent-descent to the game. Finally, we discuss a recipe for incorporating off-policy data and show that combining the risk-seeking objective with replay data yields a double benefit in terms of statistical efficiency. We conclude with some results showing good performance of a deep RL agent using the technique on the challenging 'DeepSea' environment, showing significant performance improvements even over other efficient exploration techniques, as well as improved performance on the Atari benchmark.

[1]Google DeepMind, London. Correspondence to: Brendan O'Donoghue <bodonoghue85@gmail.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

1. Introduction

Reinforcement learning (RL) involves an agent interacting with an environment over time attempting to maximize its total return (Sutton & Barto, 1998; Puterman, 2014; Meyn, 2022). Initially the agent does not know about the environment and must learn about it from experience. As the agent navigates the environment it receives noisy observations which it can use to update its (posterior) beliefs about the environment (Ghavamzadeh et al., 2015). Therefore, the RL problem is a *statistical inference problem* wrapped in a *control problem*, and the two problems must be tackled simultaneously for good data efficiency (Lu et al., 2021). This is because the policy of the agent affects the data it will collect, which in turn affects the policy, and so on. This is in contrast to supervised learning, where the performance of a classifier (for instance) does not influence the data it will later observe. Failure to properly consider the statistical aspect of the RL problem will result in agents that require exponential amounts of experience for good performance. So far, deep RL as a field has largely accepted this tradeoff, requiring enormous computational budgets to solve relatively simple problems. On the other hand, correctly considering the statistical inference problem and the control problem together has the potential to dramatically reduce the compute requirements to solve problems and potentially unlock new domains and capabilities far outside of the range of current agents.

Understood in this way, RL is about choosing what actions to take, and consequently which data to collect, in order to maximize long-term return. To do this an agent must sometimes take actions that lead to states where it has epistemic uncertainty about the value of those states, and sometimes take actions that lead to more certain payoff. The tension between these two modes is the 'explore-exploit' dilemma (Auer, 2002; Kearns & Singh, 2002; Dimitrakakis & Ortner, 2018). When it comes to exploration in *deep* RL there are two main focus areas of research. The primary line of work is generating better *estimates of uncertainty*, typically by exploiting some aspect of a neural network (Singh et al., 2004; Barto, 2013; Stadie et al., 2015; Bellemare et al., 2016; Ostrovski et al., 2017; Burda et al., 2018; Pathak et al., 2017). Getting accurate uncertainty estimates from deep neural networks is a 'holy grail' of research in deep learning in general (Osband et al., 2021),

and in reinforcement learning good uncertainty estimates are crucial for good performance of any practical exploration algorithm. The second area of research is *how best to use uncertainty estimates for efficient exploration*, which is the focus of this work and as such any of the referenced methods for generating uncertainty estimates are compatible with the approach discussed herein. A lot of prior work in this area has simply converted the uncertainty estimates into an 'optimism in the face of uncertainty' bonus added to the rewards and then applied off-the-shelf RL algorithms to the modified Markov decision process (MDP) (Dayan & Sejnowski, 1996; Strehl & Littman, 2008; Bellemare et al., 2016; Tang et al., 2017). This approach is inspired by theoretical results based on optimism bonuses which show that in an episodic tabular MDP setting where the modified MDP is solved exactly, these strategies can yield good regret bounds (Auer et al., 2008; Jaksch et al., 2010; Azar et al., 2017; Jin et al., 2018). However, translating the performance to deep RL has been challenging. Consider the fact that some of the most impressive results in modern deep RL have had no sophisticated exploration strategies, relying instead on simple local dithering strategies (Mnih et al., 2015; Silver et al., 2016; Berner et al., 2019) or making extensive use of human data (Vinyals et al., 2019).

Although optimism is the most popular exploration technique in deep RL, there are several alternative approaches. One line of research is not to consider uncertainty explicitly, but instead to add some structured noise to dithering, such as Lévy flights (Dabney et al., 2020), or adding noise to the weights of the neural network (Fortunato et al., 2017; Plappert et al., 2017). These approaches have shown some promising results although they fall strictly into the category of heuristic and do not achieve good performance on challenging unit-test exploration domains like DeepSea (Osband et al., 2019). Another line of research involves Thompson sampling and various approximations to it (Thompson, 1933; Strens, 2000; Osband et al., 2013; Russo et al., 2018; Osband et al., 2016). Although Thompson sampling has excellent performance in tabular settings it is not yet clear how to translate that performance into deep RL settings reliably as a full implementation of Thompson sampling requires sampling from the posterior over policies, which is intractable for all but the simplest tabular domains. Another drawback of Thompson sampling is that it cannot handle either the multi-agent case nor the constrained case (O'Donoghue et al., 2020; O'Donoghue & Lattimore, 2021). Since we expect real-world agents to be in situations with multiple agents and to be bound by constraints this is a major disadvantage.

In this paper we endow a policy-gradient agent with an epistemic-risk-seeking utility function which summarizes both the expected return and the epistemic uncertainty into a single value (O'Donoghue, 2021; Eriksson & Dimitrakakis, 2019). How risk-seeking the agent is is controlled by a single scalar parameter which is tuned (*i.e.*, learned) to balance exploration and exploitation. The approach is based on a *dual* view of the recent 'K-learning' algorithm, which is a value learning, model-based, Bayesian RL approach with a guaranteed Bayesian regret bound in tabular domains (O'Donoghue, 2021). We derive a model-free and policy-based algorithm, which allows us to approximately solve for the optimal policy using stochastic feedback and online experience using policy gradients, and to use a deep neural network to parameterize our policy. Moreover, we can show that the approach enjoys Bayesian regret guarantees even in the face of function approximation. The final algorithm we present is an extension of policy gradient (Sutton et al., 1999; Konda & Tsitsiklis, 2003) with entropy regularization. Combining policy gradient with entropy regularization is a common heuristic and typically a small amount of entropy 'bonus' is used to discourage the policy from becoming deterministic and thereby losing the ability to 'explore' (Mnih et al., 2016). That being said, simply adding entropy regularization is not sufficient for deep exploration since entropy regularization only encourages local dithering (Osband, 2016; O'Donoghue et al., 2018). In this work we show that entropy regularization combined with a carefully tuned uncertainty bonus is a principled approach to deep exploration. Our approach formulates the problem as a two-player game where one player is attempting to find the policy that maximizes the optimistic reward and the other player is tuning how risk-seeking the policy is in order to minimize expected regret. The solution of this game yields the optimal K-learning policy with the associated performance guarantees. Unlike standard optimism approaches the K-learning policy is stationary (*i.e.*, not dependent on the number of elapsed episodes other than through the posteriors) and stochastic, and it varies slowly as data is collected, which makes it more amenable to online approximation. Unlike Thompson sampling, K-learning does not require a full sample from the posterior at each episode and it can handle both the multi-agent and the constrained cases when suitably modified (O'Donoghue et al., 2020; O'Donoghue & Lattimore, 2021). In practice on a hard exploration unit-test our approach outperforms deep RL approximations to both Thompson sampling and optimism, as we shall show in the numerical experiments. Our results suggest that the approach in this manuscript may close some of the gap between theory and practice for efficient exploration in deep RL.

2. Preliminaries

We consider an RL problem where an agent interacts with an unknown environment over a number of episodes. We model the environment as a finite state-

action time-inhomogeneous MDP given by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, L, \{P_l\}_{l=1}^L, \{R_l\}_{l=1}^L, \rho)$, where L is the horizon length, $\mathcal{S} = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_L \cup \{s_{L+1}\}$ is the set of states including terminating state s_{L+1}, $\mathcal{A}$ is the set of possible actions, $P_l : \mathcal{S}_l \times \mathcal{A} \to \Delta(\mathcal{S}_{l+1})$ denotes the state transition kernel at layer l, $R_l : \mathcal{S}_l \times \mathcal{A} \to \Delta(\mathbb{R})$ is the reward function at layer l with mean reward $r_l \in \mathbb{R}^{|\mathcal{S}_l| \times |\mathcal{A}|}$, and $\rho \in \Delta(\mathcal{S}_1)$ is the initial state distribution. A policy $\pi \in \Delta(\mathcal{A})^{|\mathcal{S}|}$ is a distribution over actions for each state, and we shall denote the probability of action a in state s at timestep l as $\pi_l(s, a)$. The agent starts in some state $s_1 \in \mathcal{S}_1$ sampled according to ρ, then for each step in the episode $l = 1, \ldots, L$ the agent is in state s_l, takes action $a_l \sim \pi_l(s_l, \cdot)$, receives reward sampled from $R_l(s_l, a_l)$, and transitions to state $s_{l+1} \in \mathcal{S}_{l+1}$ according to $P_l(\cdot \mid s_l, a_l)$. The episode ends when the terminating state s_{L+1} is reached, the initial state is sampled again and another episode begins.

For a given policy π we define value functions for each $(s, a) \in \mathcal{S}_l \times \mathcal{A}$, $l = 1, \ldots, L$, as

$$Q_l^\pi(s, a) = r_l(s, a) + \sum_{s' \in \mathcal{S}_{l+1}} P_l(s' \mid s, a) V_{l+1}^\pi(s'),$$
$$V_l^\pi(s) = \sum_a \pi_l(s, a) Q_l^\pi(s, a),$$

where we define $V_{L+1}^\pi \equiv 0$. The optimal values are defined for $l = 1, \ldots, L$ as

$$Q_l^\star(s, a) = r_l(s, a) + \sum_{s' \in \mathcal{S}_{l+1}} P_l(s' \mid s, a) V_{l+1}^\star(s'),$$
$$V_l^\star(s) = \max_a Q_l^\star(s, a),$$

and we define $V_{L+1}^\star \equiv 0$. The policy that achieves the max is given by

$$\pi_l^\star(s, a) = \mathbf{1}(a = \operatorname{argmax}(Q_l^\star(s, a))), \quad l = 1, \ldots, L,$$

assuming the argmax is unique, otherwise any policy that has support only on the maximum entries of $Q^\star$ is optimal.

