Journal of Machine Learning Research 23 (2022) 1-68 Submitted 9/20; Revised 11/21; Published 9/22

Behavior Priors for Efficient Reinforcement Learning

Dhruva Tirumala[1,2] DHRUVAT@DEEPMIND.COM
Alexandre Galashov[1] AGALASHOV@DEEPMIND.COM
Hyeonwoo Noh[1,3] HYEONWOONOH@OPENAI.COM
Leonard Hasenclever[1] LEONARDH@DEEPMIND.COM
Razvan Pascanu[1] RAZP@DEEPMIND.COM
Jonathan Schwarz[1,2] SCHWARZJN@DEEPMIND.COM
Guillaume Desjardins[1] GDESJARDINS@DEEPMIND.COM
Wojciech Marian Czarnecki[1] LEJLOT@DEEPMIND.COM
Arun Ahuja[1] ARAHUJA@DEEPMIND.COM
Yee Whye Teh[1,2] YWTEH@DEEPMIND.COM
Nicolas Heess[1,2] HEESS@DEEPMIND.COM
[1] *DeepMind, R7, 14-18 Handyside Street, London, N1C 4DN, UK*
[2] *University College London, London WC1E 6BT, UK*
[3] *OpenAI, 3180 18th St, San Francisco, CA 94110 (contributions while at 1)*

Editor: Jan Peters

Abstract

As we deploy reinforcement learning agents to solve increasingly challenging problems, methods that allow us to inject prior knowledge about the structure of the world and effective solution strategies becomes increasingly important. In this work we consider how information and architectural constraints can be combined with ideas from the probabilistic modeling literature to learn *behavior priors* that capture the common movement and interaction patterns that are shared across a set of related tasks or contexts. For example the day-to day behavior of humans comprises distinctive locomotion and manipulation patterns that recur across many different situations and goals. We discuss how such behavior patterns can be captured using probabilistic trajectory models and how these can be integrated effectively into reinforcement learning schemes, e.g. to facilitate multi-task and transfer learning. We then extend these ideas to latent variable models and consider a formulation to learn hierarchical priors that capture different aspects of the behavior in reusable modules. We discuss how such latent variable formulations connect to related work on hierarchical reinforcement learning (HRL) and mutual information and curiosity based objectives, thereby offering an alternative perspective on existing ideas. We demonstrate the effectiveness of our framework by applying it to a range of simulated continuous control domains, videos of which can be found at the following url: `https://sites.google.com/view/behavior-priors`.

Keywords: reinforcement learning, probabilistic graphical models, control as inference, hierarchical reinforcement learning, transfer learning

1. Introduction

Recent advances have greatly improved data efficiency, scalability, and stability of reinforcement learning (RL) algorithms leading to successful applications in a number of domains (Mnih et al., 2015; Silver et al., 2016; Heess et al., 2017; Riedmiller et al., 2018; OpenAI et al., 2018; OpenAI, 2018). Many problems, however, remain challenging to solve or require large (often impractically so) numbers of interactions with the environment; a situation that is likely to get worse as we attempt to push the boundaries to tackle increasingly challenging and diverse problems.

One way to address this issue is to leverage methods that can inject prior knowledge into the learning process. Knowledge extracted from experts or from previously solved tasks can help inform solutions to new ones, e.g. by accelerating learning or by constraining solutions to have useful properties (like smoothness). Accordingly there has been much interest in methods that facilitate transfer and generalization across different subfields of the RL community, including work on transfer learning (Rusu et al., 2016; Christiano et al., 2016; Teh et al., 2017; Clavera et al., 2017; Barreto et al., 2019), meta learning (Duan et al., 2016; Wang et al., 2016; Finn et al., 2017; Mishra et al., 2017; Rakelly et al., 2019; Humplik et al., 2019) and hierarchical reinforcement learning (HRL) (Precup, 2000; Heess et al., 2016; Bacon et al., 2017; Vezhnevets et al., 2017; Frans et al., 2018; Wulfmeier et al., 2020a). For example, recent success in the game of StarCraft (Vinyals et al., 2019) relies on knowledge of useful skills and behaviors that were extracted from expert human demonstrations. The ability to extract and reuse behaviors can also be leveraged in the multi-task setting. While solving several tasks simultaneously is nominally harder, the ability to share knowledge between tasks may in fact make the problem easier. For example, this is often the case when tasks form a curriculum where the solutions to easier problems can inform the solutions to harder ones (e.g. Riedmiller et al., 2018).

