Predictable Reinforcement Learning Dynamics through Entropy Rate Minimization

Anonymous authors
Paper under double-blind review

Abstract

In Reinforcement Learning (RL), agents have no incentive to exhibit predictable behaviors, and are often pushed (through e.g. policy entropy regularisation) to randomise their actions in favor of exploration. This often makes it challenging for other agents and humans to predict an agent's behavior, triggering unsafe scenarios (e.g. in human-robot interaction). We propose a novel method to induce predictable behavior in RL agents, termed *Predictability-Aware RL* (PARL), employing the agent's trajectory *entropy rate* to quantify predictability. Our method maximizes a linear combination of a standard discounted reward and the negative entropy rate, thus trading off optimality with predictability. We show how the entropy rate can be formally cast as an average reward, how entropy-rate value functions can be estimated from a learned model and incorporate this in policy-gradient algorithms, and demonstrate how this approach produces predictable (near-optimal) policies in tasks inspired by human-robot use-cases.

1 Introduction

As Reinforcement Learning (RL) (Sutton & Barto, 2018) agents are deployed to interact with humans, it becomes crucial to ensure that their behaviors are predictable. A robot trained under general RL algorithms operating in a human environment has no incentive to follow trajectories that are easy to predict. This makes it challenging for other robots or humans to forecast the robot's behavior, affecting coordination and interactions, and possibly triggering unsafe scenarios. RL algorithms are oblivious to the predictability of behaviors they induce in agents: one aims to maximize an expected reward, regardless of how unpredictable trajectories taken by the agents may be. In fact, many algorithms propose some form of regularisation in action complexity (Schulman et al., 2017; Han & Sung, 2021) or value functions (Pitis et al., 2020; Zhao et al., 2019; Kim et al., 2023) for better exploration, inducing higher aleatoric uncertainty in agents' trajectories.

We quantify predictability of an RL agent's trajectories by employing the notion of *entropy rate*: the infinite-horizon time-average entropy of the agent's trajectories, which measures the complexity of the trajectory distributions induced in RL agents. Higher entropy rate implies more complex and less predictable trajectories, and vice-versa. Similar information-theoretic metrics have been widely used to quantify (un)predictability of stochastic processes Shannon (1948); Savas et al. (2021); Biondi et al. (2014); Duan et al. (2019); Stefansson & Johansson (2021).

Motivation In general, RL agents are oblivious to the information theoretic loads they generate with their behavior. In a world where agents do not exist in a vacuum (even if we train RL agents in single-agent settings, these agents will rarely be deployed in isolation), one could argue there is an advantage to inducing a *complexity awareness* in RL agents; *If an agent can solve a task generating lower information rates, it should do so.* Lower information rates correspond (intuitively and formally, through forms of entropy) with lower uncertainty. However, we do not argue that this is a necessary feature in all agents (or even always desireable). We simply argue that it is an interesting feature to consider for general RL agents that can benefit the deployment of RL agents, propose a formal approach to target this, and evaluate how this impacts such agents in different settings.

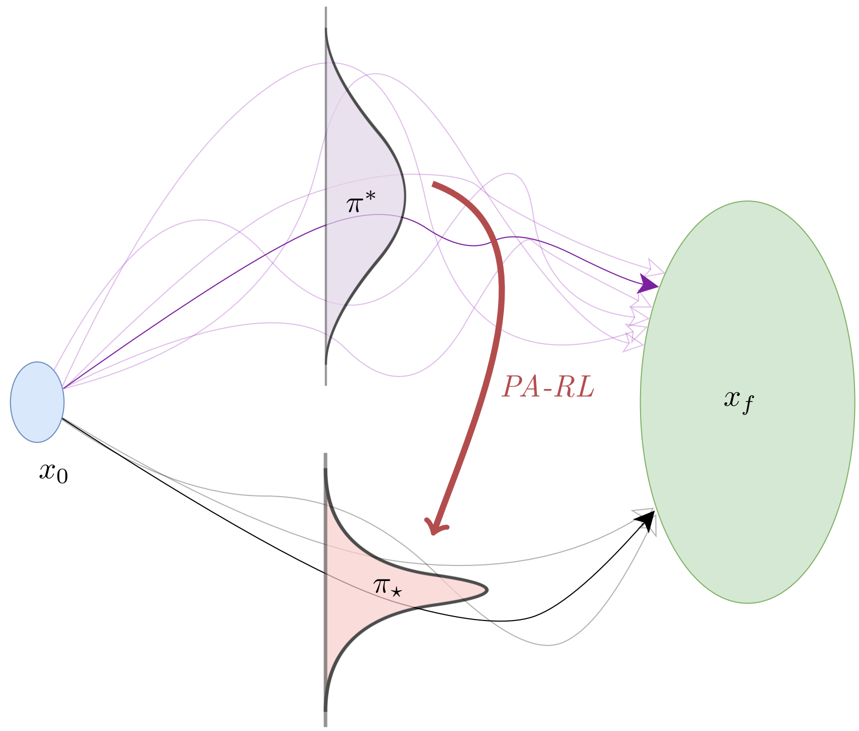


Figure 1: Qualitative representation of PARL. Trajectories are represented symbolically as connecting an initial with a final set of states. PARL shifts the policies towards smaller trajectory entropy.

Contributions We propose a novel approach to model-based RL that induces more predictable behavior in RL agents, termed *Predictability-Aware RL* (PARL). We maximize the linear combination of a standard discounted cumulative reward and the negative entropy rate, thus trading-off optimality with predictability. Towards this, we cast entropy-rate minimization as an expected average reward minimization problem, with a policy-dependent reward function, called *local entropy.* To circumvent local entropy's policy-dependency and enable the use of on- and off-policy RL algorithms, we introduce a state-action-dependent surrogate entropy, and show that deterministic policies minimizing the average surrogate entropy exist and *also minimize the entropy rate.* Further, we show how, employing a learned model and the surrogate reward, we can estimate entropy-rate value functions, and incorporate this in policy-gradient schemes. Finally, we showcase how PARL produces much more predictable agents while achieving near-optimal rewards in several robotics and autonomous driving tasks.

1.1 Related work

The idea of introducing some form of entropy objectives in policy gradient algorithms has been extensively explored Williams & Peng (1991); Fox et al. (2015); Peters et al. (2010); Zimin & Neu (2013); Neu et al. (2017); Tiapkin et al. (2023). In most instances, these regularization terms are designed to either help policy randomization and exploration, or to stabilize RL algorithms. However, these works focus on policy (state-action) entropy maximization, and do not focus on trajectory entropy and how it affects predictability of RL agents.

Instead, Guo et al. (2023b;a); Biondi et al. (2014); George et al. (2018); Duan et al. (2019); Stefansson & Johansson (2021); Savas et al. (2020; 2021) consider entropy (rate) maximization in (PO)MDPs, to yield unpredictable behaviors. However, these works require full knowledge of the model, and entropy (rate) maximization is cast as a non-linear program. Instead, in our work, the model is not known, and we resort to learning entropy-rate value functions.

Recent work Lu et al. (2020); Eysenbach et al. (2021); Park & Levine (2023) has tackled robustness and generalization via introducing Information-Theoretic penalty terms in the reward function. In particular, Eysenbach et al. 2021 makes the explicit connection from such information-theoretic penalties to the emergence of predictable behavior in RL agents, and uses mutual information penalties to restrict the bits of information that the agents are allowed to use, resulting in simpler, less complex policies. We address directly this predictability problem by the minimisation of entropy rates in RL agents' stochastic dynamics. This allows us to provide theoretical results regarding existence and convergence of optimal (predictable) policies towards *minimum entropy-rate* agents, and make our scheme generalizeable to *any RL algorithm* (on and off policy). Finally, our work is tangentially related to alignment and interpretability in RL Shah et al. (2019); Carroll et al. (2019), where human-agent interaction requires exhibiting human-interpretable behavior. Further, work on legibility of robot motion Dragan et al. (2013); Liu et al. (2023); Busch et al. (2017) shares our motivation; to make robotic systems behavior more legible by humans.

2 Background

We, first, introduce preliminary concepts employed throughout this work. For more detail on Markov chains and decision processes, the reader is referred to Puterman (2014). Given a set $\mathcal{A}$, $\Delta(\mathcal{A})$ denotes the probability simplex over $\mathcal{A}$, and $\mathcal{A}^k$ denotes the k-times Cartesian product $\mathcal{A} \times \mathcal{A} \times \cdots \times \mathcal{A}$. If $\mathcal{A}$ is finite, $|\mathcal{A}|$ denotes its cardinality. Given two probability distributions μ, ν, we use $D_{TV}(\mu\|\nu)$ as the total variation distance between two distributions. We use $\mathrm{supp}(\mu)$ to denote the support of μ. Given two vectors ξ, η, we write $\xi \succeq \eta$, if each i-th entry of ξ is bigger or equal to the i-th entry of η.

2.1 Markov processes and Rewards

A Markov Chain (MC) is a tuple $\mathcal{C} = (\mathcal{X}, P, \mu_0)$ where $\mathcal{X}$ is a finite set of states, $P : \mathcal{X} \times \mathcal{X} \to [0, 1]$ is a transition probability measure and $\mu_0 \in \Delta(\mathcal{X})$ is a probability distribution over initial states. Specifically, $P(x, y)$ is the probability of transitioning from state x to state y. $P^t(x, y)$ is the probability of landing in y after t time-steps, starting from x. The limit transition function is $P^* := \lim_{t\to\infty} P^t$. We use uppercase X_t to refer to the random variable that is the state of the random process governed by the MC at time t, and lower case to indicate specific states, *e.g.* $x \in \mathcal{X}$. Similarly, a trajectory or a path is a sequence of states $\mathbf{x}_k = \{x_0, x_1, ..., x_k\}$, where $x_i \in \mathcal{X}$, and $\mathcal{P}_k$ denotes the set of all $(k + 1)$-length paths. We also denote $X_{t:k} = \{X_t, X_{t+1}, ..., X_k\}$. Further, we define $p : \mathcal{P}_\infty \to [0, 1]$ as a probability measure over the Borel σ-algebra $\mathcal{B}(X_{0:\infty})$ of infinite-length paths of a MC[1] conditioned to initial distribution μ_0. A Markov Decision Process (MDP) is a tuple $\mathcal{M} = (\mathcal{X}, \mathcal{U}, P, R, \mu_0)$ where $\mathcal{X}$ is a finite set of states, $\mathcal{U}$ is a finite set of actions, $P : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to [0, 1]$ is a probability measure of the transitions between states given an action, $R : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to \mathbb{R}$ is a bounded reward function and $\mu_0 \in \Delta(\mathcal{X})$ is the probability distribution of initial states. A stationary Markov policy is a stochastic kernel $\pi : \mathcal{X} \to \Delta(\mathcal{U})$. With abuse of notation, we use $\pi(u \mid x)$ as the probability of taking action u at state x, under policy π. Let Π be the set of all stationary Markov policies, and $\Pi^D \subseteq \Pi$ the set of deterministic policies. The composition of an MDP $\mathcal{M}$ and a policy $\pi \in \Pi$ generates a MC with transition probabilities $P_\pi(x, y) := \sum_{u\in\mathcal{U}} \pi(u \mid x)P(x, u, y)$. If said MC admits a unique stationary distribution, we denote it by μ^π, where $\mu^\pi : \mathcal{X} \to [0, 1]$. We, also, use the shorthand $R^\pi_t \equiv \mathbb{E}_{u\sim\pi(x)}[R(X_t, u, X_{t+1})]$.

Assumption 1. *Any fixed policy π in MDP $\mathcal{M}$ induces an aperiodic and irreducible MC.*

Discounted Reward MDPs In discounted cumulative reward maximization problems the goal is to find a policy π' that maximizes the discounted sum of rewards for discount factor $\gamma \in [0, 1)$: i.e. $\pi' \in \arg\max_{\pi\in\Pi} \mathbb{E}[\sum_{t=0}^\infty \gamma^t R^\pi_t \mid X_0 = x]$, for all $x \in \mathcal{X}$. In the case of discounted cumulative reward maximization Puterman (2014), given a policy π, the *value function* under π, $V^\pi : \mathcal{X} \mapsto \mathbb{R}$, is $V^\pi(x) := \mathbb{E}[\sum_{t=0}^\infty \gamma^t R^\pi_t \mid X_0 = x]$. The *action-value function* (or Q-function) under π is given by $Q^\pi(x, u) := \sum_{y\in\mathcal{X}} P(x, u, y)(R(x, u, y) + \gamma V^\pi(y))$.

[1] This measure is well defined by the Ionescu-Tulcea Theorem, see *e.g.* Dudley (2018).