2.1. Regret

The regret of a policy is the expected shortfall between the performance of the policy and the optimal performance. In this paper we take a Bayesian approach, which is to say we assume the agent has access to prior information about the MDP, represented by a distribution ϕ, and we are interested in the expected regret with respect to this prior. Concretely, for a policy π we define the Bayesian regret for a single episode as

$$\mathcal{R}(\pi, \phi) = \mathbb{E}_{\mathcal{M} \sim \phi}(\mathbb{E}_{s \sim \rho}(V_1^{\mathcal{M}, \star}(s) - V_1^{\mathcal{M}, \pi}(s))).$$

For clarity we have made the dependence of the value functions on $\mathcal{M}$ explicit here, but we shall suppress the dependency in the notation hereafter. If algorithm Alg produces policy sequence $\pi^1, \pi^2, \ldots$ based on observed histories $\mathcal{F}_1, \mathcal{F}_2, \ldots$, where $\mathcal{F}_t$ is all the observed history of states, actions, and rewards before episode t then, due to the tower property of conditional expectation, the *cumulative Bayesian regret* of Alg over N episodes is given by

$$\mathcal{BR}(\text{Alg}, \phi) = \mathbb{E} \sum_{t=1}^N \mathcal{R}(\pi^t, \phi^t) \tag{1}$$

where $\phi^t = \phi(\cdot \mid \mathcal{F}_t)$. Loosely speaking, agents that have low regret explore efficiently and generate high reward. So minimizing the cumulative Bayesian regret is important for good performance.

3. K-Learning

For the value functions in §2 to be computable they require exact knowledge of the mean reward r and transition matrix P. However, in reinforcement learning these are initially unknown and must be learned about from experience. K-learning was derived by endowing the agent with a *risk-seeking* exponential utility function $u : \mathbb{R} \to \mathbb{R}$ which converts uncertainties to value, defined for any $\tau \geq 0$ as $u_\tau(x) = \tau(\exp(x/\tau) - 1)$. We can compute the *certainty-equivalent* value under this utility for any random variable $X : \Omega \to \mathbb{R}$ as $J_\tau = u_\tau^{-1}(\mathbb{E}u_\tau(X)) = \tau \log \mathbb{E} \exp(X/\tau)$, and from Jensen's inequality we have $J_\tau \geq \mathbb{E}X$ for all $\tau \geq 0$. For example, random variable $X \sim \mathcal{N}(\mu, \sigma^2)$ has certainty equivalent value under u_τ of $J_\tau = \mu + \sigma^2/2\tau$. Clearly greater uncertainty (or risk) σ increases this value, and $\tau \geq 0$ controls the tradeoff. In the context of reinforcement learning the uncertainty we are interested in is the *epistemic* uncertainty about the unknown parameters of the MDP, and the risk-seeking utility function can be used to summarize the beliefs about an unknown MDP into a risk-seeking value. As shown by O'Donoghue (2021) the risk-seeking values are computable by solving a Bellman equation. Concretely, given posterior information ϕ we define the 'risk-seeking' reward function for each $(s, a) \in \mathcal{S}_l \times \mathcal{A}$, $l = 1, \ldots, L$, as

$$r_{l,\tau}(s, a) = \bar{r}_l(s, a) + \sigma_l^2(s, a)/2\tau,$$

where $\bar{r} = \mathbb{E}_\phi r$ and $\sigma \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ is a measure of the uncertainty about the MDP under ϕ (see O'Donoghue (2021) for details on how σ should be chosen, for now we shall just assume it is given). Then for any policy π and constant $\tau > 0$ we define risk-seeking value functions for each $(s, a) \in \mathcal{S}_l \times \mathcal{A}$, $l = 1, \ldots, L$, as

$$K_{l,\tau}^\pi(s, a) = r_{l,\tau}(s, a) + \sum_{s' \in \mathcal{S}_{l+1}} \bar{P}_l(s' \mid s, a) J_{l+1,\tau}^\pi(s'),$$
$$J_{l,\tau}^\pi(s) = \sum_a \pi_l(s, a) K_{l,\tau}^\pi(s, a) + \tau H(\pi_l(s, \cdot)), \tag{2}$$

where $\bar{P} = \mathbb{E}_\phi P$ and H denotes the entropy (Cover & Thomas, 2012), and we define $J^\pi_{L+1,\tau} \equiv 0$. Similarly to the optimal Q-values, we can define optimal K-values for each $(s,a) \in \mathcal{S}_l \times \mathcal{A}$, $l = 1,\dots,L$, and any $\tau > 0$ as follows

$$K^\star_{l,\tau}(s,a) = r_{l,\tau}(s,a) + \sum_{s' \in \mathcal{S}_{l+1}} \bar{P}_l(s' \mid s,a) J^\star_{l+1,\tau}(s'_l),$$

where

$$\begin{aligned} J^\star_{l,\tau}(s) &= \max_{\pi_l \in \Delta(\mathcal{A})} \left(\sum_a \pi_l(s,a) K^\star_{l,\tau}(s,a) + \tau H(\pi_l(s,\cdot)) \right) \\ &= \tau \log \sum_{a \in \mathcal{A}} \exp(K^\star_{l,\tau}(s,a)/\tau), \end{aligned}$$

where again we define $J^\star_{L+1,\tau} \equiv 0$. The policy that achieves the max is given by the 'Boltzmann' policy over the K-values, that is, for each $(s,a) \in \mathcal{S}_l \times \mathcal{A}$, $l = 1,\dots,L$

$$\pi^\star_{l,\tau}(s,a) = \exp\left(\frac{K^\star_{l,\tau}(s,a) - J^\star_{l,\tau}(s)}{\tau} \right). \tag{3}$$

Observe that if the agent has no uncertainty (*i.e.*, $\sigma = 0$), then letting $\tau \to 0$ recovers the original Q and V formulations in §2. The risk-seeking Bellman equation captures both the expected value and the uncertainty, and both propagate through the MDP to other states and actions. It is the 'temperature' parameter τ that is controlling the trade-off between them. So far τ is a free-variable, in the sequel we shall show how to optimize it so as to minimize regret.

The main result of O'Donoghue (2021) is that following the policy in Eq. (3) guarantees a sublinear Bayesian regret bound for appropriate choices of σ and τ. In other words, the policy associated with the optimal K-values balances exploration and exploitation efficiently. However, finding the policy requires solving a Bellman equation for the optimal K-values and the analysis was restricted to tabular cases. This paper builds on that work in three main ways:

1. We present a new objective over policies, rather than values, which can be solved using policy gradients to obtain the policy in Eq. (3).
2. The algorithm we derive is entirely model-free, whereas the previous work was model-based.
3. We extend the analysis and experiments to cover non-tabular and function approximation cases.

All the quantities we presented in this section are functions of the current beliefs ϕ, however, for brevity we have suppressed this dependence in the notation.

4. Saddle-Point Problem

If we assume that the posterior over the reward and transition functions are layerwise-independent, then it is straightforward to show that for any $\tau \geq 0$ and for $l = 1,\dots,L$

$$K^\pi_{l,\tau}(s,a) \geq \mathbb{E}_\phi Q^\pi_l(s,a), \quad J^\pi_{l,\tau}(s) \geq \mathbb{E}_\phi V^\pi_l(s).$$

Furthermore, in (O'Donoghue, 2021) it was shown that under some additional assumptions the optimal values satisfy for $l = 1,\dots,L$

$$K^\star_{l,\tau}(s,a) \geq \mathbb{E}_\phi Q^\star_l(s,a), \quad J^\star_{l,\tau}(s) \geq \mathbb{E}_\phi V^\star_l(s)$$

for an appropriate choice of σ and any $\tau \geq 0$. This means that the K-values are *optimistic*, and following policy (3) is an instance of optimism in the face of uncertainty. For our purposes in this paper we shall assume the following bound holds.

Assumption 1. $\mathbb{E}_{s\sim\rho} J^\star_{1,\tau}(s) \geq \mathbb{E}_{s\sim\rho}\mathbb{E}_\phi V^\star_1(s),\ \forall\, \tau \geq 0.$

Under Assumption 1, finding the *tightest* bound in the family requires solving $\min_\tau \mathbb{E}_{s\sim\rho} J^\star_{1,\tau}(s)$, and since for any τ we have $\max_\pi \mathbb{E}_{s\sim\rho} J^\pi_{1,\tau}(s) = \mathbb{E}_{s\sim\rho} J^\star_{1,\tau}(s)$, we obtain the following saddle-point problem:

$$\max_{\pi\in\Pi} \min_{\tau\geq 0} \mathbb{E}_{s\sim\rho} J^\pi_{1,\tau}(s) \tag{4}$$

where $\Pi \subseteq \Delta(|\mathcal{A}|)^{|\mathcal{S}|}$ is some possibly restricted policy space. The solution to this saddle-point problem yields the tightest upper-bound on the expected value of the optimal value function $\mathbb{E}_{s\sim\rho}\mathbb{E}_\phi V^\star_1(s)$ under the posterior ϕ, and, as we shall show, it also minimizes a bound on the Bayesian regret. Implicit in the definition of the saddle-point problem is the assumption of strong duality, which we state next. This assumption holds, for instance, if Π is convex.

Assumption 2. *Strong duality holds for* (4)*,* i.e.*,*

$$\min_{\tau\geq 0} \max_{\pi\in\Pi} \mathbb{E}_{s\sim\rho} J^\pi_{1,\tau}(s) = \max_{\pi\in\Pi} \min_{\tau\geq 0} \mathbb{E}_{s\sim\rho} J^\pi_{1,\tau}(s).$$

The saddle-point problem (4) is our main problem of interest, and the rest of this manuscript is dedicated to solving it and interpreting the solutions.

4.1. The connection to Bayesian regret

In order to provide a connection between the saddle-point problem (4) and the Bayesian regret (1) let us define a few quantities of interest. First, we define a notion of *optimism* for a given π and $\tau \geq 0$

$$\mathrm{Optimism}(\pi,\tau) \coloneqq \mathbb{E}_{s\sim\rho}\left(J^\pi_{1,\tau}(s) - \mathbb{E}_\phi V^\pi_1(s) \right).$$

Since $J^\pi_{1,\tau}(s) \geq \mathbb{E}_\phi V^\pi_1(s)$ for all τ, the $\mathrm{Optimism}$ is measuring how much 'bonus' is derived from the risk-seeking exponential utility, relative to the (risk-neutral) expected

value. Let us also define a notion of 'distance' from a policy to the optimal optimistic policy as the expected KL-divergence between the policies under the stationary distribution generated by π (Cover & Thomas, 2012), that is,

$$\mathrm{Dist}(\pi, \tau) \coloneqq \sum_{l=1}^{L} \tau \mathbb{E}_{\pi} \mathrm{KL}(\pi_l(s, \cdot) \mid\mid \pi^{\star}_{l,\tau}(s, \cdot)).$$

It turns out that we can relate the KL-divergence and the suboptimality gap for any policy (O'Donoghue, 2022; Mei et al., 2020).