A related question that then arises naturally is which representations are best suited to capture and reuse prior knowledge. One approach is to directly use prior data as a way to constrain the space of solutions (Vinyals et al., 2019; Fujimoto et al., 2018; Wang et al., 2020). An alternate approach that has gained much popularity expounds the use of hierarchical policies in order to combine together and sequence various skills and behaviors. These skills may be pre-defined, for instance, as motor primitives for control (Ijspeert et al., 2003; Kober and Peters, 2009), pre-learned with supervised learning or RL (e.g. Heess et al., 2016; Merel et al., 2019; Paraschos et al., 2013; Lioutikov et al., 2017), or can be learned on the fly through the use of sub-goal based architectures (Dayan and Hinton, 1993; Vezhnevets et al., 2017; Nachum et al., 2018). Alternatively, they are often motivated as models better suited to represent temporally correlated behaviors (Sutton et al., 1999; Precup, 2000; Daniel et al., 2016b; Bacon et al., 2017; Frans et al., 2018) and trained in an end-to-end manner. In this work, we present a unifying perspective to introduce priors into the RL problem. The framework we develop presents an alternative view that allows us to understand some previous approaches in a new light.

Our approach views the problem of extracting reusable knowledge through the lens of probabilistic modeling. We build on the insight that policies combined with the environment dynamics define a distribution over trajectories. This perspective allows us to borrow tools and ideas from the rich literature on probabilistic models to express flexible inductive

biases. We use this to develop a systematic framework around expressing prior knowledge, which we dub *behavior priors*, and which can express knowledge about solutions to tasks at different levels of detail and generality. They can be hand-defined or learned from data, integrated into reinforcement learning schemes, and deployed in different learning scenarios, e.g. to constrain the solution or to guide exploration. The framework admits for modular or hierarchical models which allow to selectively constrain or generalize certain aspects of the behavior such as low-level skills or high-level goals. The main contributions of our work can be summarized as follows:

- ***Behavior Priors* model trajectory distributions**: We develop the intuition of *behavior priors* as *distributions over trajectories* that can be used to guide exploration and constrain the space of solutions. In this view, a good prior is one that is general enough to capture the solutions to many tasks of interest while also being restrictive enough for tractable exploration.

- **Generalization and model structure**: We demonstrate how the parametric form of the prior can be used to selectively model different aspects of a trajectory distribution, including simple properties such as smoothness but also complicated, long-horizon and goal-directed behavior. In particular we discuss how more restricted forms of the prior can encourage generalization and empirically show that this can lead to faster learning.

- **Hierarchy, modularity, and model structure**: We develop a general framework that supports learning *behavior prior* models that can be structured into multiple modules or hierarchical layers that communicate through latent variables. We show how such structure can be used to further control the inductive bias and to achieve a separation of concerns, e.g. of low-level skills and high-level goals, and how it can be used to selectively transfer or constrain aspects of the behavior.

- **Connections to Hierarchical RL**: We discuss the relationship between the proposed probabilistic trajectory models and other lines of work in the RL literature. We show how common motifs from HRL can be motivated from the perspective of *behavior priors* , but also that model hierarchy is not a prerequisite for modeling hierarchically structured behavior.

- **Information theoretic regularization in RL**: We further highlight connections between our work and information theoretic regularization schemes applied in prior work. We find that our *behavior priors* can be motivated from the perspective of bounded rationality and information bottleneck, but that some of the models that we discuss also bear similarity to approaches motivated by curiosity and intrinsic motivation.

The rest of this work is split into two main parts. After some background for notation in Section 2, in Sections 3 and 4 we introduce our method and conduct an initial set of experiments for analysis and ablation to help ground these ideas. Following that, we extend our method to learn structured models in Section 5 and empirically evaluate them in Section 6. In Section 7, we describe a unifying framework that places our work in a broader context

of related work on HRL and methods based on mutual information and intrinsic rewards. Finally, we conclude with Section 8.