Average Reward MDPs In average reward maximization problems, we aim at maximizing the *reward rate* (or gain) $g^\pi(x)$, defined as, together with the *bias*[2]

$$g^\pi(x) := \mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T} R_t^\pi \mid X_0 = x\right], \; b^\pi(x) := \mathbb{E}\left[\lim_{T\to\infty} \sum_{t=0}^{T} (R_t^\pi - g(X_t)) \mid X_0 = x\right]. \tag{1}$$

Note that the bias is the expected difference between the stationary rate and the rewards obtained by initialising the system at a given state. For $\pi \in \Pi$, the average-reward (action) value-functions[3] $V^\pi_{\text{avg}} : \mathcal{X} \to \mathbb{R}$ is defined by Abounadi et al. (2001) $V^\pi_{\text{avg}}(x) := \mathbb{E}_{u\sim\pi, y\sim P(x,u,\cdot)}[R(x,u,y) - g^\pi + V^\pi_{\text{avg}}(y)]$ and $Q^\pi_{\text{avg}}(x,u) := \mathbb{E}_{y\sim P(x,u,\cdot)}\left[R(x,u,y) - g^\pi + V^\pi_{\text{avg}}(y)\right]$. The optimal (action) value functions (which exists for ergodic MDPs) satisfy $V^*_{\text{avg}}(x) = \max_{u\in\mathcal{U}} \mathbb{E}_{y\sim P(x,u,\cdot)}\left[R(x,u,y) - g^* + V^*_{\text{avg}}(y)\right]$ and $Q^*_{\text{avg}}(x,u) = \mathbb{E}_{y\sim P(x,u,\cdot)}\left[R(x,u,y) - g^* + V^*_{\text{avg}}(y)\right]$ where g^* is the optimal reward rate.

2.2 Shannon Entropy and MDPs

For a discrete random variable A with finite support $\mathcal{A}$, Shannon entropy Shannon (1948) is a measure of uncertainty induced by its distribution, and it is defined as $h(A) := -\sum_{a\in\mathcal{A}} \Pr(A = a)\log(\Pr(A = a))$. Shannon entropy measures the amount of information encoded in a random variable: a uniform distribution maximizes entropy (minimal information), and a Dirac distribution minimizes it (maximal information). For an MDP under policy π, we define the conditional entropy (Cover, 1999; Biondi et al., 2014) of X_{T+1} given $X_{0:T}$ as:

$$h^\pi(X_{T+1} \mid X_{0:T}) := -\sum_{y\in\mathcal{X}, \mathbf{x}_T\in\mathcal{X}^T} p\left(X_{T+1} = y, X_{0:T} = \mathbf{x}_T\right)\log(p\left(X_{T+1} = y \mid X_{0:T} = \mathbf{x}_T\right)),$$

and the joint entropy[4] of the path $X_{0:T}$ is

$$h^\pi(X_{0:T}) := h(X_0) + \sum_{t=1}^{T} h(X_t \mid X_{0:T-1}).$$

Definition 1 (Entropy Rate Shannon (1948))**.** *Whenever the limit exists, the entropy rate of an MDP $\mathcal{M}$ under policy π is defined by $\bar{h}^\pi := \lim_{T\to\infty} \frac{1}{T} h(X_{0:T})$.*

The entropy rate represents the rate of *diversity* in the information generated by the induced MC's paths. Smaller entropy rates imply more predictable trajectories of the induced MC.

3 Problem Statement

The problem considered in this work is the following. Consider an unknown MDP $\mathcal{M} = (\mathcal{X}, \mathcal{U}, P, R, \mu_0)$ and let Assumption 1 hold. Further, assume that we can sample transitions $\big(x, u, y, R(x,u,y)\big)$ applying any action $u \in \mathcal{U}$ and letting $\mathcal{M}$ evolve according to $P(x, u, \cdot)$. We want

$$\pi_\star \in \arg\max_{\pi\in\Pi} \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_t^\pi] - k\bar{h}^\pi, \tag{2}$$

where $k > 0$ is a tuneable parameter[5]. In words, we are looking for policies that maximize a tuneable weighted linear combination of the negative entropy $-\bar{h}^\pi$ and a standard expected discounted cumulative reward. As such, *we establish a trade-off, which is tuned via the parameter k, between entropy rate minimization (i.e. predictability) and optimality w.r.t. the cumulative reward.*

[2]The bias can also be written in vector form as $b = (I - P + P^*)^{-1}(I - P^*)R^\pi$ where $R^\pi \in \mathbb{R}^{|\mathcal{X}|}$ is the vector of state rewards, $R^\pi_x = \mathbb{E}_{u\sim\pi}[R(x,u,y)]$. See Chapter 8 in Puterman (2014).

[3]Observe that, for ergodic MDPs, $b^\pi(x) = V^\pi_{\text{avg}}\pi(x)$.

[4]The second equality is obtained by applying the general product rule to the joint probabilities of $\mathcal{Y}_T$.

[5]We choose to cast the problem as a maximization of a linear combination of objectives to allow agents to find efficient trade-offs. This problem could similarly be solved through other (multi-objective) optimization methods Skalse et al. (2022); Hayes et al. (2022). We consider this to be orthogonal to the main point of the work, and leave it as an application-dependent choice.

Proposed approach We first show how the entropy rate $\bar{h}^\pi$ can be treated as an average reward criterion, with the so-called *local entropy* l^π as its corresponding local reward. Then, because l^π is policy-dependent, we introduce a surrogate reward, that solely depends on states and actions and can be learned in-the-loop. We show that deterministic policies minimizing the expected average surrogate reward exist and also minimize the actual entropy rate. Moreover, we prove that, given a learned model of the MDP, we are able to (locally optimally) approximate the value function associated to the entropy rate, via learning the surrogate's value functions. Based on these results, we propose a (model-based[6]) RL algorithm with its maximization objective being the combination of the cumulative reward and an average reward involving the surrogate local entropy.

4 Entropy Rates: Estimation and Learning

Towards writing the entropy rate $\bar{h}^\pi$ as an expected average reward, let us define the *local entropy*, under policy π, for state $x \in \mathcal{X}$ as

$$l^\pi(x) := -\sum_{y \in X} P_\pi(x,y) \log P_\pi(x,y). \tag{3}$$

Now, making use of the Markov property, the entropy rate for an MDP reduces to

$$\begin{aligned} \bar{h}^\pi &= \lim_{T\to\infty} \frac{1}{T}\left(h(X_0) + \sum_{t=1}^{T} h(X_t \mid X_{t-1}) \right) \\ &= \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \mathbb{E}[l^\pi(X_t)] = \mathbb{E}\big[\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} l^\pi(X_t) \big]. \end{aligned} \tag{4}$$

We can treat the local entropy l^π under policy π as a *policy-dependent reward (or cost) function*, since l^π is stationary, history independent and bounded. Thus, $\bar{h}^\pi$ is treated as an expected average reward, with reward function l^π.

4.1 A Surrogate for Local Entropy

In conventional RL settings, one is able to sample rewards (and transitions, e.g. from a simulator). However, here, part of the expected reward to be maximized in equation 2 is the negative entropy $-\bar{h}^\pi$, which, as aforementioned, can be seen as an expected average reward with a state- and policy-dependent local reward $l^\pi(x)$. It is not reasonable to assume that one can directly sample local entropies $l^\pi(x)$: one would have to estimate $l^\pi(x)$ through estimating transition probabilities $P_\pi(x,y)$ (by sampling transitions) and using equation 3. However, a new challenge arises: l^π *depends on the action distribution* and to apply average reward MDP theory we need the rewards to be state-action dependent. To address this, we consider a surrogate for l^π that is policy-independent:

$$s(x,u) = -\sum_{y \in X} P(x,u,y) \log\left(P(x,u,y)\right).$$

Define $\bar{h}^\pi_s(x) := \lim_{T\to\infty} \mathbb{E}[\frac{1}{T}\sum_{t=0}^{T} s(X_t, \pi(X_t)) | X_0 = x]$. The following relationships hold.

Lemma 1. *Consider MDP $\mathcal{M} = (\mathcal{X}, \mathcal{U}, P, R, \mu_0)$ and let Assumption 1 hold. The following statements hold. a) $\mathbb{E}_{u\sim\pi(x)}[s(x,u)] \leq l^\pi(x)$, for all $\pi \in \Pi$. b) $\bar{h}^\pi_s(x) = \bar{h}^\pi_s \leq \bar{h}^\pi$, for some $\bar{h}^\pi_s \in \mathbb{R}$, for all $\pi \in \Pi$. c) $\mathbb{E}_{u\sim\pi(x)}[s(x,u)] = l^\pi(x)$ and $\bar{h}^\pi_s = \bar{h}^\pi$, for all $\pi \in \Pi^D$.*

Proof of Lemma 1. For statement a), observe $l^\pi(x)$ can be expressed as

$$l^\pi(x) = -\mathbb{E}\left[P(x,u,y) | u \sim \pi(x)\right] \log\left(\mathbb{E}\left[P(x,u,y) | u \sim \pi(x)\right] \right),$$

[6]We use the term *model-based*, since we require learning a (approximated) representation of the dynamics of the MDP to estimate the entropy. However, the choice of whether to improve the policy using pure model free algorithms versus using the learned model is left as a design choice, beyond the scope of this work.

and recall

$$\mathbb{E}_{u\sim\pi(x)}[s(x,u)] = -\mathbb{E}_{u\sim\pi(x)}\big[\sum_{y\in X} P(x,u,y)\log\left(P(x,u,y)\right)\big].$$

Then, from Jensen's inequality, $\mathbb{E}_{u\sim\pi(x)}[s(x,u)] \leq l^{\pi}$.

In Statement *b*), the fact that $\bar{h}^{\pi}_{s}(x)$ is constant follows from Theorem 3 by considering $R(x,u,y) \equiv s(x,u)$. Now, note that we can write $\bar{h}^{\pi}_{s} = \sum_{x\in\mathcal{X}} \mathbb{E}_{u\sim\pi(x)}[s(x,u)]\mu^{\pi}(x)$ and $\bar{h}^{\pi} = \sum_{x\in\mathcal{X}} l^{\pi}(x)\mu^{\pi}(x)$ (see Puterman (2014)). Thus:

$$\bar{h}^{\pi}_{s} = \sum_{x\in\mathcal{X}} \mathbb{E}_{u\sim\pi(x)}[s(x,u)]\mu^{\pi}(x) \leq \sum_{x\in\mathcal{X}} l^{\pi}(x)\mu^{\pi}(x) = \bar{h}^{\pi},$$

where we employed Statement *a*).

For Statement *c*), take $\pi \in \Pi^{D}$. Then, $\pi(u' \mid x) = 1$ and $\pi(u \mid x) = 0$ for an action $u' \in \mathcal{U}$ and all $u \neq u'$. Then

$$\mathbb{E}_{u\sim\pi(x)}[s(x,u)] = \sum_{y\in X} P(x,u',y)\log\left(P(x,u',y)\right) = l^{\pi}(x).$$

The fact that $\bar{h}^{\pi}_{s} = \bar{h}^{\pi}$ follows, then, trivially. □

4.2 Minimum Entropy Policies

Based on Lemma 1, we derive one of our main results:

Theorem 1. *Consider MDP $\mathcal{M} = (\mathcal{X}, \mathcal{U}, P, R, \mu_0)$ and let Assumption 1. The following hold: a) There exists a deterministic policy $\hat{\pi} \in \Pi^{D}$ minimizing the surrogate entropy rate , i.e. $\hat{\pi} \in \arg\min_{\pi} \bar{h}^{\pi}_{s}$. b) Any $\hat{\pi} \in \Pi^{D}$ minimizing the surrogate entropy rate also minimizes the true entropy rate: $\hat{\pi} \in \arg\min_{\pi\in\Pi} \bar{h}^{\pi}_{s}$ and $\hat{\pi} \in \Pi^{D} \implies \hat{\pi} \in \arg\min_{\pi\in\Pi} \bar{h}^{\pi}$. Additionally, deterministic policies locally minimizing $\bar{h}^{\pi}_{s}$ also locally minimize $\bar{h}^{\pi}$. c) There exists a deterministic policy $\hat{\pi} \in \Pi^{D}$ such that $\hat{\pi} \in \arg\min_{\pi} \bar{h}^{\pi}$.*

Proof of Theorem 1. The first statement follows directly from Theorem 3, which guarantees that there is at least one deterministic policy $\hat{\pi}$ that minimizes the surrogate entropy rate $\bar{h}^{\pi}_{s}$. Then, since $\hat{\pi} \in \Pi^{D}$, from Lemma 1 statements *b*) and *c*), we have that the following holds for all $\pi \in \Pi$:

$$\bar{h}^{\hat{\pi}} = \bar{h}^{\hat{\pi}} \leq \bar{h}^{\pi}_{s} \leq \bar{h}^{\pi}$$

Thus, $\hat{\pi}$ minimizes $\bar{h}^{\pi}$ and it follows that $\hat{\pi} \in \arg\min_{\pi\in\Pi} \bar{h}^{\pi}_{s}$ and $\hat{\pi} \in \Pi^{D} \implies \hat{\pi} \in \arg\min_{\pi\in\Pi} \bar{h}^{\pi}$. The same argument also applies locally, thereby yileding that deterministic local minimizers of $\bar{h}^{\pi}_{s}$ are also local minimizers of $\bar{h}^{\pi}$. Finally, the third statement follows as a combination of the other two. □

Theorem 1 is an utterly relevant result for our work. First, it guarantees that minimizing policies both for $\bar{h}^{\pi}_{s}$ and $\bar{h}^{\pi}$ exist. More importantly, it tells us that, *to minimize the entropy rate of an RL agent, it is sufficient to minimize the surrogate entropy rate.* Since (globally) minimizing $\bar{h}^{\pi}_{s}$ implies minimizing $\bar{h}^{\pi}$ and since s is policy-independent, in contrast to l^{π}, in what follows, our RL algorithm uses estimates of s to minimize $\bar{h}^{\pi}_{s}$, instead of estimates of l^{π} to minimize $\bar{h}^{\pi}$.[7]