Lemma 1. *[Cor. 1 (O'Donoghue, 2022)] For any $\tau > 0$ and policy $\pi \in \Delta(\mathcal{A})^{|\mathcal{S}|}$ we have:*

$$\mathrm{Dist}(\pi, \tau) = \mathbb{E}_{s\sim\rho}(J^{\star}_{1,\tau}(s) - J^{\pi}_{1,\tau}(s)).$$

With this we are ready to present the following decomposition.

Lemma 2. *Under Assumption 1 we can bound the Bayesian regret in a single episode for any policy π as*

$$\mathcal{R}(\pi, \phi) \leq \mathrm{Dist}(\pi, \tau) + \mathrm{Optimism}(\pi, \tau), \quad \forall \tau \geq 0.$$

The proof is deferred to Appendix B. This lemma shows that we can decompose the Bayesian regret bound into two terms. One term is a distance from the policy to the optimal optimistic policy, and the other term relates to the amount of optimism in the policy. Next we show how the saddle-point problem we are solving (4) relates to this decomposition.

Theorem 1. *Assume 1 and 2, and let (π_*, τ_*) be a solution to the saddle-point problem* (4)*, then*

$$\mathcal{R}(\pi_*, \phi) \leq \min_{\pi\in\Pi} \mathrm{Dist}(\pi, \tau_*) + \min_{\tau} \mathrm{Optimism}(\pi_*, \tau).$$

We defer the proof to Appendix B. The above Theorem tells us that even though the 'players' are competing in a zero-sum game, they are in a sense cooperating to minimize the Bayesian regret of the resulting policy. The solutions to the saddle-point problem (4) are each minimizing one component that contributes to the Bayesian regret bound in the decomposition we derived in Lemma 2, and ignoring the other. In summary, we can interpret the saddle point problem as follows:

- The policy player π is maximizing the entropy-regularized optimistic reward, where the amount of optimism is controlled by τ. Equivalently, it is minimizing the expected KL-divergence to the optimal optimistic policy, and thereby minimizing one component contributing to the regret bound.
- The risk-seeking player τ is balancing the reward bonus and entropy regularization in order to minimize the upper bound on the value function under π. Equivalently, it is minimizing the amount of optimism in the policy, and thereby minimizing the other component contributing to the regret bound.

Next we show a concentration result for the optimism term.

Lemma 3. *Assume that the priors are layerwise-independent and that the uncertainty at each state-action decays as $\sigma^2(s, a) = \sigma^2/n(s, a)$ for some $\sigma > 0$, where $n(s, a)$ is the visitation count of the agent to (s, a). Then for any sequence of policies π_t, $t = 1, \ldots, N$ after $T = NL$ timesteps we have*

$$\mathbb{E} \sum_{t=1}^{N} \min_{\tau} \mathrm{Optimism}_{\phi^t}(\pi_t, \tau) \leq \tilde{O}(\sigma\sqrt{|\mathcal{S}||\mathcal{A}|T}),$$

where $\tilde{O}$ suppresses logarithmic terms.

The proof is included in Appendix B. Note that the above holds for any sequence of policies and has no dependence on the feasible policy set Π. This lemma tells us that under any sequence of policies, under the optimal choice of τ the expected cumulative sum of the $\mathrm{Optimism}$ terms grows sub-linearly. If $\Pi = \Delta(\mathcal{A})^{|\mathcal{S}|}$, then the optimal policy satisfies $\mathrm{Dist}(\pi, \tau) = 0$ in the above bound and corresponds exactly to the K-learning policy in Eq. (3), so have the we following corollary.

Corollary 1. *Assume 1 and 2 and let $\Pi = \Delta(\mathcal{A})^{|\mathcal{S}|}$. If algorithm* Alg *produces the policy that solves the saddle-point* (4) *for each episode t then after $T = NL$ timesteps*

$$\mathcal{BR}(\mathrm{Alg}, \phi) \leq \tilde{O}(\sigma\sqrt{|\mathcal{S}||\mathcal{A}|T}).$$

We can ensure that Assumption 1 holds in the case of bounded rewards, *i.e.*, $|r| \leq 1$ a.s., by setting $\sigma = O(L)$, which recovers the bound in O'Donoghue (2021).

4.2. Function approximation

In this manuscript we are interested in efficient reinforcement learning in non-tabular settings. In this case we must resort to using function approximators to parameterize the policy (or the value function) and we are interested in how well our function approximators will perform. Here we discuss the relationship between the capacity of the function approximator and the regret for our approach.

Consider the case where we are using an approximation architecture with feasible policy set $\Pi \subset \Delta(\mathcal{A})^{|\mathcal{S}|}$ chosen such that we can guarantee that for any τ

$$\min_{\pi\in\Pi} \max_{s,l} \mathrm{KL}(\pi_l(s, \cdot) \mid\mid \pi^{\star}_{l,\tau}(s, \cdot)) \leq \epsilon/\tau,$$

from which we have $\min_{\pi\in\Pi} \mathrm{Dist}(\pi, \tau) \leq \epsilon L$. This might occur if, for instance, we have an approximation architecture that can approximate the value functions up to a

small constant $\epsilon > 0$. Consider the regret of the policy $\pi_* = \text{argmax}_{\pi \in \Pi} \min_\tau \mathbb{E}_{s\sim\rho} J^\pi_{1,\tau}(s)$. In this case, using Theorem 1, we can bound the per-episode Bayesian regret as

$$\mathcal{R}(\pi_*, \phi) \leq \min_\tau \text{Optimism}(\pi, \tau) + \epsilon L.$$

and so the algorithm Alg_Π producing policies $\pi^t_* \in \Pi$, $t = 1, \ldots, N$ enjoys bound

$$\mathcal{BR}(\text{Alg}_\Pi, \phi) \leq \tilde{O}(\sigma\sqrt{|\mathcal{S}||\mathcal{A}|T}) + \epsilon T,$$

where $T = NL$ is the total number of timesteps. In other words, we can translate the error from the function approximation directly into a regret bound when solving the saddle-point problem (4), and richer function classes will yield better bounds.

On the other hand, consider the case where our approximation architecture is flexible enough to represent *any* policy, but our algorithm for choosing the policy employs an approximation procedure, such as online policy gradient. In that case the KL-divergence from the current policy to the optimistic policy is not zero, but if the policy is converging towards the optimal policy at some rate, then we may be able bound the sum of the KL divergences. There has been much recent work examining the convergence rate of policy gradient and entropy regularized policy gradient (under somewhat restrictive assumptions on the initial state distribution ρ) (Agarwal et al., 2021; Zhang et al., 2021; Bhandari & Russo, 2019; 2021; Mei et al., 2020). We leave to future work combining the results in this paper with results from the literature for the derivation of regret bounds in that case.

5. Epistemic-Risk-Seeking Actor-Critic

We have a derived a two-player zero-sum game, the solution of which yields a policy that explores efficiently by minimizing a bound on Bayesian regret. There are many possible approaches one could use to solve the saddle point problem, even in the purely online RL setting. In this section we describe a very simple approach that works reasonably well in practice, though it is likely that more sophisticated variants of policy algorithms would perform better (Schulman et al., 2017; Abdolmaleki et al., 2018; Schulman et al., 2015; Kakade, 2001). Our approach is to derive gradients for both the policy parameters and the risk-seeking parameter, then to update them online simultaneously using stochastic gradients. If we parameterize the policy π by some $\theta \in \Theta$, then the gradient of the saddle-point problem (4) with respect to θ is given by

$$\mathbb{E}_{s\sim\rho} \nabla_\theta J^\pi_{1,\tau}(s) = \sum_{l=1}^{L} \mathbb{E}_\pi \Big(\nabla_\theta \log \pi(s_l, a_l) K^\pi_{l,\tau}(s_l, a_l) + \tag{5}$$

$$\tau \nabla_\theta H(\pi(s_l, \cdot)) \Big). \tag{6}$$

This is a straightforward extension of the classic policy gradient theorem adapted to our case (Sutton et al., 1999). Similarly, the gradient with respect to τ is given by

$$\mathbb{E}_{s\sim\rho} \nabla_\tau J^\pi_{1,\tau}(s) = \sum_{l=1}^{L} \mathbb{E}_\pi \left(H(\pi(s_l, \cdot) - \frac{\sigma^2(s_l, a_l)}{2\tau^2} \right).$$

Finally, we have a relationship between the gradients and the Bayesian regret bound decomposition.

Corollary 2. *The gradients of the saddle-point correspond to the gradients of the components in the Bayesian regret decomposition in Lemma 2,* i.e.*,*

$$\mathbb{E}_{s\sim\rho}(-\nabla_\theta J^\pi_{1,\tau}(s), \nabla_\tau J^\pi_{1,\tau}(s)) = (\nabla_\theta \text{Dist}(\pi, \tau), \nabla_\tau \text{Optimism}(\pi, \tau)).$$

Fixing τ and taking a step in the negative gradient with respect to θ is towards minimizing the KL distance to the optimal optimistic policy, and for fixed θ taking a step in the direction of the gradient with respect to τ is towards minimizing the amount of optimism in the policy. Seen this way, the gradient flow is in the direction of minimizing the components in the Bayesian regret bound decomposition from Lemma 2.

Importantly, both of these gradient terms can be interpreted as expectations under the state-action distribution induced by the policy π. This suggests a scheme where we sample states and actions from the distribution generated by the policy, and use the same samples to update both quantities. We call this approach *epistemic-risk-seeking actor-critic* (ERSAC), and it is implemented as Algorithm 1 (presented in the appendix). Since this algorithm is applying stochastic gradient ascent-descent, rather than solving the saddle-point problem (4) exactly, we have no known Bayesian regret guarantees. However, as we shall demonstrate empirically, this algorithm tends to perform significantly better than vanilla actor-critic in hard exploration problems.