We have open sourced the code to run the tasks and agent used in this work at `https://github.com/deepmind/deepmind-research/tree/master/box_arrangement` and `https://github.com/deepmind/acme/tree/master/acme/agents/tf/svg0_prior` respectively. In addition, example videos of all tasks and an overview of our approach can be found at our accompanying website: `https://sites.google.com/view/behavior-priors`.

2. Background

We now introduce some notation and background information that will serve as the basis of the work in later sections. We start with some background definitions for RL and Markov decision processes (MDPs) (Sutton and Barto, 2018). For most of this work, we will limit our discussion to the discounted infinite-horizon setting for simplicity but our results also apply to the finite-horizon setting.

A Markov Decision Process (MDP) is defined by the following: S and A are state and action spaces, with $P : S \times A \times S \to \mathbb{R}_+$ a state-transition probability function or system dynamics and $P_0 : S \to \mathbb{R}_+$ an initial state distribution. We denote trajectories by $\tau = (s_0, a_0, s_1, a_1, \dots)$ and the state-action history at time step t including s_t (but not a_t) with $x_t = (s_0, a_0, \dots s_t)$. We consider policies π that are *history-conditional* distributions over actions $\pi(a_t|x_t)$ [1]. Given the initial state distribution, transition dynamics and policy, the joint distribution over trajectories τ is given as:

$$\pi(\tau) = P_0(s_0) \prod_{t=0}^{\infty} P(s_{t+1}|s_t, a_t)\pi(a_t|x_t), \tag{1}$$

where $s_t \in S$ is the state at time $t \geq 0$ and $a_t \in A$ the corresponding action. For notational convenience, we have overloaded π here to represent both the policy as well as the distribution over trajectories. The learning objective is to maximize expected discounted returns, given by a reward function $r : S \times A \to \mathbb{R}$, and a discount factor $\gamma \in [0, 1)$. Given a trajectory τ, the discounted return is

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t). \tag{2}$$

The expected discounted return over trajectories is then

$$\mathcal{J}(\pi) = \mathbb{E}_\pi[R(\tau)], \tag{3}$$

where the expectation is with respect to the trajectory distribution defined above. The goal of RL methods is to find an optimal policy $\pi^*(a|x)$ that maximizes the expected discounted return $\mathcal{J}(\pi)$ (Sutton and Barto, 2018).

1. We generally work with history dependent policies since we will consider restricting access to state information from policies (for information asymmetry), which may render fully observed MDPs effectively partially observed.

Given a policy, the value functions $V^\pi(x)$ and $Q^\pi(x, a)$ are defined as the expected discounted return conditioned on history x_t (and action a_t):

$$V^\pi(x_t) = \mathbb{E}_\pi[R(\tau)|x_t] = \mathbb{E}_{\pi(a|x_t)}[Q^\pi(x_t, a)],$$
$$Q^\pi(x_t, a_t) = \mathbb{E}_\pi[R(\tau)|x_t, a_t] = r(s_t, a_t) + \gamma \mathbb{E}_{P(s_{t+1}|s_t,a_t)}[V^\pi(x_{t+1})]. \tag{4}$$

3. Behavioral Priors for Control

In this work the notion of trajectory distributions induced by policies (which we defined in Equation 1) is an object of primary interest. For instance, we can think of the problem of finding a policy π that maximizes Equation (3) as that of finding a trajectory distribution for which the reward is maximal[2]; and we can similarly think of different exploration strategies as inducing different trajectory distributions. This perspective of manipulating trajectory distributions allows us to bring intuitions from the probabilistic modeling literature to bear. In particular, we will introduce a method that allows us to express prior knowledge about solution trajectories in RL. Our starting point is the KL regularized objective (Todorov, 2007; Kappen et al., 2012; Rawlik et al., 2013; Schulman et al., 2017a) [3]:

$$\mathcal{L} = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) - \alpha \gamma^t \mathrm{KL}[\pi(a_t|x_t)||\pi_0(a_t|x_t)]], \tag{5}$$

where 'KL' refers to the Kullback-Leibler divergence, a measure of similarity (or dissimilarity) between two distributions, which is defined as:

$$\mathrm{KL}[\pi(a_t|x_t)||\pi_0(a_t|x_t)] = \mathbb{E}_{a_t \sim \pi(.|x_t)}[\log \frac{\pi(a_t|x_t)}{\pi_0(a_t|x_t)}].$$