5 Learning to Act Predictably

In the following, we show how predictability of the agent's behavior can be cast as an RL objective and combined with a primary discounted reward goal. To do this, we rely on Theorem 1 and employ the surrogate entropy $s(x,u)$ as a local reward along with its corresponding value function. We prove that, given a learned model of the MDP, we are able to approximate the true entropy rate value functions. In the next section, we combine this section's results with conventional discounted rewards and standard PG results, to

[7]Observe that we cannot employ the same method for entropy rate maximization, since the maximizer of $\bar{h}^{\pi}_{s}$ is not necessarily a maximizer of $\bar{h}^{\pi}$.

address the problem mentioned in the Problem Statement and derive a PG algorithm that maximizes the combined reward objective. We define the predictability objective to be minimized:

$$J_s(\pi) \equiv \bar{h}_s^\pi = \mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T} s\big(X_t, \pi(X_t)\big)\right].$$

Motivated by Theorem 1, we have employed the surrogate entropy as a local reward and consider the corresponding average-reward problem. As commonly done in average reward problems, we define the (surrogate) *entropy value function* for a policy π, $W^\pi : \mathcal{X} \to \mathbb{R}$ to be equal to the bias, i.e.:

$$\begin{aligned} W^\pi(x) :=& \mathbb{E}\left[\sum_{t=0}^{\infty} s\big(X_t, \pi(X_t)\big) - \bar{h}_s^\pi \mid X_0 = x\right] = \\ =& \mathbb{E}_{y\sim P_\pi(x,\cdot)}\left[s\big(x, \pi(x)\big) - \bar{h}_s^\pi + W^\pi(y)\right], \end{aligned} \tag{5}$$

Additionally, we define the (surrogate) entropy action-value function $S^\pi : \mathcal{X}\times\mathcal{U} \to \mathbb{R}$ by $S^\pi(x,u) := \mathbb{E}_{y\sim P_\pi(x,u,\cdot)}\left[s\big(x,u\big) - \bar{h}_s^\pi + W^\pi(y)\right]$. However, recall that *we do not know* the local reward s. To estimate s, one needs to have an estimate of the transition function P of the MDP. We use

$$s_\phi(x,u) = -\sum_{y\in\mathcal{X}} P_\phi(x,u,y)\log\big(P_\phi(x,u,y)\big)$$

(and $\bar{h}_{s_\phi}^\pi$ correspondingly, for its associated rate) to denote the – parameterised by ϕ – estimate of s, which results from a corresponding estimate P_ϕ of P (i.e. P_ϕ is the learned model). Similarly, we will use J_{s_ϕ}, W_ϕ^π and S_ϕ^π to denote value functions computed with the model estimates s_ϕ. Now, it is crucial to know that by using the model estimates s_ϕ we are still able to approximate well the objective J_s and the value functions W^π, S^π. Let us first show that for a small error between P_ϕ and P (i.e. small modeling error), the error between s_ϕ and s and the objectives $J_s(\pi) \equiv \bar{h}_s^\pi$ and $J_{s_\phi}(\pi) \equiv \bar{h}_{s_\phi}^\pi$ is also small.

Proposition 1. *Consider MDP $\mathcal{M} = (\mathcal{X}, \mathcal{U}, P, R, \mu_0)$ and let Assumption 1 hold. Consider $P_\phi : \mathcal{X}\times\mathcal{U}\times\mathcal{X} \to [0,1]$, parameterised by $\phi \in \Phi$. Assume that the total variation error between P_ϕ and P is bounded as, $\forall$ $x \in \mathcal{X}$ and $u \in \mathcal{U}$, $\max_{x\in\mathcal{X},u\in\mathcal{U}} D_{TV}\big(P_\phi(x,u,\cdot)\|P(x,u,\cdot)\big) \le \epsilon$, for some ϵ, with $0 \le \epsilon \le 1$. Then, $\|s_\phi(x,u) - s(x,u)\|_\infty \le K(\epsilon)$, and the surrogate entropy rate error for any policy π,*

$$\mathbb{E}\big[\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T} s_\phi(X_t,\pi(X_t)) \mid X_0 \sim \mu_0\big] - \bar{h}_s^\pi \le K(\epsilon),$$

where $K(\epsilon) = \epsilon\log\big(|\mathcal{X}| - 1\big) - \epsilon\log\epsilon - (1-\epsilon)\log(1-\epsilon)$.

Proof of Proposition 1. Observe that $s(x,u)$ and $s_\phi(x,u)$ are the entropies of probability distributions $P(x,u,\cdot)$ and $P_\phi(x,u,\cdot)$, respectively. Thus, from the Fannes–Audenaert inequality Fannes (1973), for the special case of diagonal density matrices (representing conventional probability distributions), we obtain:

$$\|s_\phi(x,u) - s(x,u)\|_\infty \le \epsilon\log\Big(|\mathcal{X}| - 1\Big) - \epsilon\log\epsilon - (1-\epsilon)\log(1-\epsilon) = K(\epsilon),$$

Finally:

$$\begin{aligned} &\mathbb{E}\big[\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T} s_\phi(X_t,\pi(X_t)) \mid X_0 \sim \mu_0\big] - \bar{h}_s^\pi = \\ &\mathbb{E}\big[\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T} s_\phi(X_t,\pi(X_t)) - s(X_t,\pi(X_t))\big] \le \\ &\mathbb{E}\Big[\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T} \big|s_\phi(X_t,\pi(X_t)) - s(X_t,\pi(X_t))\big|\Big] \le \mathbb{E}\big[\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T} K(\epsilon)\big] = K(\epsilon). \end{aligned}$$

□

Observe that as $\epsilon \to 0$, i.e. as the learned model approaches the real one, then the surrogate entropy rate converges to the actual one[8] (since $K(\epsilon) \to 0$). This result indicates that we can indeed use s_ϕ, obtained by a learned model P_ϕ, instead of the unknown s, as the error between the objectives $J_s(\pi) \equiv \bar{h}^\pi_s$ and $J_{s_\phi}(\pi) \equiv \bar{h}^\pi_{s_\phi}$ is small, for small model errors. Assume now, without loss of generality that we have parameterised entropy value function (*critic*) S_ω with parameters $\omega \in \Omega$. We show that a standard on-policy algorithm, with policy π, with value function approximation S_ω, using the approximated model P_ϕ, learns entropy value functions that are in a $\delta(\epsilon)$-neighbourhood of the true entropy value functions S^π, and $\delta(\epsilon)$ vanishes with ϵ.

Assumption 2. *Any learning rate* $\alpha_t \in (0,1)$ *satisfies* $\sum_{t=0}^{\infty} \alpha_t = \infty$, $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$.

Assumption 3. *The model* P_ϕ *satisfies* $\max_{x\in\mathcal{X},u\in\mathcal{U}} D_{TV}(P_\phi(x,u,\cdot)\|P(x,u,\cdot)) \leq \epsilon$, *for some (small)* $\epsilon \in [0,1]$.

Proposition 2. *Consider an MDP* $\mathcal{M}$*, a policy* π*, a learned model* P_ϕ *of the MDP and critic* S_ω *linear on* ω*, and* $\omega \in \Omega \subset \mathbb{R}^n$*, where* Ω *is compact. Let Assumptions 1, and 3 hold. At every step* k *of parameter iteration, let us collect* k *trajectories* $\mathcal{T}_k$ *of length* T*, and construct (unbiased) estimates*[9] $\hat{S}^\pi_\phi$*. Let the critic parameters* $\omega \in \Omega$ *be updated as* $\omega_{k+1} = \omega_k - \beta_k \hat{\Delta\omega}_k$*, with* $\omega_0 \in \Omega$*,* β_k *being a learning rate satisfying Assumption 2, and*

$$\hat{\Delta\omega}_k = \left(\hat{S}^\pi_\phi(x_k,u_k) - S_\omega(x_k,u_k)\right)\frac{\partial S_\omega(x_k,u_k)}{\partial\omega}.$$

Then, ω *converge to a* $\delta(\epsilon)$*-neighbourhood of one of the (local) minimizers of* $\mathbb{E}_{\substack{x\sim\mu^\pi \\ u\sim\pi_\theta(x)}}\left[\frac{1}{2}\left(S^\pi(x,u) - S_\omega(x,u)\right)^2\right]$*, where* $\delta(\epsilon)$ *is vanishing with* ϵ*.*

Proof of Proposition 2. Since S_ω is linear on $\omega \in \Omega$ and Ω is compact, there exists at least one minimizer ω^*. Now, from equation 1 and Theorem 8.2.6 (Puterman, 2014), S^π and S^π_ϕ can be written in vector form (over the states $\mathcal{X}$) as:

$$S^\pi = (I - P_\pi + P^*_\pi)^{-1}(I - P^*_\pi)s^\pi,$$
$$S^\pi_\phi = (I - P_\pi + P^*_\pi)^{-1}(I - P^*_\pi)s^\pi_\phi,$$

where s^π is the vector representation of $s(\cdot,\pi(\cdot))$ (and analogously for s^π_ϕ). Therefore, from Corollary 1, $\|S^\pi(x) - S^\pi_\phi(x)\|_\infty \leq K(\epsilon)$. Then, we can write without loss of generality

$$S^\pi(x,u) = S^\pi_\phi(x,u) + \eta(\epsilon),$$

with $\eta(\epsilon)$ being $O(\epsilon)$. Finally, we can write the parameter iteration as

$$\omega_{t+1} = \omega_t + \beta_t\big[-\nabla_\omega L^\pi_\omega + M_{t+1} + \eta(\epsilon)\big],$$

with $L^\pi_\omega := \mathbb{E}_{\substack{x\sim\mu^\pi \\ u\sim\pi_\theta(x)}}\left[\frac{1}{2}\left(S^\pi(x,u) - S_\omega(x,u)\right)^2\right]$ and the term $M_{t+1} := \left(\hat{S}^\pi(x,u) - S^\pi(x,u)\right)\frac{\partial S_\omega(x,u)}{\partial\omega}$ is a Martingale with bounded variance (since s is bounded).

Therefore, by Theorem 6 in (Borkar, 2009), the iterates converge to some point $\omega_t \to \Omega^*_\delta(\pi_\theta)$ *almost surely* as $t \to \infty$, with $\Omega^*_\delta(\pi_\theta)$ being the $O(\delta)$ neighbourhood of the stationary points satisfying $\nabla_\omega L^\pi_\omega = 0$. $\square$

In other words, for small model errors, the value function approximator converges to a locally optimal value function approximation of the true value function S^π.

5.1 Predictability-Aware Policy Gradient

Now, we are ready to address the Problem Statement, combining the entropy rate objective with a discounted reward objective. In what follows, assume that we have a parameterised policy π_θ with parameters $\theta \in \Theta$. Let $J(\pi_\theta) = \mathbb{E}[\sum_{t=0}^{\infty}\gamma^t R^{\pi_\theta}_t]$. We use Q_ξ with parameters $\xi \in \Xi$ for the parameterised critic of the discounted reward objective (when using a form of actor-critic algorithm).

[8] This result echoes the Simulation Lemma in Kearns & Singh (2002), but with a bound derived in infinite horizon by using the entropy properties.

[9] Via *e.g.* TD(0) value estimation Sutton & Barto (2018).

Theorem 2. *Consider an MDP $\mathcal{M}$, parameterised policy π_θ, a learned model P_ϕ of the MDP and (linear) critic S_ω. Let Assumptions 1 and 3 hold. Let a given PG algorithm maximize (locally) the discounted reward objective $J(\pi_\theta) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_t^{\pi_\theta}]$. Let the value function Q_ξ (or V_ξ) be parameterised by $\xi \in \Xi$, and the entropy value function S_ω (or W_ω) have the same parameterisation class. Then, the same PG algorithm with updates*

$$\theta \leftarrow \mathrm{proj}_\Theta \left[\theta + \alpha_t \left(\hat{\nabla}_\theta J(\pi_\theta) - k\hat{\nabla}_\theta J_{s_\phi}(\pi_\theta)\right)\right]$$

converges to a local maximum of the combined objective $J(\pi_\theta) - kJ_{s_\phi}(\pi_\theta)$.

Proof of Theorem 2 (Sketch). By standard PG arguments Sutton et al. (1999), if a PG algorithm converges to a local maximum of the objective $J(\pi_\theta)$ then the updates $\hat{\nabla}_\theta J(\pi_\theta)$ are in the direction of the gradient (up to stochastic approximation noise). By the same arguments, given Proposition 2, the same algorithm converges to a local minimum of the entropy value function W_ω through updates $-\hat{\nabla}_\theta J_s(\pi_\theta)$, and these are in the direction of the true gradient (again, up to stochastic approximation noise). Then, the linear combination of gradient updates $\hat{\nabla}_\theta J(\pi_\theta) - k\hat{\nabla}_\theta J_s(\pi_\theta)$ is in the direction of the gradient of the combined objective $J(\pi_\theta) - kJ_s(\pi_\theta)$. Finally, since both objectives are locally concave (necessary condition following from existence of gradient schemes that locally maximize them), their linear combination is also locally concave. This concludes the proof. □

Remark 1. *Regarding pure entropy rate minimization, i.e. without the discounted reward objective, as already proven by Theorem 1, a policy that is globally optimal for the surrogate entropy rate $J_s(\pi)$ is also optimal for the actual entropy rate $\bar{h}^\pi$. The same holds for locally optimal deterministic policies. However, in general, this is not the case for stochastic local minimizers.*

Following a vanilla policy gradient structure, in Algorithm 1 we first sample a trajectory τ of length T, under a policy π_θ, and store it in a buffer $\mathcal{D}$ (for training the approximate model P_ϕ). Then, we use $\mathcal{D}$ to train P_ϕ; update the estimated entropy rate; compute estimated objective gradients $\hat{\nabla}_\theta J(\pi_\theta)$, $\hat{\nabla}_\theta J_{s_\phi}(\pi_\theta)$ from trajectory τ[10]; and finally update the policy and critics S_ω, Q_ξ.