Estimating the uncertainty σ. In Algorithm 1 we left the process of deriving the estimator of K^π open. An estimator that performed well in practice is to use online TD$-\lambda$ with $\lambda = 0.8$ and a rollout length of $N = 50$ (Sutton & Barto, 1998).We have also left the source of the uncertainty signal $\sigma(s, a)$ undefined. There is much work in the deep RL literature that could be plugged into the algorithm here as discussed in §1. For our experiments we

augmented the neural network with an ensemble of reward prediction heads with randomized prior functions (Osband et al., 2018), and used the variance of the ensemble predictions as the uncertainty signal.

Comparison to other actor-critic methods. Algorithm 1 is a relatively small modification of a vanilla actor critic, the modifications are in blue. They are primarily the addition of the uncertainty terms, learning the τ risk-seeking parameter, and the addition of entropy regularization weighted with the learned τ. In our experiments we shall refer to the algorithm without these modifications as *vanilla actor-critic*. The presence of the reward predictors in Algorithm 1 can act as an auxiliary task and potentially improve the representation learned by the neural network thereby improving performance. This would give our agent an advantage over vanilla actor-critic that has nothing to do with exploration. To counter that, we also give the vanilla actor-critic agent the same reward prediction task, but we do not use the uncertainty estimates they generate.

A common pattern in optimistic deep RL algorithms is to simply add an optimism bonus to the rewards based on the standard deviation of the uncertainty, *i.e.*, replace the reward with $r^+(s,a) = \bar{r}(s,a) + \mu\sigma(s,a)$ for some hyper-parameter $\mu > 0$, and then run a vanilla actor-critic algorithm using this reward. In our experiments we shall refer to this variant as *simple optimism actor-critic*, where the uncertainty signal is the same ensemble approach as used by Algorithm 1 and all results are presented after tuning the μ hyper-parameter.

6. DeepSea Numerical Results

In the DeepSea environment the agent finds itself at the top left of an $L \times L$ grid and must navigate it to find the reward in the bottom right corner, see Figure 6. At each time-step the agent descends one row and must choose to move one column left or right. This is a challenging exploration unit-test because the agent needs to select the action 'move right' L times in a row in order to reach the goal (Osband et al., 2019) (in practice the actions corresponding to right and left are different in each state to prevent an agent with a bias for taking one action repeatedly from solving the problem unfairly). An agent that is acting randomly will take time exponential in L to reach the goal. However, agents that are exploring efficiently should reach the goal in time *polynomial* in L. Although DeepSea can be made a tabular environment, in this experiment we feed a one-hot representation of the agent location into a neural network in order to test how various deep RL approaches work. We compare four approaches: Vanilla actor-critic, ERSAC (Alg. 1) with a reward predictor ensemble size of 10, simple optimism actor-critic with the same uncertainty signal as Alg 1, and Bootstrapped DQN (Osband et al., 2016) with 10 elements in the value ensemble and 10 randomized priors (one per ensemble member). All agents had the same basic network architecture. Bootstrapped DQN performs an update with batch size of 128 samples every actor step which is the default in the agent implemented in the 'bsuite' (Osband et al., 2019). This uses substantially more compute and wall-clock time than the other approaches, and required a GPU to run efficiently. In Figure 1 we show the results of the four approaches. In that figure the blue dots represent solved DeepSea instances (where solved means the agent reached the goal reliably) and red dots are unsolved. The x-axis is depth and the y-axis is the number of episodes until that depth is solved. The grey dashed line is exponential in depth, which is the dependence we expect a naive agent to have. If the agent is consistently below this line, then it is exploring well.

As we can see, the naive vanilla actor-critic algorithm suffers from an exponential dependence on depth and consequently cannot solve depths of greater than around 14 within 10^5 episodes. Bootstrapped DQN is much faster than Algorithm 1 at learning the small DeepSea instances since it uses significant amounts of replay (though we close this gap in §7). However, as the DeepSea size grows it suddenly fails, unable to solve DeepSea instances larger than around size 50. The simple optimism actor-critic does provide some benefit over vanilla actor-critic, as it is able to solve DeepSeas out to approximately depth 50, however, the dependency on depth is significantly worse than ERSAC. ERSAC (Algorithm 1) is able to solve DeepSea instances out to size 100 without a clear performance degradation. In Figure 2 we show on a log-log plot that Algorithm 1 has an empirical *quadratic* dependency on depth, a major improvement over the exponential dependency of the naive actor-critic approach.

In Appendix C we further analyze the performance on DeepSea, and the sensitivity of Algorithm 1 to various hyper-parameters. We also test far deeper DeepSeas, including showing performance on a DeepSea of depth 250 where 99 out of 100 seeds reached the goal with 10^6 episodes. To the best of our knowledge no other deep RL algorithm has been able to solve such hard instances of DeepSea.

7. Incorporating Off-Policy Data

So far our discussion of Algorithm 1 has been entirely about the on-policy case. In practice however, state-of-the-art deep RL agents use a substantial amount of experience replay data, which vastly improves data efficiency and overall performance (Mnih et al., 2015; O'Donoghue et al., 2017; Hessel et al., 2018). Since exploration is also about increasing data efficiency, being able to combine re-

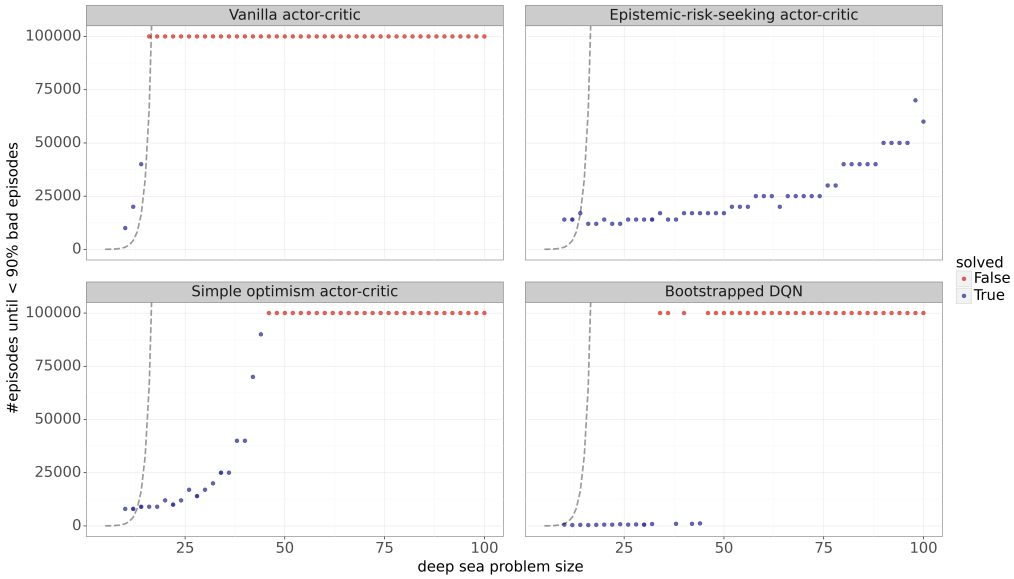

Figure 1. ERSAC is able to solve far deeper DeepSea instances than Bootstrapped DQN, despite requiring significantly less compute. Adding simple optimism to actor-critic provides some benefit, but it struggles to solver deeper instances. Vanilla actor-critic requires exponential experience to solve DeepSeas of increasing depth.

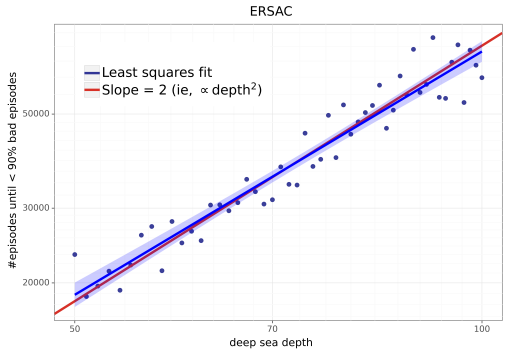

Figure 2. Algorithm 1 has an empirical quadratic dependency on depth when solving DeepSea.

play and principled exploration would yield a double improvement. In this section we extend Algorithm 1 to use off-policy replay data and show that combining the risk-seeking objective and replay can provide large performance improvements. To do that we make the following updates to the core algorithm:

- Add state-action-reward-noise (s_t, a_t, r_t, ζ_t), $t = 1, 2, \ldots,$ transition data to a replay buffer, where $\zeta_t \sim \mathcal{N}(0, \rho I_K)$ is independent noise with variance $\rho \geq 0$, and K is the size of the ensemble.
- Mix on-policy data with off-policy data sampled from the replay buffer according to a prioritization scheme (Schaul et al., 2015).
- Apply V-trace clipped importance sampling corrections to the off-policy trajectories (Espeholt et al., 2018).
- Use the reward + noise as targets for the reward prediction ensemble (Dwaracherla et al., 2022).

The above setup adds a small amount of Gaussian noise to the targets for the reward ensemble. This is necessary to prevent collapsing the uncertainty estimates from the use of replay data. It is important that the noise terms be added to the replay since this ensures that the epistemic uncertainty decays with the number of real data, rather than the number of replay steps. Using randomly initialized reward heads, randomized prior functions, and adding noise to the replay buffer (a form of Bayesian bootstrapping) follows the recipe analyzed in Dwaracherla et al. (2022) for good uncertainty estimates using ensembles. The V-trace clipped importance sampling re-weights the data coming from off-policy data according to how likely it is under the *current* policy, so that the gradient update in (5) is still (approximately) under the correct measure when using replay (Munos et al., 2016; Espeholt et al., 2018). We shall refer to Algorithm 1 when we add the changes above as 'ERSAC + replay'.