3.1 KL-Regularized RL

Intuitively, the objective in Equation (5) trades off maximizing returns with staying close (in the KL sense) to the trajectories associated with some reference behavior: π_0. Broadly speaking we can classify existing approaches that make use of this objective into two categories based on the choice of π_0. One approach, simply sets π_0 to be a uniform distribution over actions resulting in the entropy-regularized objective as used by Ziebart (2010); Schulman et al. (2017a); Haarnoja et al. (2017, 2018b); Hausman et al. (2018). This approach has been motivated in multiple ways (e.g Ahmed et al., 2019), and, in practice, it has been shown to be helpful in preventing the policy from collapsing to a deterministic solution; improving exploration; and increasing the robustness of the policy to perturbations in the environment. A second class of algorithms optimize $\mathcal{L}$ with respect to both π and π_0 and

2. Note that for MDPs and assuming no constraints on what a given policy class to which π belongs can represent, a deterministic optimal policy will exist. If the transition dynamics are also deterministic then the trajectory distribution may collapse to a single trajectory which we can represent as a product of indicator functions.
3. We derive this per-timestep KL-regularized objective in Appendix D.1 when π and π_0 have a form that is constrained by the system dynamics.

are often referred to as *EM-policy search* algorithms. Examples of this style of algorithms include Peters et al. (2010); Toussaint and Storkey (2006); Rawlik et al. (2013); Levine and Koltun (2013); Levine et al. (2016); Montgomery and Levine (2016); Chebotar et al. (2016); Abdolmaleki et al. (2018b). Although details vary, the common feature is that π_0 allows to implement a form of trust-region that limits the change in π on each iteration but does not necessarily constrain the final solution. Different specific forms for π and π_0 can then lead to alternate optimization schemes.

In this work we focus on a different perspective. We consider cases where π_0 provides structured prior knowledge about solution trajectories. In this view π_0 can be thought of as a *behavior prior* , and Equation (5) results in a trade off between reward and closeness of the policy π to the prior distribution over trajectories defined by π_0. The prior π_0 can be seen as a constraint, regularizer, or shaping reward that guides the learning process and shapes the final solution. We discuss how π_0 can be learned from data; how the form of π_0 determines the kind of knowledge that it captures and how this in consequence determines how it affects the learning outcome in a number of different transfer scenarios.

3.2 Multi-Task RL

To gain more intuition for the objective in Equation (5) we first consider the multi-task RL scenario from Teh et al. (2017), with a distribution over tasks $p(w)$ where tasks are indexed by $w \in \mathcal{W}$. The tasks share the transition dynamics but differ in their reward functions r_w. We consider the KL-regularized objective with task-specific policy π but a 'shared' prior π_0 which has no knowledge of the task:

$$\mathcal{L} = \sum_w p(w) \mathbb{E}_{\pi_w} \left[\sum_t \gamma^t r_w(s_t, a_t) - \gamma^t \mathrm{KL}[\pi_w(a_t|x_t)||\pi_0(a_t|x_t)] \right], \tag{6}$$

For a given π_0 and task w, we then obtain the optimal policy π_w as follows:

$$\pi_w^*(a|x_t) = \pi_0(a_t|x_t) \exp(Q_w^*(x_t, a) - V_w^*(x_t)), \tag{7}$$

where Q_w^* and V_w^* are the optimal state-action value function and state value function for task w respectively which are given by,

$$V_w^*(x_t) = \max_{\pi_w \sim \Pi} \mathbb{E}_{x_t \sim d_{\pi_w,t}} V_w^\pi(x_t),$$
$$Q_w^*(x_t, a) = r(s_t, a) + \gamma \mathbb{E}_{P(x_{t+1}|x_t,a)}[V_w^*(x_{t+1})],$$

where Π denotes the space of all policies and

$$d_{\pi_w,t} = P_0(s_0) \prod_{t'=0}^{t-1} \pi_w(a_{t'}|s_{t'}) P(s_{t'+1}|s_{t'}, a_t).$$