Remark 2. *If we were to consider average reward MDPs instead of discounted reward MDPs, the formulation of the optimization problem solved in Theorem 2 results in a more natural interpretation when adding entropy rate objectives. See Appendix 7 for details.*

Algorithm 1 Predictability Aware Policy Gradient

Require: P_ϕ, π_θ, critics W_ω, V_ξ
Require: α_t, $k > 0$
  for E epochs **do**
    $\mathcal{D} \leftarrow$ Trajectory τ of length T.
    Train P_ϕ from $\mathcal{D}$.
    $\bar{h}^\pi_{s_\phi} \leftarrow \frac{1}{T}\sum_{x,u\in\tau} s_\phi(x,u)$.
    Compute $\hat{\nabla}_\theta J(\pi_\theta)$, $\hat{\nabla}_\theta J_{s_\phi}(\pi_\theta)$ from τ.
    $\theta \leftarrow \mathrm{proj}_\Theta \left[\theta + \alpha_t \left(\hat{\nabla}_\theta J(\pi_\theta) - k\hat{\nabla}_\theta J_{s_\phi}(\pi_\theta)\right)\right]$
    Update S_ω (and Q_ξ if used)
  end for

5.2 Implementation

Policy Learning We implement our predictability-aware scheme as first, an on-policy version based on an average-reward PPO algorithm Ma et al. (2021); we call this PAPPO. In particular, for every collected trajectory τ, we update the estimate $\bar{h}^\pi_{s_\phi}$ and (surrogate) entropy value function W_ω parameterised by $\omega \in \Omega$ from the collected samples, and we compute *entropy advantages* for all $(x, u, y, s_\phi(x,u))$ in the collected

[10]This can be done through any policy gradient algorithm at choice.

trajectories as: $\hat{A}^{\pi}_{s} = s_\phi(x,u) - \bar{h}^{\pi}_{s_\phi} + W_\omega(y) - W_\omega(x)$. Then we apply the gradient steps as in PPO Schulman et al. (2015; 2017) by clipping the policy updates. Second, we implement it in an off-policy fashion to compare with recent results on information-theoretic RL, based on a Soft Actor-Critic (Haarnoja et al., 2018) implementation; we call this PASAC. The only modification necessary is the learning of a second Q function (S) for the surrogate trajectory entropy, and the policy loss is computed as a weighted sum of Q and S. See Appendix B.1 for details on PASAC.

Model Learning To learn the approximated model P_ϕ, we assume the transitions to follow Gaussian distributions, similarly to Janner et al. (2019).[11] Following the definition of *differential entropy* of a continuous Gaussian distribution, $s_\phi(x,u) = \log(\sigma^2_{xu}) + K$, where $K = \frac{1}{2}\big(\log(2\pi) + 1\big)$. Therefore, we can estimate the entropy directly by the variance output of our model. Furthermore, since we only need to estimate the variance per transition $(x,u) \to y$, it is sufficient to construct a model $f_\phi : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ that approximates the *mean* $f_\phi(x,u) \approx \int yP(x,u,y)dy$, and we do this through minimizing a mean-squared error loss of transition samples in our model. Then, we estimate the entropy of an observed transition as $s_\phi(x,u) = \log(\mathbb{E}_{y\sim P(x,u,\cdot)}[(f_\phi(x,u) - y)^2])$.

Remark 3. *This is an important feature that greatly simplifies the implementation of our approach. Learning models of dynamical systems is notoriously difficult Moerland et al. (2022); Janner et al. (2019) in general RL environments. In our case, to estimate the entropy (and thus the predictability) we learn a model that predicts the mean transition, and we reconstruct the entropy from the differential entropy estimation of the sampled trajectories, assuming the transitions are Gaussian. Another approach would be to* e.g. *use the relative entropy against a uniform distribution, resulting in a similar objective under Gaussian assumptions.*

6 Experiments

We implemented PARL on a set of robotics and autonomous driving tasks, evaluated the obtained rewards and entropy rates, and compared against different baselines. For the MuJoCo hyperparameters, we took pre-tuned values from Raffin et al. (2019). For the experiments using PASAC and explicit comparison against other SAC-based baselines including RPC (Eysenbach et al. 2021), see Appendix B.1. [12]

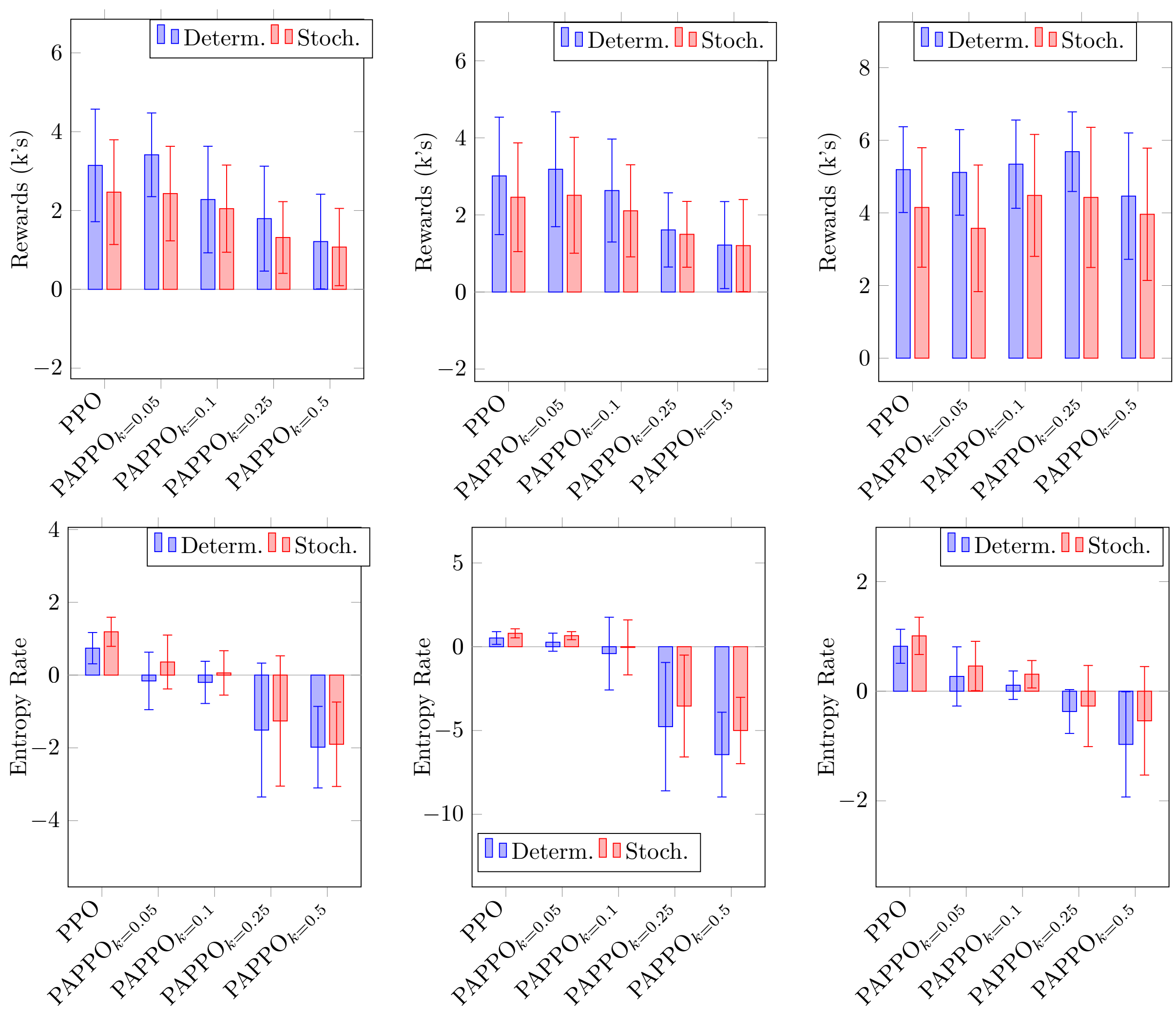
Rewards, Entropies and Ablation To evaluate the influence of the trade-off parameter k, we test PAPPO on MuJoCo tasks and compare to on-policy baselines. We train all agents using the same hyper-parameters, and we only vary the trade-off k in the PARL agents to evaluate the influence. Additionally, we run both the deterministic and stochastic resulting policies (deterministic chooses the mean action that comes out of the policy, stochastic samples from it). The results are reported in Figure 2. Note that, as Mujoco tasks have continuous state and action spaces, entropy rates may be negative.[13]

Trajectory Complexity in MuJoCo The effect of entropy-rate minimization and PAPPO on trajectory distributions is showcased in Figure 3. We evaluated trained agents over 10 full episodes, and plotted the observed trajectories in task space to compare trajectory distributions. We plot as representative variables the z-position and angle of the front tip of the robot. In both cases, we observe that PAPPO policies induce more regular, clustered trajectories, which suggests smaller distributional complexity. Generally, PAPPO trajectories have considerably smaller variance, and visit a smaller portion of the state-space. In the Walker and Ant environments this difference is very pronounced, as PAPPO trajectories are concentrated in a very small region of the state-space, and especially in terms of the angles observed, are limited to a much smaller range.

[11] This is a strong assumption, and in many multi-modal problems, it may not be sufficient to capture the dynamics. Note, however, that our method is compatible with any representation of learned model P_ϕ.

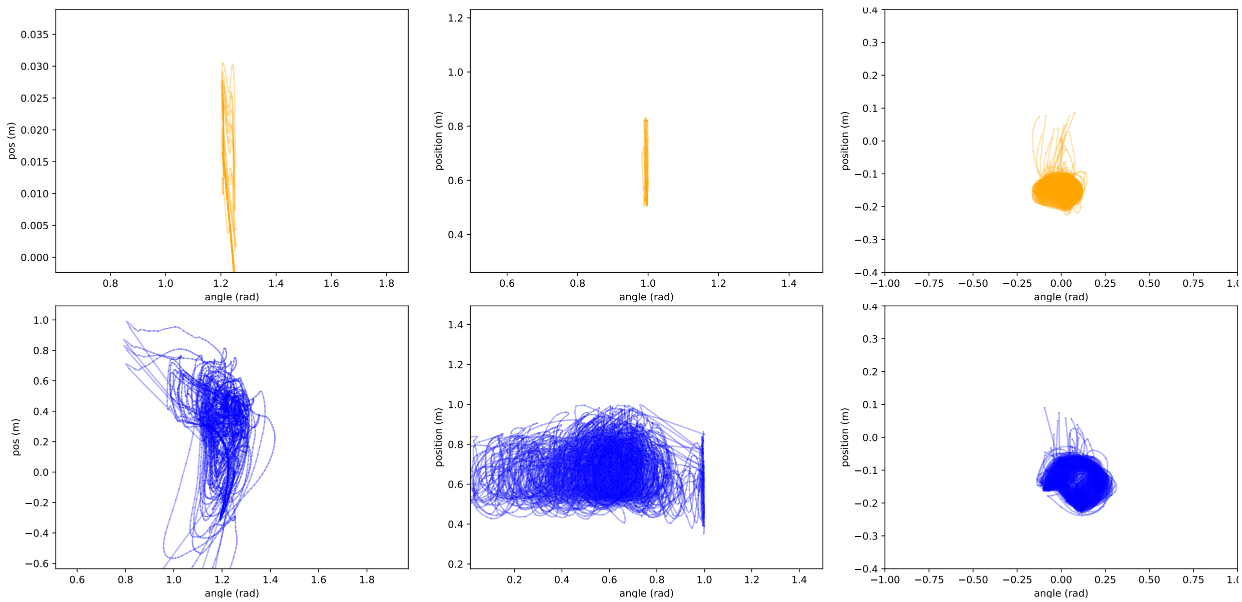
[12] We have also designed two additional representative robotic tasks where agents use PARL to avoid unnecessary stochasticity in the environment. See the Appendix for these.

[13] Recall that, for our implementation, to address continuous state/action tasks, we use *differential entropy* to quantify predictability for continuous random variables, which may indeed be negative. A different metric one may use is *relative entropy* (the KL-divergence from the uniform distribution).



(a) Walker (b) Ant (c) HalfCheetah

Figure 2: Rewards and Entropy rates as a function of k.



(a) Walker (b) Ant (c) HalfCheetah

Figure 3: 10 trajectory plots for agents with seed 0. Blue is PPO, orange is PAPPO with $k = 0.5$.

Table 1: Results for autonomous driving environments.