In Fig. 3 we compare the performance of ERSAC on DeepSea both with and without replay data. It is clear that adding replay data substantially improves performance, while maintaining the empirical quadratic dependence of solve time on depth (see Fig. 13). Overall, the ERSAC + replay agent yields about a $4\times$ data efficiency improvement over the pure on-policy version. The off-policy agent here used a batch size of 16 with an offline-data fraction of 0.97 per batch. Replay was prioritized by TD-error and when sampling the replay prioritization exponent was 1.0 (Schaul et al., 2015). The replay noise parameter was $\rho = 0.1$. All

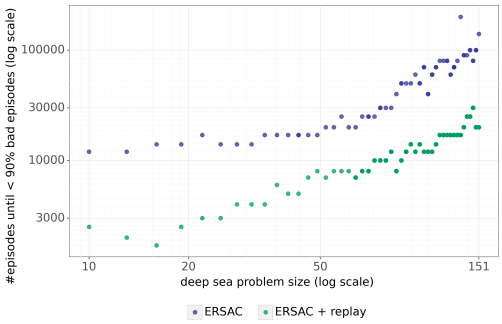

Figure 3. Adding replay to ERSAC improves data efficiency by a factor of about $4\times$ on DeepSea. Note the depth here goes to 151.

other settings were identical to the on-policy variant. In order to show the advantage of using noise in the replay buffer, we show results with and without noise on DeepSea in Figure 12.

There are two main ways in which replay may improve performance on DeepSea. First, it may reach the goal faster. Second, once the goal is reached it may 'latch on' faster, that is it may return to the goal consistently in fewer episodes. In Fig. 4 we show the reward of the agents on a depth 100 instance of DeepSea, averaged over 100 random seeds. It is clear that using replay is *both* finding the goal and latching on faster. However, we note that the on-policy version of the algorithm reached the goal 99 times out of 100, whereas the off-policy version reached the goal only 93 times. This suggest that the replay version may not be quite as robust as the on-policy version, at least for the hyper-parameters we used.

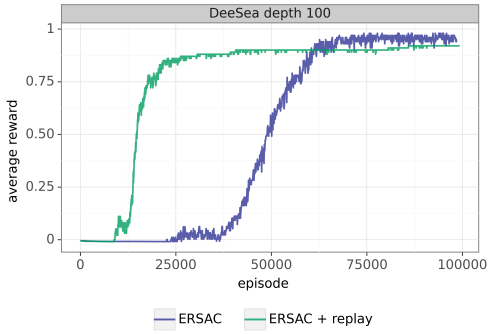

Figure 4. Adding replay to ERSAC improves both the time until goal first reached and the latching on speed in DeepSea.

8. Atari Numerical Results

Finally, we compare ERSAC + replay to an Actor-critic + replay agent on the Atari benchmark (Bellemare et al., 2012). Our setup involves actors generating experience and sending them to a learner, which mixes the online data and offline data from a replay buffer to update the network weights (Hessel et al., 2021; Mnih et al., 2016). Our agent is relatively simple compared to modern state-of-the-art Atari agents since it is missing components like model-based rollouts, distributional heads, auxiliary tasks, *etc.* The point of these experiments is not to produce state-of-the-art results, but to provide evidence of a clear benefit when the addition of the risk-seeking objective function is incorporated into a policy-gradient based agent. We ran both agents on the full Atari suite and averaged the results over five seeds. Between the agents all hyper-parameters in common were set to the same values, and tuned for the replay actor-critic agent performance. The replay actor-critic agent used a fixed entropy regularization of 0.02. The per-game results for all 57 games are presented in Figure 15, and Figure 5 shows the median human-normalized performance across the entire suite (calculated in the same way as Hessel et al. (2018)). Clearly the addition of the risk-seeking objective in Algorithm 1 is providing a significant benefit over the actor-critic agent. The ERSAC agent reaches the peak performance of the actor-critic agent in about $1.8\times$ fewer environment frames, for essentially the same computational cost. The advantage comes from the fact that the risk-seeking objective leads to deep exploration, which results in finding higher rewarding states and in better cumulative performance.

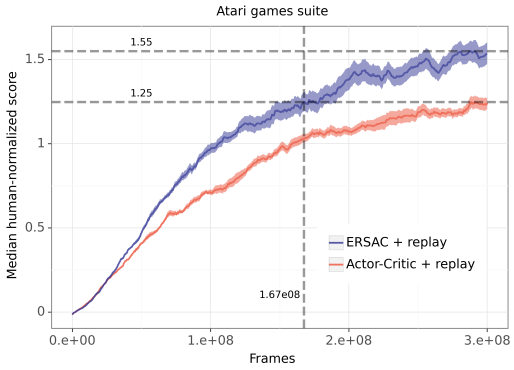

Figure 5. ERSAC reaches the same median performance on the Atari suite as the actor-critic baseline in about $1.8\times$ fewer frames.

9. Conclusion

We presented a new policy-gradient algorithm for efficient exploration. It was derived by endowing the agent with an epistemic-risk-seeking utility function, where the amount of risk-seeking is controlled by a risk-seeking parameter. The formulation entails solving a zero-sum game between the policy and the risk-seeking parameter. The policy is updated to maximize the optimistic reward, and the risk-seeking parameter is tuned to minimize regret. This procedure is a small modification to vanilla actor-critic but produces vastly improved results on challenging exploration problems.

References

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Heess, R. M. N., and Riedmiller, M. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations (ICLR)*, 2018.

Agarwal, A., Kakade, S. M., Lee, J. D., and Mahajan, G. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *J. Mach. Learn. Res.*, 22(98):1–76, 2021.

Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

Auer, P., Jaksch, T., and Ortner, R. Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21, 2008.

Azar, M. G., Osband, I., and Munos, R. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, pp. 263–272, 2017.

Barto, A. G. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, pp. 17–47. Springer, 2013.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pp. 1471–1479, 2016.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 2012.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Bhandari, J. and Russo, D. Global optimality guarantees for policy gradient methods. *arXiv preprint arXiv:1906.01786*, 2019.

Bhandari, J. and Russo, D. On the linear convergence of policy gradient methods for finite mdps. In *International Conference on Artificial Intelligence and Statistics*, pp. 2386–2394. PMLR, 2021.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Cover, T. M. and Thomas, J. A. *Elements of information theory*. John Wiley & Sons, 2012.

Dabney, W., Ostrovski, G., and Barreto, A. Temporally-extended ϵ-greedy exploration. *arXiv preprint arXiv:2006.01782*, 2020.

Dayan, P. and Sejnowski, T. J. Exploration bonuses and dual control. *Machine Learning*, 25(1):5–22, 1996.

Dimitrakakis, C. and Ortner, R. Decision making under uncertainty and reinforcement learning, 2018.

Dwaracherla, V., Wen, Z., Osband, I., Lu, X., Asghari, S. M., and Van Roy, B. Ensembles for uncertainty estimation: Benefits of prior functions and bootstrapping. *arXiv preprint arXiv:2206.03633*, 2022.

Eriksson, H. and Dimitrakakis, C. Epistemic risk-sensitive reinforcement learning. *arXiv preprint arXiv:1906.06273*, 2019.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Eysenbach, B. and Levine, S. If maxent RL is the answer, what is the question? *arXiv preprint arXiv:1910.01913*, 2019.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

Ghavamzadeh, M., Mannor, S., Pineau, J., and Tamar, A. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Hessel, M., Kroiss, M., Clark, A., Kemaev, I., Quan, J., Keck, T., Viola, F., and van Hasselt, H. Podracer architectures for scalable reinforcement learning. *arXiv preprint arXiv:2104.06272*, 2021.

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

Jin, C., Allen-Zhu, Z., Bubeck, S., and Jordan, M. I. Is Q-learning provably efficient? *Advances in neural information processing systems*, 31, 2018.

Kakade, S. A natural policy gradient. In *Advances in Neural Information Processing Systems*, volume 14, pp. 1531–1538, 2001.

Kearns, M. and Singh, S. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

Konda, V. R. and Tsitsiklis, J. N. On actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42 (4):1143–1166, 2003.

Lu, X., Van Roy, B., Dwaracherla, V., Ibrahimi, M., Osband, I., and Wen, Z. Reinforcement learning, bit by bit. *arXiv preprint arXiv:2103.04047*, 2021.

Mei, J., Xiao, C., Szepesvari, C., and Schuurmans, D. On the global convergence rates of softmax policy gradient methods. In *International Conference on Machine Learning*, pp. 6820–6829. PMLR, July 2020.

Meyn, S. *Control Systems and Reinforcement Learning*. Cambridge University Press, 2022.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, 02 2015. URL http://dx.doi.org/10.1038/nature14236.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pp. 1928–1937, 2016.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. Safe and efficient off-policy reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2772–2782, 2017.

Neu, G., Jonsson, A., and Gómez, V. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.

O'Donoghue, B. Variational Bayesian reinforcement learning with regret bounds. *Advances in Neural Information Processing Systems*, 34:28208–28221, 2021.

O'Donoghue, B. On the connection between Bregman divergence and value in regularized Markov decision processes. *arXiv preprint arXiv:2210.12160*, 2022.

O'Donoghue, B. and Lattimore, T. Variational Bayesian optimistic sampling. *Advances in Neural Information Processing Systems*, 34:12507–12519, 2021.

O'Donoghue, B., Munos, R., Kavukcuoglu, K., and Mnih, V. Combining policy gradient and Q-learning. In *International Conference on Learning Representations (ICLR)*, 2017.

O'Donoghue, B., Osband, I., Munos, R., and Mnih, V. The uncertainty Bellman equation and exploration. In *International Conference on Machine Learning*, pp. 3836–3845, 2018.

O'Donoghue, B., Lattimore, T., and Osband, I. Stochastic matrix games with bandit feedback. *arXiv preprint arXiv:2006.05145*, 2020.

Osband, I. *Deep Exploration via Randomized Value Functions*. PhD thesis, Stanford University, 2016.

Osband, I., Russo, D., and Van Roy, B. (More) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, pp. 3003–3011, 2013.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped DQN. In *Advances In Neural Information Processing Systems*, pp. 4026–4034, 2016.

Osband, I., Aslanides, J., and Cassirer, A. Randomized prior functions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Osband, I., Doron, Y., Hessel, M., Aslanides, J., Sezener, E., Saraiva, A., McKinney, K., Lattimore, T., Szepesvari, C., Singh, S., Roy, B. V., Sutton, R., Silver, D., and Hasselt, H. V. Behaviour suite for reinforcement learning. *arXiv preprint arXiv:1908.03568*, 2019.