On the other hand, given a set of task specific policies π_w, the optimal prior is the one that optimizes:

$$\pi_0^* = \arg\min_{\pi_0} \sum_w p(w) \mathbb{E}_{x_t \sim d_{\pi_w,t}} [\mathrm{KL}[\pi_w(\cdot|x_t)||\pi_0(\cdot|x_t)]], \tag{8}$$

$$\tag{9}$$

and is given by (proof in Appendix D.2):

$$\pi_0^* = \sum_w p(w|x_t)\pi_w(a_t|x_t). \tag{10}$$

where $p(w|x_t)$ is the posterior distribution over task indices given the trajectory snippet x_t.

Together Equations (7) and (10) define an alternating optimization scheme that can be used to iteratively improve on the objective in Equation (6). These results provide important intuition for the behavior of the KL regularized objective with respect to π and π_0. In particular, Equation (7) suggests that given a prior π_0 the optimal task-specific policy π_w is obtained by reweighting the prior behavior with a term that is proportional to the (exponentiated) soft-Q function associated with task w. Since the policy π_w is the product of two potential functions (π_0, and $\exp Q$) it can effectively be seen as specializing the behavior suggested by the prior to the need of a particular task. Assuming that we can learn Q_w^* we could directly use Equation (7) as a representation of the policy In practice, however, we normally learn a separately parametrized task specific policy π_w as we detail in Section 3.5.

In contrast, the optimal prior π_0^* for a set of task-specific experts π_w is given as a weighted mixture of these task specific policies where the weighting is given by the posterior probability of each of these task specific policies π_w having produced trajectory x_t. In other words, the optimal prior π_0 marginalizes over the task w and produces the same trajectory distribution as if we first picked a task at random and then executed the associated expert. This is a useful policy insofar that, given an unknown task w, if we execute π_0 repeatedly, it will eventually execute the behavior of the associated expert π_w. Since π_0 represents the behavior that is sensible to execute when the task is unknown it is also referred to as a *default behavior* (Galashov et al., 2019). In practice $p(w|x_t)$ is intractable to compute but we can easily sample from the expectation in Equation (8) and fit a parametric model via distillation (i.e. by minimizing $\mathbb{E}_{\pi_w}[\log \pi_0(a_t|x_t)]$ as in Equation 8).

The optimal solutions for π_0 and π_w thus satisfy the following basic intuition: For a given distribution over tasks $p(w)$ the optimal prior π_0 contains the optimal behavior for all tasks as a task-agnostic mixture distribution. The optimal task-specific policy π_w then specializes this distribution to fit a particular task w. This result is a consequence of the properties of the KL divergence, in particular its asymmetry with respect to its arguments: For two distributions q and p the $\mathrm{KL}[q||p]$ will favor situations in which q is fully contained in the support of p, and p has support at least everywhere where q has support. In other words, the KL divergence is mode-seeking with respect to q, but mode covering with respect to p (e.g. Bishop, 2006). In the next section we will discuss how these properties can be exploited to control generalization in RL.

3.3 General Information Asymmetry for *Behavioral Priors*

Studying Equation (5) we notice that if π_0 had access to all the information of π , then the optimal solution would be to just set $\pi_0^* = \pi^*$. Thus, it is the fact that we constrain the processing capacity of π_0, by removing w from the information that π_0 has access to that results in a default policy that *generalizes* across tasks in Equation (10). We can extend this intuition by considering priors π_0 which are restricted by limiting their modeling capacity

or the information they have access to more generally. To make this precise we split x_t into two disjoint subsets x_t^G and x_t^D [4] and allow π_0 access only to x_t^D, i.e. $\pi_0(\cdot|x_t) = \pi_0(\cdot|x_t^D)$ while π retains access to all information $\pi(\cdot|x_t) = \pi(\cdot|x_t^G, x_t^D)$. The objective in Equation (5) then becomes:

$$\mathcal{L} = \mathbb{E}_\pi[\sum_t \gamma^t r(s_t, a_t)] - \gamma^t \mathrm{KL}[\pi(a_t|x_t)||\pi_0(a_t|x_t^D)]. \tag{11}$$