Highway	Rewards	Ep. Length	Avg. Speed	Entropy Rate
DQN	17.51 ± 7.73	20.92 ± 9.08	**6.51 ± 0.34**	-0.37 ± 0.16
PPO	18.88 ± 6.74	24.41 ± 8.49	5.62 ± 0.10	-0.88 ± 0.05
PAPPO$_{k=0.1}$	**21.05 ± 4.03**	28.56 ± 5.27	5.11 ± 0.05	-1.43 ± 0.09
PAPPO$_{k=0.5}$	21.03 ± 2.05	29.60 ± 2.64	5.06 ± 0.01	-1.51 ± 0.09
PAPPO$_{k=1}$	20.89 ± 2.05	29.61 ± 2.58	5.05 ± 0.00	**-1.51 ± 0.06**

Roundabout	Rewards	Ep. Length	Avg. Speed	Entropy Rate
DQN	22.53 ± 11.93	26.92 ± 14.41	3.09 ± 0.47	-0.55 ± 0.88
PPO	**29.26 ± 10.68**	32.92 ± 11.67	2.80 ± 0.12	-0.23 ± 0.27
PAPPO$_{k=0.1}$	28.86 ± 10.96	32.98 ± 12.23	2.65 ± 0.50	-0.28 ± 0.77
PAPPO$_{k=0.5}$	17.99 ± 13.34	23.45 ± 17.60	3.75 ± 0.05	-2.13 ± 0.12
PAPPO$_{k=1}$	16.83 ± 13.28	22.05 ± 17.67	**3.79 ± 0.04**	**-2.21 ± 0.14**

Predictable Driving We test PAPPO in the Highway Environment (Leurent, 2018), where an agent learns to drive at a desired speed while navigating a crowded road with other autonomous agents. The agent gets rewarded for tracking the desired speed and penalised for collisions. We consider a *highway* and a *roundabout* scenario (see Figure 4). We compare against PPO (Schulman et al., 2017) and DQN Mnih et al. (2015)) agents, and take the hyperparameters directly from Leurent (2018). The results are presented in Table 1. In the highway environment, agents slow down their speed and stop overtaking (arguably, a more predictable driving pattern). This results in longer episode lengths (and larger episodic rewards), but lower rewards per time-step (since reward is given for driving faster). In the roundabout scenario, a different behavior emerges: Agents keep a constant high speed to traverse the roundabout as fast as possible, as the roundabout is the main source of complexity.

7 Average Reward Formulation

Through the work we argue that the entropy rate can be efficiently cast and implemented as an average reward criterion, to be combined with other primary reward objectives. A natural question to ask is how does the formulation of our work adapt to the case where the primary objective is an average reward objective over the MDP rewards.

Consider the case where we are interested in optimizing a linear combination of average reward and entropy rate objectives. Then, we can write the reward objective as $J(\pi) = \mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T} R_t^\pi\right]$, and since both the average rewards and the average entropies are taken in expectation over the same probability space, the trade-off objective in equation 2 becomes

$$\arg\max_{\pi\in\Pi} J(\pi) - kJ_s(\pi) = \mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T} R_t^\pi - ks_t^\pi\right], \tag{6}$$

where we already included the surrogate entropy $s_t^\pi = s(X_t, \pi(X_t))$. Further, recall that $s(x,u) = \mathbb{E}_{y\sim P(x,u,\cdot)}[-\log P(x,u,y)]$. Then, assuming knowledge of P (or an approximation of it), one can define an adjusted reward $\tilde{R}(x,u,y) := R(x,u,y) + k\log P(x,u,y)$ and a modified value function $\tilde{V}^\pi_{\text{avg}}(x) := \mathbb{E}_{u\sim\pi, y\sim P(x,u,\cdot)}[R(x,u,y) + k\log P(x,u,y) - \tilde{g}^\pi + V^\pi_{\text{avg}}(y)]$, where $\tilde{g}^\pi = \mathbb{E}\left[\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T}\tilde{R}_t^\pi\right]$ is the modified reward rate under policy π (which is independent of the initial set of states for ergodic MDPs). Observe that, even though $\tilde{R}(x,u,y) \neq R(x,u,y) - ks(x,u)$, they are equal in expectation. Then, for any fixed policy, the expected average reward can be computed through $\tilde{R}$, and a policy π^* solving the optimization problem:

$$\pi^* \in \arg\max_{\pi\in\Pi} \tilde{V}^\pi_{\text{avg}}(x) \quad \forall x \in \mathcal{X} \tag{7}$$

is guaranteed to solve equation 6. This allows for more compact formulation, and for the learning of a single value function (instead of two), which can be desirable in some cases. It also allows for the following observation: transitions with low probabilities of being observed are penalized. Computationally, this is also advantageous, since we do not need to compute the entropy of the learned model, but instead just evaluate the likelihood of the observed transition and adjust the rewards accordingly. We have implemented a form of average reward PPO with such formulation, and is available in the project repository.

8 Discussion

Summary of Results We proposed a novel method, namely PARL, that induces more predictable behavior in RL agents by maximizing a tuneable linear combination of a standard expected reward and the negative entropy rate, thus trading-off *optimality with predictability.* In the experimental results, we see how PARL greatly reduces the entropy rates of the RL agents while achieving near-optimal rewards, depending on the trade-off parameter. In the autonomous driving setting, agents learn to be more predictable while driving around stochastic agents. In the MuJoCo experiments, PARL obtains policies that yield more clustered, less complex trajectory distributions, allowing models to predict better the dynamics. This results in, for example, a smaller range of values of orientation angles as seen in the trajectory representations on Figure 3. Additionally, following our method, the entropy rate can be directly interpreted as the average complexity necessary to correctly predict the trajectory of the agents, and if assuming Gaussian predictions, this is proportional to the log of the prediction variance observed by the agent's internal prediction model, which shows how lower entropy rates yield predictability.

Shortcomings Our scheme results in a setting where agents maximize a trade-off between two different objectives. This, combined with learning a dynamic model (which is notoriously difficult), introduces implementation challenges related to learning multiple coupled models simultaneously. We make an attempt at discussing a systematic way of addressing these in the Appendix. Additionally, for many applications, avoiding high-entropy policies may restrict the ability of RL agents to learn optimal behaviors (see, e.g., results in Eysenbach et al. 2021). And although in human-robot interaction and human-aligned AI predictability is intuitively beneficial, we cannot claim that minimizing entropy-rates is desirable for all RL applications.

Entropy and Exploration One might wonder if minimizing entropy rates of RL agents may hinder exploration. From a theoretical perspective, this is not a problem due to ergodicity. From a practical perspective, this undesired effect is highly mitigated by the fact that for the first (many) steps, the agent is still learning an adequate model of the dynamics, and therefore the entropy signal is very noisy which favors exploration.

References

J. Abounadi, D. Bertsekas, and V. S. Borkar. Learning Algorithms for Markov Decision Processes with Average Cost. *SIAM Journal on Control and Optimization*, 40(3):681–698, January 2001. ISSN 0363-0129, 1095-7138. doi: 10.1137/S0363012999361974. URL `http://epubs.siam.org/doi/10.1137/S0363012999361974`.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Fabrizio Biondi, Axel Legay, Bo Friis Nielsen, and Andrzej Wasowski. Maximizing entropy over Markov processes. *Journal of Logical and Algebraic Methods in Programming*, 83(5-6):384–399, September 2014. ISSN 23522208. doi: 10.1016/j.jlamp.2014.05.001.

Vivek S Borkar. *Stochastic approximation: a dynamical systems viewpoint*, volume 48. Springer, 2009.

Baptiste Busch, Jonathan Grizou, Manuel Lopes, and Freek Stulp. Learning legible motion from human–robot interactions. *International Journal of Social Robotics*, 9(5):765–779, 2017.

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/f5b1b89d98b7286673128a5fb112cb9a-Paper.pdf`.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

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

Anca D. Dragan, Kenton C.T. Lee, and Siddhartha S. Srinivasa. Legibility and predictability of robot motion. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 301–308, 2013. doi: 10.1109/HRI.2013.6483603.

Xiaoming Duan, Mishel George, and Francesco Bullo. Markov chains with maximum return time entropy for robotic surveillance. *IEEE Transactions on Automatic Control*, 65(1):72–86, 2019.

Richard M Dudley. *Real analysis and probability.* CRC Press, 2018.

Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Robust predictable control. *Advances in Neural Information Processing Systems*, 34:27813–27825, 2021.

Mark Fannes. A continuity property of the entropy density for spin lattice systems. *Communications in Mathematical Physics*, 31:291–294, 1973.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2015.

Mishel George, Saber Jafarpour, and Francesco Bullo. Markov chains with maximum entropy for robotic surveillance. *IEEE Transactions on Automatic Control*, 64(4):1566–1580, 2018.

Hongliang Guo, Qi Kang, Wei-Yun Yau, Marcelo H. Ang, and Daniela Rus. Em-patroller: Entropy maximized multi-robot patrolling with steady state distribution approximation. *IEEE Robotics and Automation Letters*, 8(9):5712–5719, 2023a. doi: 10.1109/LRA.2023.3300245.

Lingxiao Guo, Haoxuan Pan, Xiaoming Duan, and Jianping He. Balancing efficiency and unpredictability in multi-robot patrolling: A marl-based approach. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3504–3509, 2023b. doi: 10.1109/ICRA48891.2023.10160923.

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

Seungyul Han and Youngchul Sung. A max-min entropy framework for reinforcement learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 25732–25745. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/d7b76edf790923bf7177f7ebba5978df-Paper.pdf`.

Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36 (1):26, 2022.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49:209–232, 2002.

Dongyoung Kim, Jinwoo Shin, Pieter Abbeel, and Younggyo Seo. Accelerating reinforcement learning with value-conditional state entropy exploration, 2023.

Edouard Leurent. An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env`, 2018.

Yanyu Liu, Yifeng Zeng, Biyang Ma, Yinghui Pan, Huifan Gao, and Xiaohan Huang. Improvement and evaluation of the policy legibility in reinforcement learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pp. 3044–3046, 2023.

Xingyu Lu, Kimin Lee, Pieter Abbeel, and Stas Tiomkin. Dynamics generalization via information bottleneck in deep reinforcement learning. *arXiv preprint arXiv:2008.00614*, 2020.

Xiaoteng Ma, Xiaohang Tang, Li Xia, Jun Yang, and Qianchuan Zhao. Average-reward reinforcement learning with trust region methods. *arXiv preprint arXiv:2106.03442*, 2021.

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.

Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based Reinforcement Learning: A Survey, March 2022. URL `http://arxiv.org/abs/2006.16712`. arXiv:2006.16712 [cs, stat].

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

Seohong Park and Sergey Levine. Predictable mdp abstraction for unsupervised model-based rl. *arXiv preprint arXiv:2302.03921*, 2023.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pp. 1607–1612, 2010.

Silviu Pitis, Harris Chan, Stephen Zhao, Bradly Stadie, and Jimmy Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7750–7761. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/pitis20a.html`.

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

Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3, 2019.

Yagiz Savas, Melkior Ornik, Murat Cubuktepe, Mustafa O. Karabag, and Ufuk Topcu. Entropy Maximization for Markov Decision Processes Under Temporal Logic Constraints. *IEEE Transactions on Automatic Control*, 65(4):1552–1567, April 2020. ISSN 0018-9286, 1558-2523, 2334-3303. doi: 10.1109/TAC.2019.2922583. URL `https://ieeexplore.ieee.org/document/8735817/`.

Yagiz Savas, Michael Hibbard, Bo Wu, Takashi Tanaka, and Ufuk Topcu. Entropy Maximization for Partially Observable Markov Decision Processes. May 2021. arXiv:2105.07490 [math].

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

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

Rohin Shah, Noah Gundotra, Pieter Abbeel, and Anca Dragan. On the feasibility of learning, rather than assuming, human biases for reward inference. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5670–5679. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/shah19a.html`.

Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3): 379–423, 1948.

Joar Skalse, Lewis Hammond, Charlie Griffin, and Alessandro Abate. Lexicographic multi-objective reinforcement learning. In Lud De Raedt (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 3430–3436. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/476. URL `https://doi.org/10.24963/ijcai.2022/476`. Main Track.

Elis Stefansson and Karl H Johansson. Computing complexity-aware plans using kolmogorov complexity. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 3420–3427. IEEE, 2021.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

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.

Daniil Tiapkin, Denis Belomestny, Daniele Calandriello, Eric Moulines, Remi Munos, Alexey Naumov, Pierre Perrault, Yunhao Tang, Michal Valko, and Pierre Menard. Fast rates for maximum entropy exploration. In *International Conference on Machine Learning*, pp. 34161–34221. PMLR, 2023.

Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

Rui Zhao, Xudong Sun, and Volker Tresp. Maximum entropy-regularized multi-goal reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7553–7562. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/zhao19d.html`.

Alexander Zimin and Gergely Neu. Online learning in episodic markovian decision processes by relative entropy policy search. *Advances in neural information processing systems*, 26, 2013.

A Auxiliary Results

We include in this Appendix some existing results used throughout our work. The following Theorem is a combination of results presented by Puterman (2014) regarding the existence of optimal average reward policies.