Osband, I., Wen, Z., Asghari, S. M., Dwaracherla, V., Hao, B., Ibrahimi, M., Lawson, D., Lu, X., O'Donoghue, B., and Van Roy, B. The neural testbed: Evaluating joint predictions. *arXiv preprint arXiv:2110.04629*, 2021.

Ostrovski, G., Bellemare, M. G., Oord, A. v. d., and Munos, R. Count-based exploration with neural density models. *arXiv preprint arXiv:1703.01310*, 2017.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

Puterman, M. L. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., and Wen, Z. A tutorial on Thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. arXiv preprint arXiv:1511.05952, 2015.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *Proceedings of The 32nd International Conference on Machine Learning*, pp. 1889–1897, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Singh, S. P., Barto, A. G., and Chentanez, N. Intrinsically motivated reinforcement learning. In *NIPS*, volume 17, pp. 1281–1288, 2004.

Stadie, B. C., Levine, S., and Abbeel, P. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

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

Strens, M. A Bayesian framework for reinforcement learning. In *ICML*, pp. 943–950, 2000.

Sutton, R. and Barto, A. *Reinforcement Learning: an Introduction*. MIT Press, 1998.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 99, pp. 1057–1063, 1999.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Xi Chen, O., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

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

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in Starcraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Zhang, J., Kim, J., O'Donoghue, B., and Boyd, S. Sample efficient reinforcement learning with REINFORCE. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10887–10895, 2021.

Ziebart, B. D. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

A. Main algorithm

Algorithm 1 Epistemic-risk-seeking actor-critic (ERSAC)

Input initial parameters $\theta^0 \in \Theta$, $\tau^0 > 0$, uncertainty estimator $\sigma : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$
Input policy function $\pi_\theta : \mathcal{S} \to \Delta(\mathcal{A})$ and value function $J_\theta : \mathcal{S} \to \mathbb{R}$
For $k = 0, 1, \ldots$
  Gather trajectory $\nu = (r_1, s_1, a_1, \ldots, r_N, s_N)$ using π^k
  Compute uncertainties $\sigma(s_i, a_i)$, $i = 1, \ldots N$
  Estimate $\hat{K}^\pi_{l,\tau}(s_i, a_i)$ using $r_i, J_{\theta^k}(s_i), \sigma(s_i, a_i), \tau^k$, $i = 1, \ldots, N$, and Eq. 2
  $L_{\text{policy}} = (1/N) \sum_{i=1}^N \left(\log \pi_{\theta^k}(s_i, a_i) \texttt{stop\_grad}(\hat{K}^\pi_{l,\tau}(s_i, a_i) - J_{\theta^k}(s_i)) - \tau^k H(\pi_{\theta^k}(s_i, \cdot)) \right)$
  $L_{\text{value}} = (1/N) \sum_{i=1}^N (J_{\theta^k}(s_i) - \texttt{stop\_grad}(\hat{K}^\pi_{l,\tau}(s_i, a_i) - \tau^k \log \pi_{\theta^k}(s_i, a_i)))^2$
  $L_\tau = (1/N) \sum_{i=1}^N \left(\frac{\sigma^2(s_i, a_i)}{2\tau} + \tau H(\pi_{\theta^k}(s_i, \cdot)) \right)$
  $\theta^{k+1} = \theta^k + \eta(\nabla_\theta L_{\text{policy}} - \nabla_\theta L_{\text{value}})$
  $\tau^{k+1} = \tau^k - \eta \nabla_\tau L_\tau$
  Update uncertainty model σ using ν

B. Proofs

Lemma 2. *Under Assumption 1 we can bound the Bayesian regret in a single episode for any policy π as*

$$\mathcal{R}(\pi, \phi) \leq \text{Dist}(\pi, \tau) + \text{Optimism}(\pi, \tau), \quad \forall \tau \geq 0.$$

Proof.

$$\begin{aligned}
\mathcal{R}(\pi, \phi) &= \mathbb{E}_\phi \mathbb{E}_{s\sim\rho}(V_1^\star(s) - V_1^\pi(s)) \\
&\leq \mathbb{E}_{s\sim\rho}(J^\star_{1,\tau}(s) - \mathbb{E}_\phi V_1^\pi(s)) \\
&= \mathbb{E}_{s\sim\rho}(J^\star_{1,\tau}(s) - J^\pi_{1,\tau}(s) + J^\pi_{1,\tau}(s) - \mathbb{E}_\phi V_1^\pi(s)) \\
&= \text{Dist}(\pi, \tau) + \text{Optimism}(\pi, \tau).
\end{aligned}$$

□

Theorem 1. *Assume 1 and 2, and let (π_*, τ_*) be a solution to the saddle-point problem* (4)*, then*

$$\mathcal{R}(\pi_*, \phi) \leq \min_{\pi\in\Pi} \text{Dist}(\pi, \tau_*) + \min_\tau \text{Optimism}(\pi_*, \tau).$$

Proof. We can rewrite the saddle-point formulation in two ways. For any π we have

$$\begin{aligned}
\min_\tau \mathbb{E}_{s\sim\rho} J^\pi_{1,\tau}(s) &= \min_\tau \mathbb{E}_{s\sim\rho}(J^\pi_{1,\tau}(s) - \mathbb{E}_\phi V_1^\pi(s) + \mathbb{E}_\phi V_1^\pi(s)) \\
&= \mathbb{E}_{s\sim\rho}\mathbb{E}_\phi V_1^\pi(s) + \min_\tau \text{Optimism}(\pi, \tau),
\end{aligned}$$

and for any τ

$$\begin{aligned}
\max_{\pi\in\Pi} \mathbb{E}_{s\sim\rho} J^\pi_{1,\tau}(s) &= \max_{\pi\in\Pi} \mathbb{E}_{s\sim\rho}(J^\pi_{1,\tau}(s) - J^\star_{1,\tau}(s) + J^\star_{1,\tau}(s)) \\
&= \mathbb{E}_{s\sim\rho} J^\star_{1,\tau}(s) - \min_{\pi\in\Pi} \text{Dist}(\pi, \tau).
\end{aligned}$$

From strong duality and the fact that (π_*, τ_*) is a primal-dual optimum we know that $\max_{\pi\in\Pi} \mathbb{E}_{s\sim\rho} J^\pi_{1,\tau_*}(s) = \mathbb{E}_{s\sim\rho} J^{\pi_*}_{1,\tau_*}(s) = \min_\tau \mathbb{E}_{s\sim\rho} J^{\pi_*}_{1,\tau}(s)$, which implies

$$\mathbb{E}_{s\sim\rho} J^\star_{1,\tau_*}(s) - \min_{\pi\in\Pi} \text{Dist}_\phi(\pi, \tau_*) = \mathbb{E}_{s\sim\rho}\mathbb{E}_\phi V_1^{\pi_*}(s) + \min_\tau \text{Optimism}(\pi_*, \tau)$$

and so

$$\begin{aligned}\mathcal{R}(\pi_*, \phi) &\leq \mathbb{E}_{s\sim\rho}(J^\star_{1,\tau_*}(s) - \mathbb{E}_\phi V_1^{\pi_*}(s)) \\ &= \min_{\pi\in\Pi} \mathrm{Dist}(\pi, \tau_*) + \min_\tau \mathrm{Optimism}(\pi_*, \tau).\end{aligned}$$

□

Lemma 3. *Assume that the priors are layerwise-independent and that the uncertainty at each state-action decays as $\sigma^2(s,a) = \sigma^2/n(s,a)$ for some $\sigma > 0$, where $n(s,a)$ is the visitation count of the agent to (s,a). Then for any sequence of policies π_t, $t = 1, \ldots, N$ after $T = NL$ timesteps we have*

$$\mathbb{E}\sum_{t=1}^{N} \min_\tau \mathrm{Optimism}_{\phi^t}(\pi_t, \tau) \leq \tilde{O}(\sigma\sqrt{|\mathcal{S}||\mathcal{A}|T}),$$

where $\tilde{O}$ suppresses logarithmic terms.

Proof. At episode t denote the uncertainty at state-action (s,a) as $\sigma^2/n^t(s,a)$, where $n^t(s,a)$ is the visitation count of (s,a) before episode t. Under the assumption of independent priors across layers we can write

$$\mathbb{E}_{s\sim\rho}\mathbb{E}_{\phi^t} V_1^{\pi^t}(s) = \sum_{l=1}^{L} \mathbb{E}_{\pi^t} \bar{r}_l^t(s_l, a_l),$$

and recall that

$$\mathbb{E}_{s\sim\rho} J_{1,\tau}^{t,\pi^t}(s) = \sum_{l=1}^{L} \mathbb{E}_{\pi^t}\left(\bar{r}_l^t(s_l,a_l) + \frac{\sigma_l^2}{2\tau n^t(s_l,a_l)} + \tau H(\pi_l^t(s_l,\cdot))\right),$$

and so we can write

$$\begin{aligned}\mathrm{Optimism}(\pi_t, \tau) &= \mathbb{E}_\phi \mathbb{E}_{s\sim\rho}(J^\pi_{1,\tau}(s) - V_1^\pi(s)) \\ &= \sum_{l=1}^{L} \mathbb{E}_\pi\left(\frac{\sigma^2}{2\tau n^t(s_l,a_l)} + \tau H(\pi_l^t(s,\cdot))\right).\end{aligned}$$

Now define scalar (up to log factors which we shall ignore for brevity)

$$\tau_N = \tilde{O}(\sigma\sqrt{|\mathcal{S}||\mathcal{A}|/(LN)}).$$

Then we have

$$\begin{aligned}\sum_{t=1}^{N} \min_\tau \mathrm{Optimism}_{\phi^t}(\pi_t, \tau) &\leq \sum_{t=1}^{N} \mathrm{Optimism}_{\phi^t}(\pi_t, \tau_N) \\ &= \sum_{t=1}^{N}\sum_{l=1}^{L} \mathbb{E}_\pi\left(\frac{\sigma^2}{2\tau_N n^t(s_l,a_l)} + \tau_N H(\pi_l^t(s,\cdot))\right) \\ &\leq (1/2)\sigma^2|\mathcal{A}|(1+\log N)\tau_N^{-1}\sum_{l=1}^{L}|\mathcal{S}_l| + \tau_N NL\log|\mathcal{A}| \\ &\leq \tilde{O}(\sigma\sqrt{L|\mathcal{S}||\mathcal{A}|N}),\end{aligned}$$

where we used the fact that entropy is bounded, the pigeonhole principle Lemma 6 from (O'Donoghue, 2021), and the identity $\sum_{l=1}^{L}|\mathcal{S}_l| = |\mathcal{S}|$. The result follows by substituting in $T = NL$. □

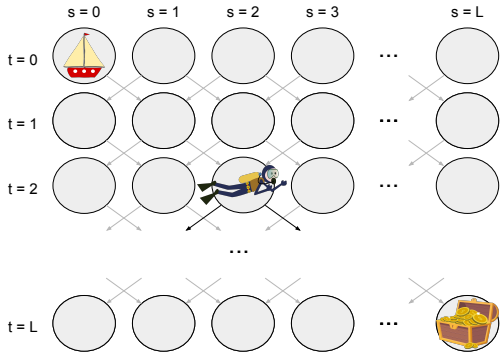

Figure 6. The DeepSea MDP is a challenging exploration 'unit-test' where the agent must navigate from the top left state to the bottom right in order to collect a positive reward. Naive exploration approaches take time exponential in depth to solve this problem.