Theorem 3 (Average Reward Policies Puterman (2014)). *Given an MDP $\mathcal{M}$ and Assumption 1, the following hold: $g^\pi(x)$ and $b^\pi(x)$ exist, $g^\pi(x) = g^\pi$ for all x (i.e. $g^\pi(x)$ is constant), and there exists a deterministic, stationary policy $\hat{\pi} \in \operatorname{argmax}_{\pi\in\Pi} g^\pi$ that maximizes the expected average reward. Additionally, the same holds if $\mathcal{U}$ is compact and R and P are continuous functions of $\mathcal{U}$.*

Theorem 4 (Stochastic Recursive Inclusions Borkar (2009)). *Let $x_n \in \mathbb{R}^d$ be a vector following a sequence:*

$$x_{n+1} = x_n + a_n(h(x_n) + M_{n+1} + \eta_n), \tag{8}$$

where $\sup_n \|x_n\| < \infty$ a.s., a_n is a learning rate satisfying Assumption 2, $h : \mathbb{R}^d \to \mathbb{R}^d$ is a Lipschitz map, M_n is a Martingale difference sequence with respect to the $\sigma-$algebra $\mathcal{F}_n := \sigma(x_0, M_1, M_2, ..., M_n)$ (and square integrable), and η_n is an error term bounded by ϵ_0. Define $H := \{x \in \mathbb{R}^d : h(x) = 0\}$. Then, for any $\delta > 0$, there exists an $\epsilon > 0$ such that $\forall \epsilon_0 \in (0, \epsilon)$ the sequence $\{x_n\}$ converges almost surely *to a δ-neighbourhood of H.*

Theorem 5 (Policy Gradient with Function Approximation Sutton et al. (1999)). *Let π_θ be a parameterised policy and $f_w : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ be a parameterised (approximation of) action value function in an MDP $\mathcal{M}$. Let the parameterisation* be compatible, i.e. *satisfy:*

$$\frac{\partial f_w(x,u)}{\partial w} = \frac{\partial \pi_\theta(x,u)}{\partial \theta} \frac{1}{\pi_\theta(x,u)}.$$

Let the parameters w and θ be updated at each step such that:

$$w_k : \sum_x \mu^{\pi_\theta}(x) \sum_u \pi_\theta(x,u)\big(Q^{\pi_\theta}(x,u) - f_{w_k}(x,u)\big)\frac{\partial f_w(x,u)}{\partial w_k} = 0,$$
$$\theta_{k+1} = \theta_k + \alpha_t \sum_x \mu^{\pi_\theta}(x) \sum_u \frac{\partial \pi_\theta(x,u)}{\partial \theta} f_{w_k}(x,u).$$

Then, $\lim_{k\to\infty} \frac{\partial \rho(\theta_k)}{\partial \theta} = 0$, where $\rho(\theta)$ is either the discounted or average reward in the MDP.

B Experimental Results and Methodology

We present here the extended experimental results, training curves and additional details corresponding to the experimental framework. All experiments were run in a single CPU, running Ubuntu 20.04, and all libraries and requirements are properly listed in the paper code.

B.1 Soft Actor-Critic Experiments

As discussed in Section 5.2, we implemented a version of PARL based on a Soft Actor-Critic implementation (Haarnoja et al., 2018). For this, we simply learn in parallel an average reward Q function to optimize the entropy rate (S_ω), and combine the Q functions to compute the actor objective as:

$$\begin{aligned} J_{sac}(\pi) = \mathbb{E}_{x\sim\mathcal{D}}\big[\mathbb{E}_{u\sim\pi}\big[\alpha \log \pi(u \mid x) - Q_\xi(x,u) + \\ + kS_\omega(x,u)\big]\big]. \end{aligned} \tag{9}$$

As a baseline for our SAC implementation, we use RPC (Eysenbach et al., 2021), which is another SAC based algorithm aiming to maximize rewards while compressing policies to a maximum complexity, to achieve more simple, robust and predictable behaviours. Please note, the comparison is merely *qualitative*: RPC does not optimize for entropy rate in the agent's behaviour, however simpler policies do induce smaller entropy rates

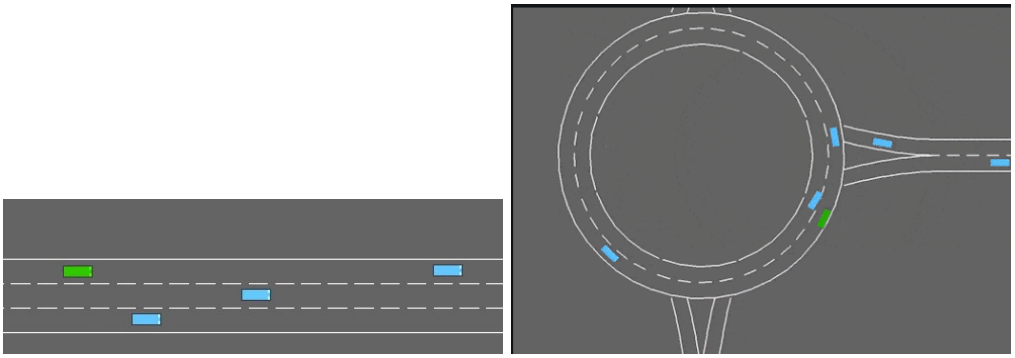


Figure 4: Self-Driving Environments.

(as seen in the reported results) and thus it is still useful as a baseline to evaluate the impact of entropy rate objectives in SAC agents. We trained all agents with the same parameters, 5 agents per parameter combination and evaluated over 50 independent episodes. For RPC, we trained agents with different policy compression rates (2 bits and 5 bits), and for PASAC with two different trade-off parameters ($k = 1$ and $k = 0.5$). We train both PASAC and RPC with the same architecture for the prediction models (*decoder* in RPC), and we use an identity encoder for RPC to make the prediction models equivalent, and the entropy to be estimated in task space (not in a latent space). The results are presented in table 2

Table 2: Results for MuJoCo environments, SAC-based algorithms.

Ant	Rewards	Ep. Length	Entropy Rate
$\text{PASAC}_{k=1}$	**6173.43 ± 442.87**	997.12 ± 48.45	**-0.70 ± 0.12**
$\text{PASAC}_{k=0.5}$	5235.99 ± 1268.48	957.72 ± 168.55	-0.27 ± 0.44
RPC_{2bit}	2084.51 ± 674.39	983.44 ± 108.75	1.81 ± 0.56
RPC_{5bit}	5640.18 ± 496.67	996.15 ± 60.78	1.92 ± 0.12

HalfCheetah	Rewards	Ep. Length	Entropy Rate
$\text{PASAC}_{k=1}$	**11134.25 ± 1398.77**	1000.00 ± 0.0	**-0.02 ± 0.21**
$\text{PASAC}_{k=0.5}$	11014.73 ± 816.36	1000.00 ± 0.0	0.07 ± 0.22
RPC_{2bit}	5105.49 ± 470.98	1000.00 ± 0.0	2.94 ± 0.12
RPC_{5bit}	6003.90 ± 666.42	1000.00 ± 0.0	**2.55 ± 0.17**

Hopper	Rewards	Ep. Length	Entropy Rate
$\text{PASAC}_{k=1}$	1556.92 ± 946.95	479.17 ± 300.50	**-4.28 ± 0.30**
$\text{PASAC}_{k=0.5}$	2645.24 ± 1140.11	739.70 ± 332.04	-3.69 ± 0.24
RPC_{2bit}	**2667.34 ± 1091.18**	732.44 ± 319.22	0.98 ± 0.15
RPC_{5bit}	2456.70 ± 1236.72	655.78 ± 339.75	1.19 ± 0.09

B.2 Interactive Robot Tasks

We created two tasks inspired by real human-robot use-cases, where it is beneficial for agents to avoid high entropy state-space regions. These are based on Minigrid environments Chevalier-Boisvert et al. (2023), see Figure 5. The agents can execute actions $\mathcal{U} = \{\text{forward}, \text{turn-left}, \text{turn-right}, \text{toggle}\}$, and the observation space is $\mathcal{X} = \mathbb{R}^n$ such that the observation includes the robot's position and orientation, the position of obstacles and the state of environment features (*e.g.* the switch). In both tasks, the agent gets a reward of 1 for reaching the goal, or a negative reward of -1 for colliding or falling in the lava. Both grid environments were wrapped in a normalizing vectorized wrapper, to normalize observations, but the model data was kept un-normalized.

Task 1: Turning Off Obstacles This task is designed as a dynamic obstacle navigation task, where the motion of the obstacles can be stopped (the obstacles can be switched off) by the agent toggling a switch at a small cost of rewards. The environment is depicted in Figure 5 (left). The switch is to the left of the agent, shown as a green bar (orange if off). The agent gets a reward of $r = 0.95$ if it turns the obstacle off. The intuition behind this task is that agents do not learn to turn off the obstacles, and attempt to navigate the environment. This, however, induces less predictable dynamics since the obstacle keeps adding noise to the

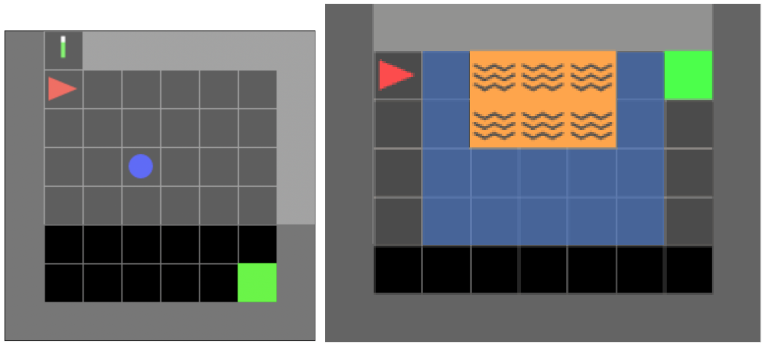
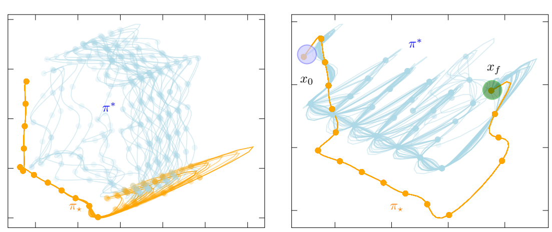


Figure 5: Interactive Tasks. Left is a moving obstacle navigation task where the agent has the option to de-activate the obstacles. Right is a navigation problem where the ground is slippery, resulting in random motion.

observation, and the agent is forced to take high variance trajectories to avoid it. Our Predictability-Aware algorithm converges to policies that consistently disable the stochasticity of the obstacles, navigating the environment freely afterwards, while staying near-optimal.

Task 2: Slippery Navigation The second task is inspired by a cliff navigation environment, where a large portion of the ground is slippery, but a path around it is not. In this problem, the slippery part has uncertain transitions (i.e. a given action does not yield always the same result, because the robot might slip), but the non-slippery path induces deterministic behavior (the robot follows the direction dictated by the action). The agent needs to navigate to the green square avoiding the lava. If it enters the *slippery* region, it has a probability of $p = 0.35$ of spinning and changing direction randomly. The intuition behind this environment is that PPO agents do not learn to avoid the slippery regions, resulting in higher entropy rates and less predictable behaviors. On the contrary, PAPPO agents consistently avoid the slippery regions. This can be seen in Figure 6.



(a) Dyn Obstacle Env. (b) Slippery Nav. Env.

Figure 6: 2D trajectory projections. Blue is a PPO policy, orange is a PAPPO policy.

B.3 MuJoCo Experiments

We include here the full set of trajectory plots in task space, both for z position of the front tip versus angle of the front tip, and x-y velocities of the front tip (See Figures 7, 8, 9, 10, 11 and 12). Additionally, we include in tables 4 and 3 the full numerical results for the simulated agents. Each result is reported computed for 10 independently trained agents, and each agent evaluated over 50 independent trajectories.

Trajectory Representations As expected, the observed trajectories for the case of PARL agents present a much less complex (lower entropy) distribution. In particular, for the slippery navigation task where agents have the choice of taking fully deterministic paths, it is even more obvious that the PARL agent chooses to execute the same trajectory over and over, where PPO agents result in a more complex distribution due to the traversing of the stochastic regions.

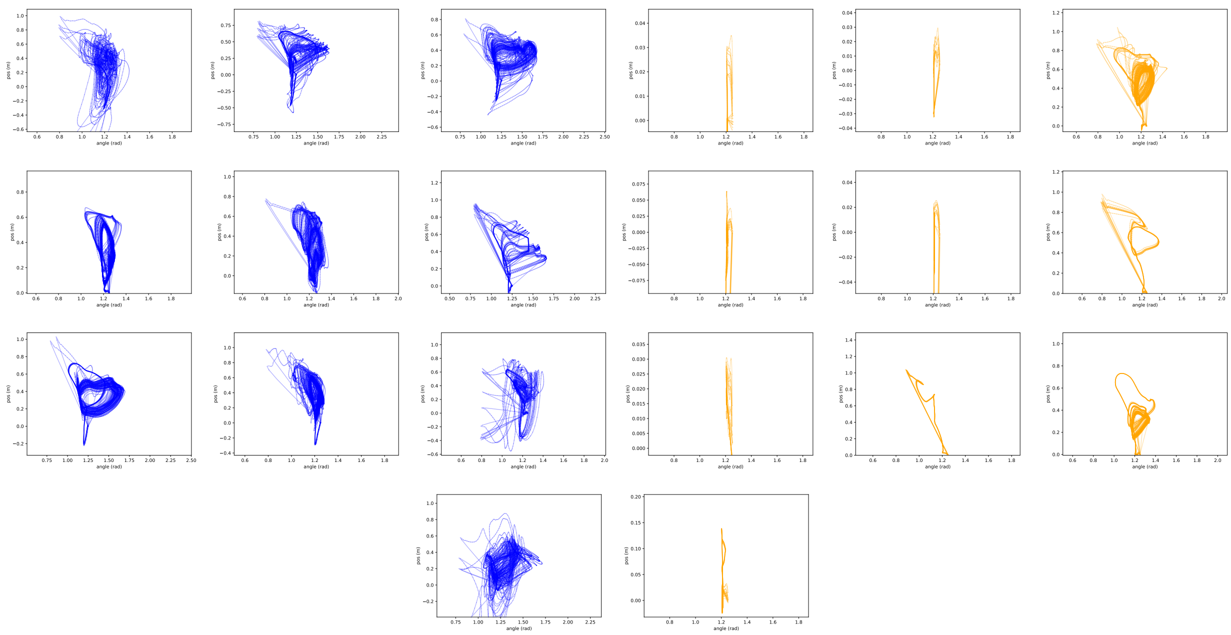


Figure 7: Walker Trajectory Plots, x-axis is torso angle in radians, y-axis is z coordinate position of the torso. Blue are PPO agents, orange are PARL agents.