C. DeepSea results discussion

K-learning has a worst-case $\tilde{O}(L\sqrt{|\mathcal{S}||\mathcal{A}|T})$ Bayesian regret in tabular domains. In a DeepSea of depth d we have $L = d$, $S = d^2$, $A = 2$, so this regret bound would translate as $\tilde{O}(d^2\sqrt{T})$, and to have *average* regret below some threshold would require $O(d^4)$ timesteps, or $O(d^3)$ episodes. Algorithm 1 is an online, stochastic policy gradient based approximation to K-learning, so we have no known regret bound guarantee. However, in Figure (2) we find that empirically for Algorithm 1 the number of episodes required to 'solve' a DeepSea instance appears to have a quadratic dependency on depth, a factor of d better than the worst-case bound. Naive approaches to exploration require episodes scaling as $O(2^d)$, so a quadratic dependency is a substantial improvement.

Our agent used TD-λ with $\lambda = 0.8$, Figure 7 we show the performance of the agent as a function of the λ parameter. It appears that values of $\lambda \geq 0.6$ perform well, able to solve most or all of the DeepSea instances out to depth 100. Figure 8 shows the robustness of the method to TD-λ rollout length. For very small rollouts (*e.g.*, 1) the benefit of Algorithm 1 over vanilla actor-critic is minor, however the epistemic-risk-seeking agent is able to solve practically all DeepSea instances to depth 100 reliably for just a rollout of length 25.

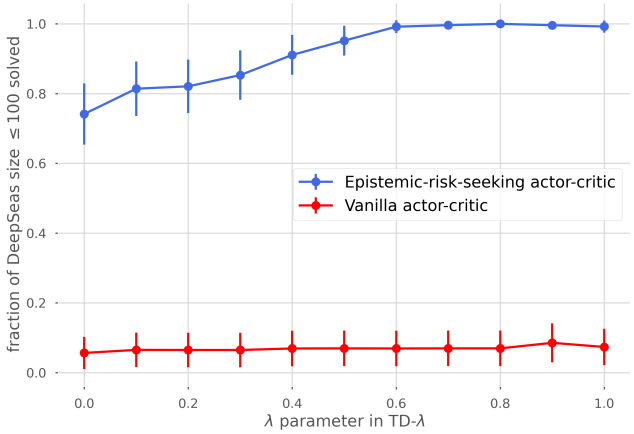

Figure 7. When using TD-λ to estimate the K-values in Algorithm 1 larger λ values tend to perform better in DeepSea.

Finally, we also tested how important *learning* the risk-seeking parameter τ is, as done in Algorithm 1. In Figure 9 we

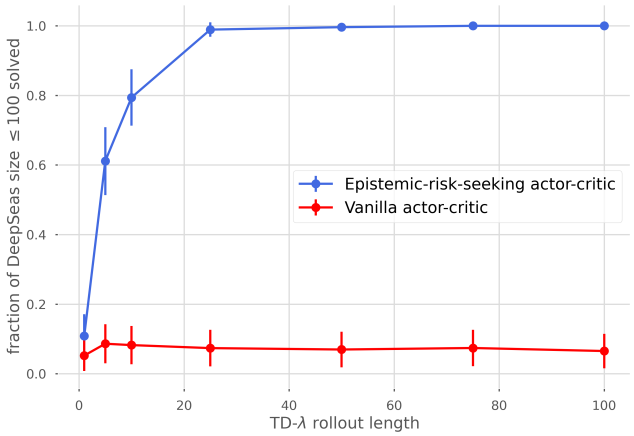

Figure 8. When using TD-λ to estimate the K-values in Algorithm 1 even relatively small rollout lengths are able to solve deeper DeepSea instances. Even a rollout length of 5 is able to solve more than 60% of DeepSea instances, and 25 is enough to solve practically all of them.

compare the approach in Algorithm 1 to simply using a fixed τ parameter. From this Figure it appears that there is a fixed choice of τ that matches the learned approach on DeepSea performance. However, the performance of the agent is highly dependent on this parameter and even small deviations can dramatically degrade reliability. On the other hand, Algorithm 1, which learns τ from data, is able to solve almost all the DeepSea instances robustly over a wide range of initial choices of τ, which suggests that the update rule that minimizes the zero-sum game (4) over τ is effective at tuning the amount of risk-seeking for efficient exploration.

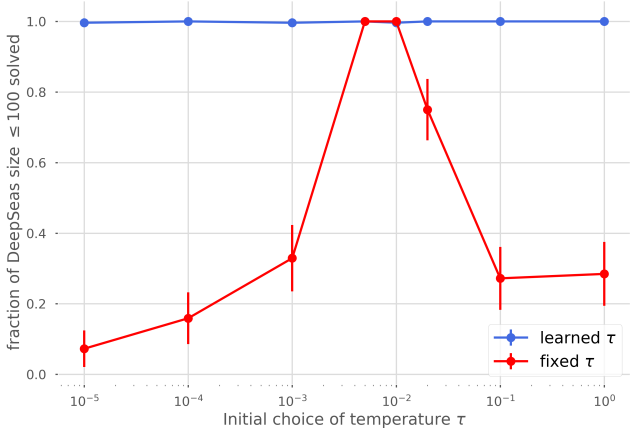

Figure 9. The optimal fixed risk-seeking parameter τ can produce good results, but learning τ via Algorithm 1 is far more robust.

The excellent performance of Algorithm 1 on DeepSea raises the question: What is the maximum depth that the algorithm is able to consistently solve? We ran the algorithm on a DeepSea of depth 250 with 100 random seeds to see how the performance degraded with depth. To handle the longer episode length before a reward we increased both the discount factor to $\gamma = 0.999$ and the λ factor in TD-λ to 0.95. The average performance is plotted in 10. Overall, 99 out of the 100 seeds managed to reach the goal within 10^6 episodes. In order to reach a positive reward the agent must make the exact right sequence of 250 actions and any deviation is impossible to recover from. This is a very difficult problem and one

that would require an enormous number of episodes for a simple dithering agent, since $2^{250} \approx 10^{75}$. This suggests that Algorithm 1 is able to handle extremely deep and difficult DeepSeas without much degradation in performance, and the limit has not yet been reached.

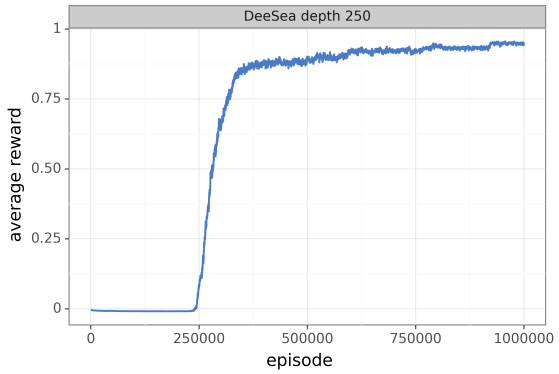

Figure 10. Performance on DeepSea of depth 250 for Algorithm 1 averaged over 100 seeds. Overall, 99 out of 100 seeds managed to reach the goal within 10^6 episodes.

C.1. DeepSea replay experiments

In Figure 12 we show the benefit of adding noise to the reward samples in the replay buffer. It is clear that without the addition of noise the replay is destroying the uncertainty estimates and leading to worse performance than without replay. However, once the noise is added the agent with replay outperforms the purely online agent. Figure 11 is the same as Figure 3 except on a linear, rather than log, scale. Figure 13 shows that adding replay to ERSAC does not appear to alter the quadratic dependency of solve time on depth.

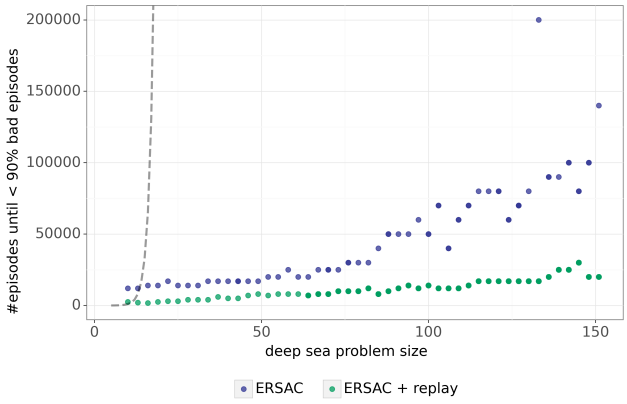

Figure 11. Adding replay to the epistemic-risk-seeking actor-critic improves data efficiency by a factor of about $4\times$. Note the depth here goes out to 151.

D. Atari results

In Figure 14 we present the performance of Algorithm 1 compared to the vanilla actor-critic algorithm on a collection of 7 hard exploration games from the Atari 57 suite (Bellemare et al., 2012). In Figure 15 we compare the performance of the agents across all 57 Atari games.