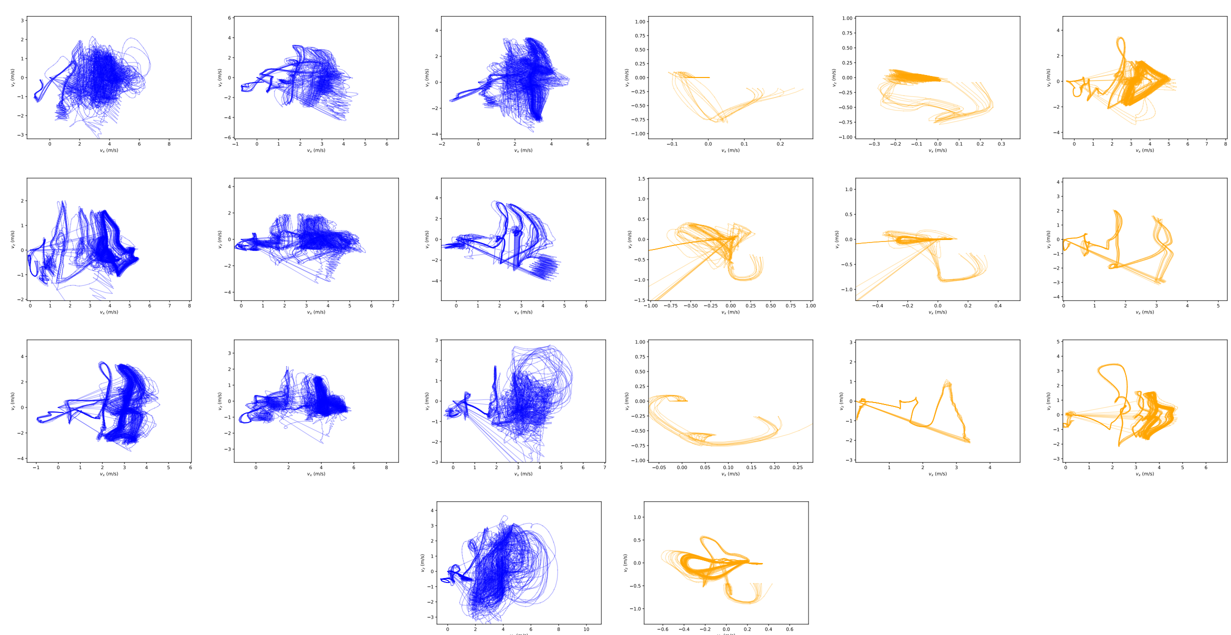


Figure 8: Walker Trajectory Plots, x axis is x velocity, y axis is y velocity. Blue are PPO agents, orange are PARL agents.

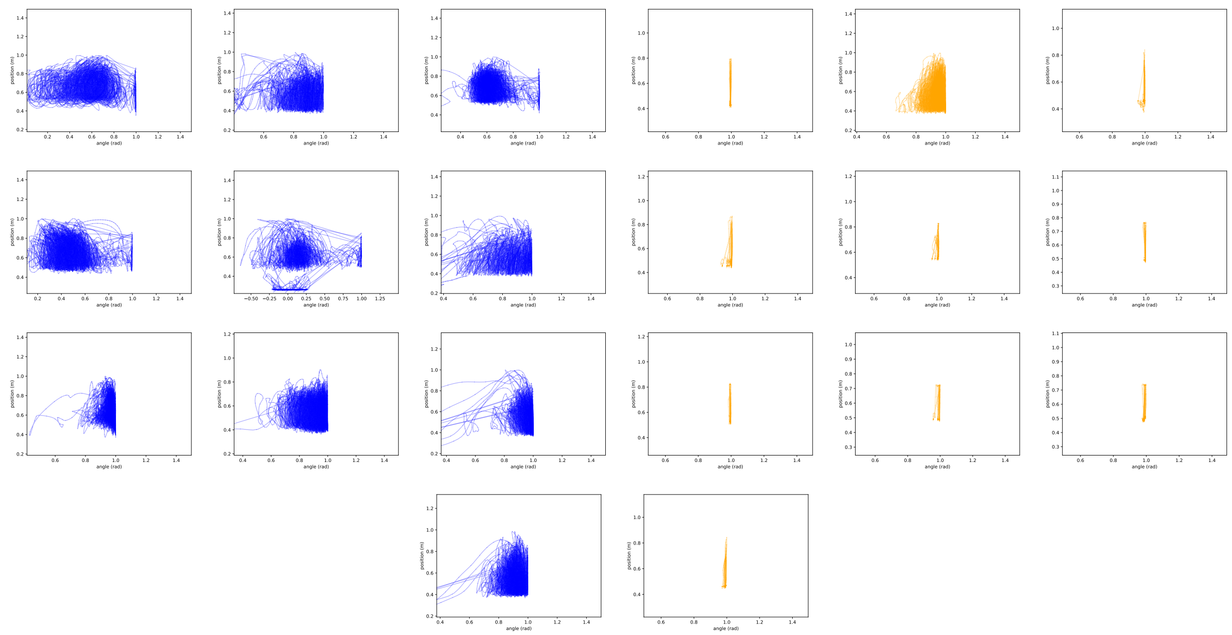


Figure 9: Ant Trajectory Plots, x-axis is torso angle in radians, y-axis is z coordinate position of the torso. Blue are PPO agents, orange are PARL agents.

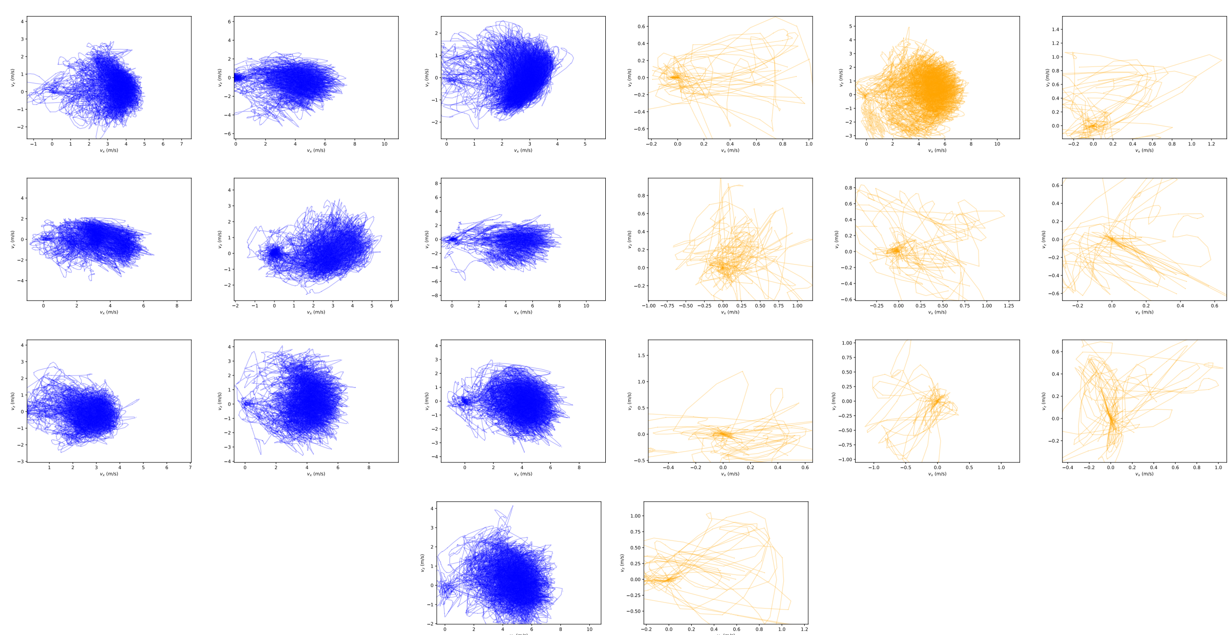


Figure 10: Ant Trajectory Plots, x axis is x velocity, y axis is y velocity. Blue are PPO agents, orange are PARL agents.

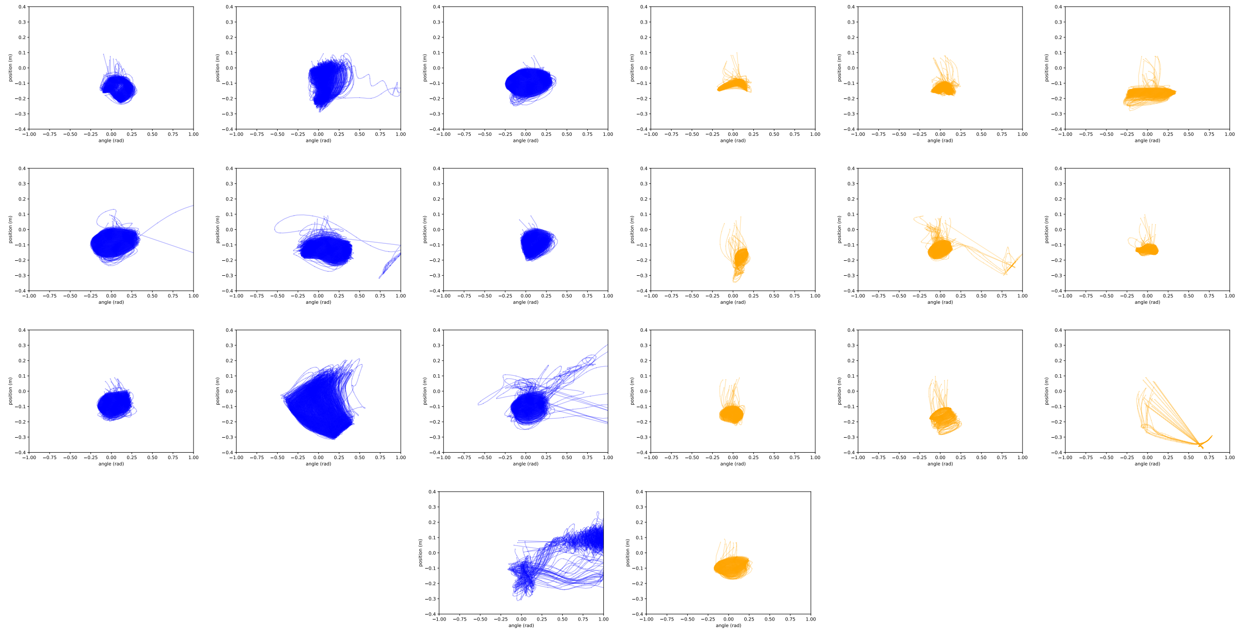


Figure 11: HalfCheetah Trajectory Plots, x-axis is torso angle in radians, y-axis is z coordinate position of the torso. Blue are PPO agents, orange are PARL agents.

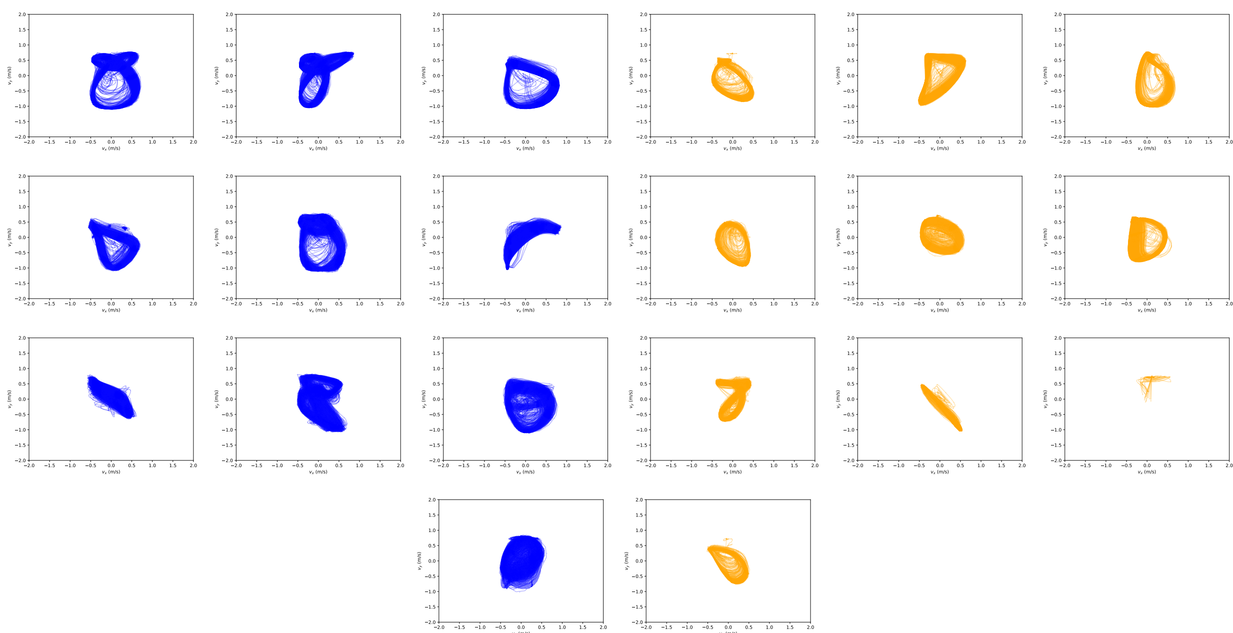


Figure 12: HalfCheetah Trajectory Plots, x axis is x velocity, y axis is y velocity. Blue are PPO agents, orange are PARL agents.