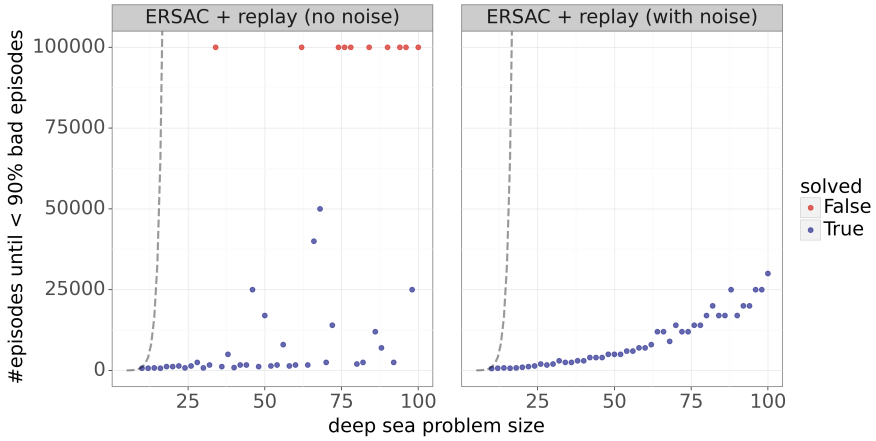

Figure 12. Adding noise to the reward targets when using replay dramatically improves the uncertainty estimates and the performance of the agent.

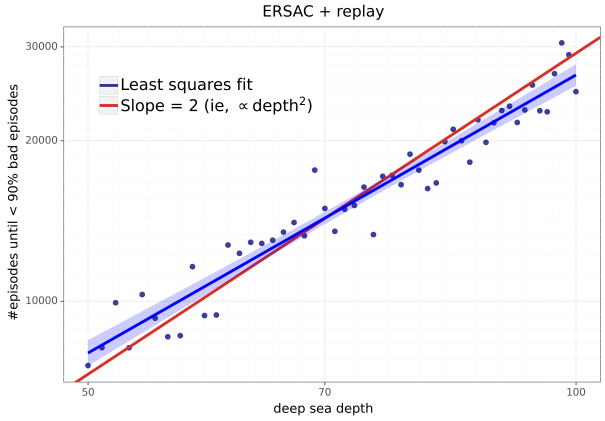

Figure 13. The solve time for ERSAC + replay on DeepSea has the same empirical quadratic dependency with depth as ERSAC without replay, but it is about $4\times$ faster overall.

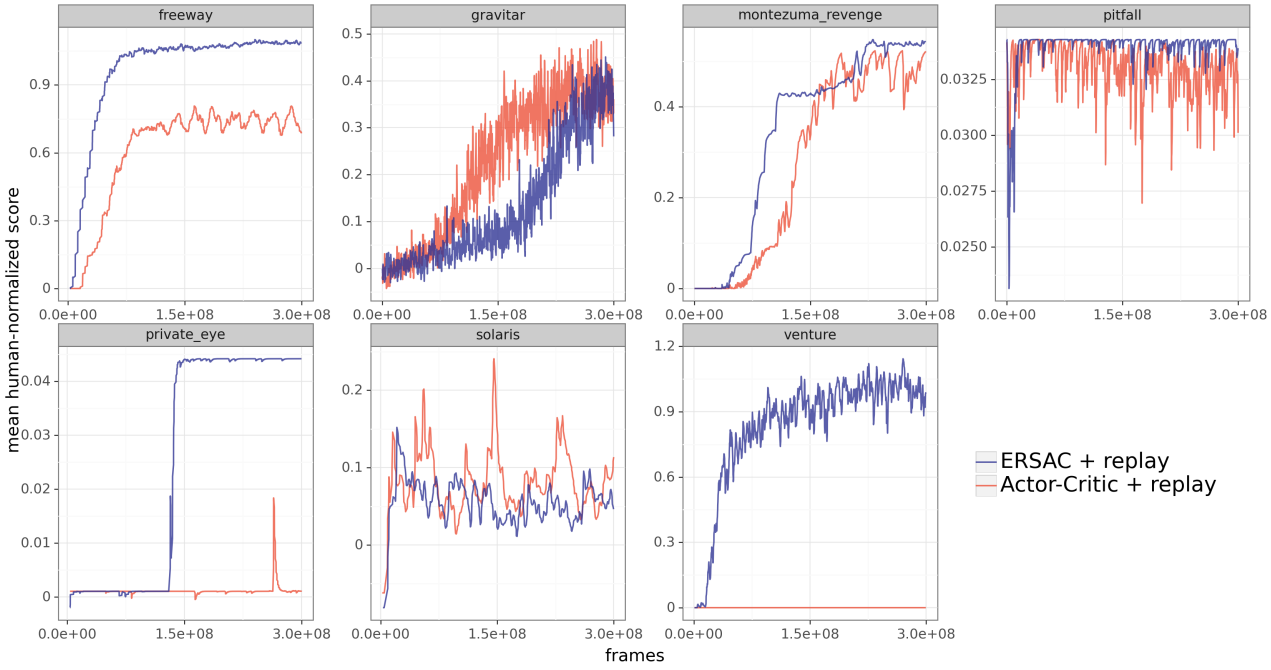

Figure 14. In this collection of hard exploration Atari games we see that the epistemic actor-critic algorithm provides a performance improvement over Replay actor-critic in four of the 7 games. In particular, there is a significant performance improvement for the very hard exploration game 'Montezuma's revenge'.

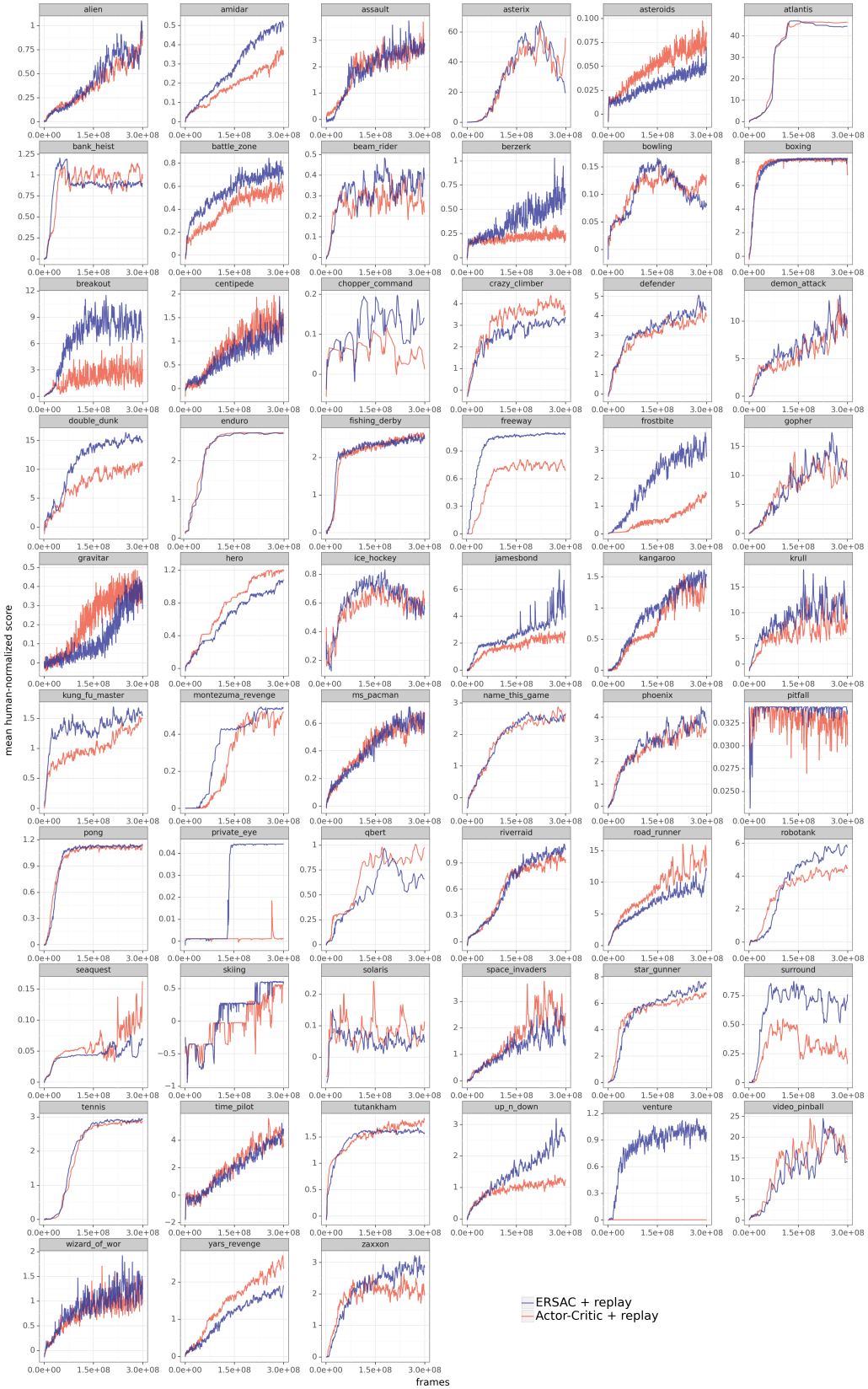

Figure 15. Performance of ERSAC and an actor-critic agent across all 57 Atari games.

E. Future work

We conclude with some discussion about future directions for this work. One question that this work raises is whether it is appropriate to have a single risk-seeking (entropy regularization) parameter τ for all states and actions (Ziebart, 2010; Neu et al., 2017; O'Donoghue et al., 2017; Nachum et al., 2017; Eysenbach & Levine, 2019; Haarnoja et al., 2018). Some preliminary work (O'Donoghue & Lattimore, 2021) suggests that in fact it is both possible and advantageous to have a separate risk-seeking parameter for each state-action pair. In future work we may wish to investigate this. Simple actor-critic methods are no longer state-of-the-art, with most effective policy-based agents employing a range of different tactics to improve performance such as trust-regions, Q-value critics, natural gradients, model-based rollouts *etc*. An interesting extension would be to incorporate the techniques discussed here into those agents. We discussed at a high-level the regret of the formulation we derive in §2.1, and showed empirical regret scaling results in Figure 2. In future work it would be interesting to combine the results of this work with theoretical results on the convergence rate of policy gradient algorithms to derive a concrete regret bound for a epistemic-risk-seeking policy-gradient algorithm.