Table 3: Results for MuJoCo environments, Stochastic Policies.

Walker	Rewards	Ep. Length	Entropy Rate
PPO	**2466.27 ± 1329.71**	638.29 ± 296.82	1.19 ± 0.40
$\text{PAPPO}_{k=0.05}$	2430.12 ± 1199.45	626.77 ± 263.34	0.36 ± 0.74
$\text{PAPPO}_{k=0.1}$	2047.30 ± 1106.09	551.40 ± 252.01	0.06 ± 0.61
$\text{PAPPO}_{k=0.25}$	1315.47 ± 910.03	550.18 ± 296.42	-1.26 ± 1.79
$\text{PAPPO}_{k=0.5}$	1072.88 ± 980.41	648.38 ± 358.74	**-1.90 ± 1.16**

Ant	Rewards	Ep. Length	Entropy Rate
PPO	2459.29 ± 1411.75	734.79 ± 342.21	0.80 ± 0.27
$\text{PAPPO}_{k=0.05}$	**2510.29 ± 1504.11**	774.54 ± 343.79	0.66 ± 0.24
$\text{PAPPO}_{k=0.1}$	2107.21 ± 1195.23	786.88 ± 330.96	-0.04 ± 1.64
$\text{PAPPO}_{k=0.25}$	1497.67 ± 854.79	950.96 ± 186.39	-3.54 ± 3.04
$\text{PAPPO}_{k=0.5}$	1204.47 ± 1076.30	987.53 ± 87.62	**-5.00 ± 1.98**

HalfCheetah	Rewards	Ep. Length	Entropy Rate
PPO	4150.53 ± 1645.16	1000.0 ± 0.0	1.01 ± 0.34
$\text{PAPPO}_{k=0.05}$	3575.56 ± 1743.91	1000.0 ± 0.0	0.46 ± 0.45
$\text{PAPPO}_{k=0.1}$	**4482.48 ± 1679.52**	1000.0 ± 0.0	0.31 ± 0.25
$\text{PAPPO}_{k=0.25}$	4427.07 ± 1930.60	1000.0 ± 0.0	-0.27 ± 0.74
$\text{PAPPO}_{k=0.5}$	3962.70 ± 1822.80	1000.0 ± 0.0	**-0.54 ± 0.99**

Table 4: Results for MuJoCo environments, Deterministic policies.

Walker	Rewards	Ep. Length	Entropy Rate
PPO	3144.66 ± 1428.29	769.80 ± 297.99	0.74 ± 0.43
$\text{PAPPO}_{k=0.05}$	**3413.70 ± 1062.22**	848.21 ± 218.93	-0.16 ± 0.79
$\text{PAPPO}_{k=0.1}$	2278.67 ± 1352.92	598.13 ± 301.35	-0.20 ± 0.58
$\text{PAPPO}_{k=0.25}$	1794.17 ± 1331.01	662.40 ± 319.09	-1.51 ± 1.84
$\text{PAPPO}_{k=0.5}$	1212.20 ± 1201.21	672.42 ± 368.88	**-1.98 ± 1.12**

Ant	Rewards	Ep. Length	Entropy Rate
PPO	3013.54 ± 1524.71	808.31 ± 313.55	0.52 ± 0.38
$\text{PAPPO}_{k=0.05}$	**3185.06 ± 1490.56**	883.68 ± 260.07	0.27 ± 0.54
$\text{PAPPO}_{k=0.1}$	2633.50 ± 1336.18	829.38 ± 301.95	-0.41 ± 2.17
$\text{PAPPO}_{k=0.25}$	1611.57 ± 963.54	964.98 ± 155.04	-4.77 ± 3.83
$\text{PAPPO}_{k=0.5}$	1218.27 ± 1129.12	986.36 ± 99.72	**-6.44 ± 2.53**

HalfCheetah	Rewards	Ep. Length	Entropy Rate
PPO	5192.49 ± 1181.34	1000.0 ± 0.0	0.82 ± 0.31
$\text{PAPPO}_{k=0.05}$	5115.65 ± 1178.23	1000.0 ± 0.0	0.27 ± 0.54
$\text{PAPPO}_{k=0.1}$	5342.42 ± 1215.20	1000.0 ± 0.0	0.11 ± 0.26
$\text{PAPPO}_{k=0.25}$	**5686.88 ± 1096.06**	1000.0 ± 0.0	-0.37 ± 0.40
$\text{PAPPO}_{k=0.5}$	4462.63 ± 1740.05	1000.0 ± 0.0	**-0.97 ± 0.96**

B.4 Learning Results

We include the learning curves for the trained agents on all the environments included in the paper.

B.5 Model Learning

Our proposed predictable RL scheme consists of a model-based architecture where the agent learns simultaneously a model P_ϕ for the transition function and a policy π_θ and value functions V_ξ, W_ω for the discounted rewards and entropy rates. Simultaneously to a policy and a value function, we learn a model P_ϕ to approximate the transitions (means) in the environment. For this, we train a neural network with inputs $(x, u) \in \mathcal{X} \times \mathcal{U}$ and outputs the mean next state $\bar{y}_{xu}$. The model is trained using the MSE loss for stored data $\mathcal{D} = \{(x, u, y)\}$:

$$\mathcal{L}_y = \frac{1}{2|\mathcal{D}|} \sum_{\mathcal{D}} \left(\bar{y}_{xu} - y\right)^2.$$

We do this by considering $\mathcal{D}$ to be a replay buffer (to reduce bias towards current policy parameters), and at each iteration we perform K mini-batch updates of the model sampling uniformly from the buffer. Additionally, we pre-train the model a set number of steps before beginning to update the agents, by running a fixed number of environment steps with a randomly initialised policy, and training the model on this preliminary data. All models are implemented as feed-forward networks with ReLU activations.

Entropy estimation We found that it is more numerically stable to use the variance estimations as the surrogate entropy (we do this since the log function is monotonically increasing, and thus maximizing the

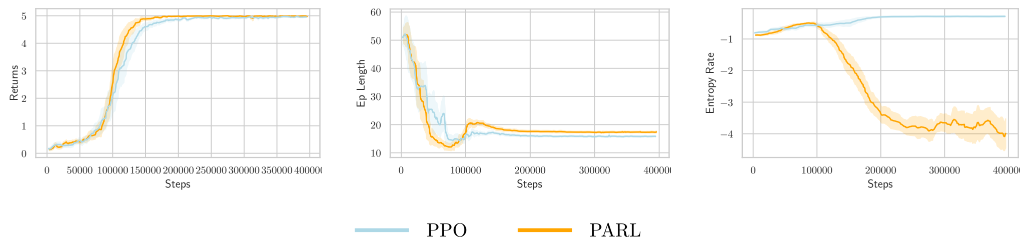


Figure 13: Training results for Slippery Navigation task.

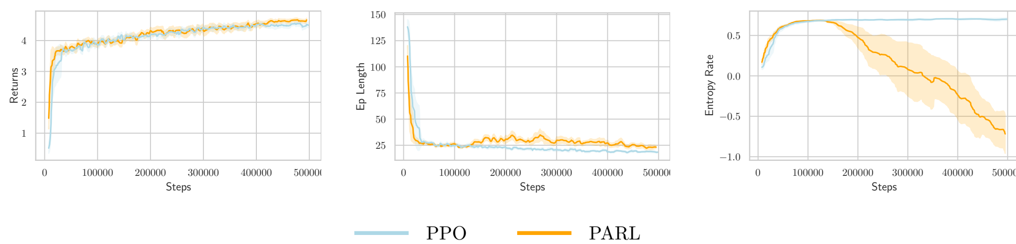


Figure 14: Training results for Obstacle Navigation task.

variance maximizes the logarithm of the variance). This prevented entropy values to explode for environments where some of the transitions are deterministic, thus yielding very large (negative) entropies.

B.6 Tuning and Hyperparamters

The tuning of PARL, due to its modular structure, can be done through the following steps:

1. Tune (adequate) parameters for vanilla RL algorithm used (e.g. PPO).
2. Without the predictable objectives, tune the model learning parameters using the vanilla hyperparameters.
3. Freezing both agent and model parameters, tune the trade-off parameter k and specific PARL parameters (e.g. entropy value function updates) to desired behaviors.

For our experiments, we took PPO and SAC parameters tuned from Stable-Baselines3 Raffin et al. (2019) and Haarnoja et al. (2018), and used automatic hyperparameter tuning Akiba et al. (2019) for model and predictability parameters. In the implementation we introduced a delay parameter to allow agents to start optimizing the policy for some steps without minimizing the entropy rate. For all hyperparameters used in every environment and implementation details we refer the reader to the supplementary material.

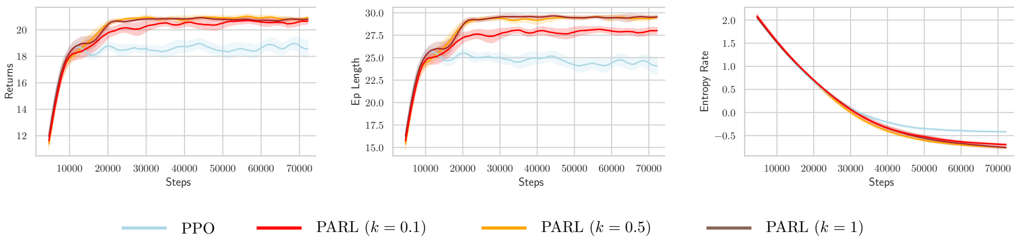


Figure 15: Training results for Highway environment.

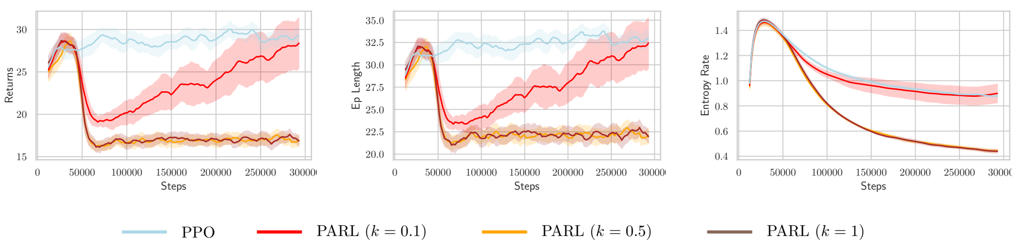


Figure 16: Training results for Roundabout environment.

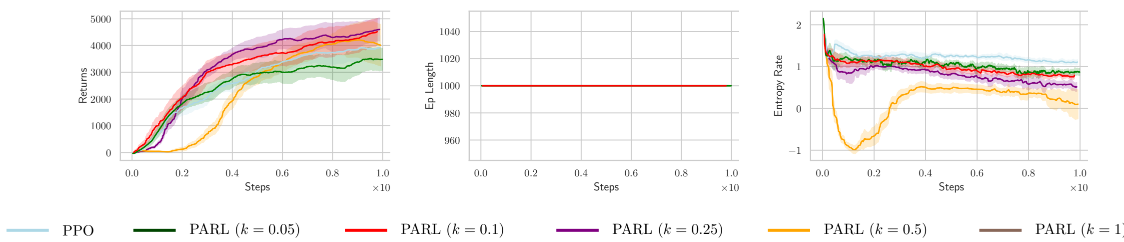


Figure 17: Training results for HalfCheetah-v4 environment.

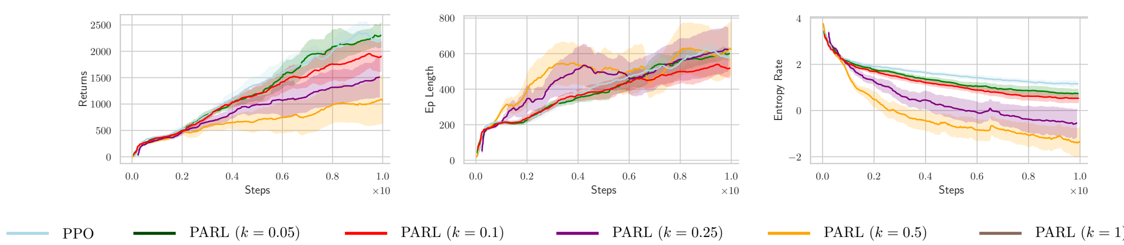


Figure 18: Training results for Walker2d-v4 environment.

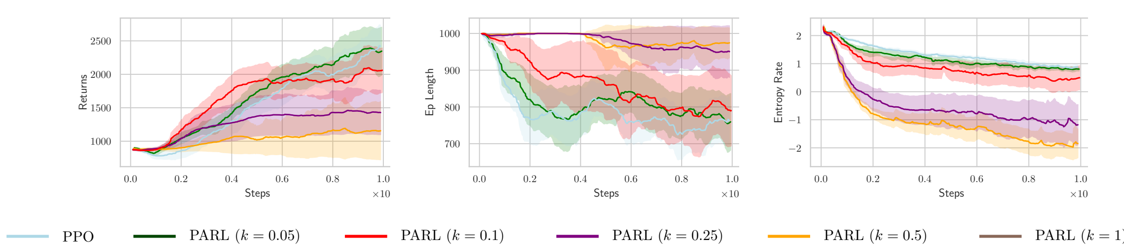


Figure 19: Training results for Ant-v4 environment.