In the notation of the previous section $x_t^G = w$ and $x_t^D = x_t$. But we could, for instance, also choose $x_t^D = (a_0, a_1, \dots a_t)$ and thus allow π_0 to only model temporal correlations between actions. In our experiments we will consider more complex examples such as the case of a simulated legged 'robot' which needs to navigate to targets and manipulate objects in its environment. We will give π_0 access to different subsets of features of the state space, such as observations containing proprioceptive information (joint positions, velocities, etc.), or a subset of exteroceptive observations such as the location and pose of objects, and we study how the behavior modeled by π_0 changes for such choices of x_t^D and how this affects the overall learning dynamics.

The interplay between π and π_0 is complex. An informative π_0 simplifies the learning problem since it effectively restricts the trajectory space that needs to be searched[5]. In the multi-task scenario of the previous section, and in the more general case of asymmetry described in this section, π_0 can generalize shared behaviors across tasks (or across different values of x_t^G, i.e. across different parts of the trajectory space). However, the presence of π_0 will also affect the solution learned for π in the sense that the overall objective will favor π that are closer to π_0, for instance because they have little reliance on x_t^G. This may improve robustness of π and help generalization to settings of x_t^G not seen during training. We will discuss this last point in more detail in Section 7 when we discuss the relation to information bottleneck.

It is worth noting that when, as is usually the case, π_0 is separately parametrized as π_0^ϕ and learned (e.g. as described in Section 3.5), then its parametric form will further affect the learning outcome. For instance, choosing π_0^ϕ to be a linear function of x_t [6] will limit its ability to model the dependence of a_t on x_t. Furthermore, choosing the parametric model for π_0^ϕ to be a Gaussian may limit its ability to model the mixture distribution in Equation (10). Here, too, the direction of the KL will force π_0^ϕ to extrapolate. Overall, the information that π_0 is conditioned on, the parametric form of the function approximator, and the form of the sampling distribution determine the behavior model that π_0 learns and how it encourages generalization across the trajectory space.

Finally, an interesting aspect of the joint optimization scheme is that as the prior learns, it can play the role of a 'shaping reward'. The structure that it learns from the solutions to some tasks can be used to inform solutions to others. For instance, in a locomotion task,

4. In line with terminology from the previous section we use superscripts D for 'default' and G for 'goal' although we now consider a more general use.
5. Concretely, the second term in Equation (6) can be considered as a shaping reward; and Equation (7) emphasizes how π_0 restricts the solutions for π. Trajectories not included in the support of π_0 will not be included in the support of π and are thus never sampled.
6. When we say that π_0 (or π) is a linear function of x_t we mean that the parameters of the policy distribution, e.g. the mean and the variance in if π_0 is Gaussian, are linear functions of x_t.

the same underlying movement patterns are needed for goals that are close by and those that are further away. We will explore some of these effects in our experiments in Section 4 and include a simple 2-D example to help develop an intuition for this in Appendix B.

3.4 Connection to variational Expectation Maximization (EM)

The entropy regularized objective is sometimes motivated by comparing it to the problem of computing a variational approximation to the log-marginal likelihood (or, equivalently, the log-normalization constant) in a probabilistic model. Similarly, we can establish a connection between Equation (6) and the general variational framework for learning latent variable models (Dempster et al., 1977) (which we briefly review in Appendix A). In that setup the goal is to learn a probabilistic model p_θ of some data $x_{1:N} = (x_1, \dots x_N)$ by maximizing the log marginal likelihood $\log p_\theta(x) = \sum_i \log p_\theta(x_i)$ where $p_\theta(x) = \int p_\theta(x, z)dz$. This likelihood can be lower bounded by $\sum_i \mathbb{E}_q(z|x_i)[\log p_\theta(x_i|z) - \log(\frac{q_\phi(z|x_i)}{p_\theta(z)})]$ where q_ϕ is a learned approximation to the true posterior.

We can draw a correspondence between this framework and the objective from Equation (5). From this perspective each data point x_i is a task in a multi-task setting and the latent variable z defines a trajectory τ in the corresponding MDP. In other words the conditional probability $\log p_\theta(x_i|z)$ measures the sum of rewards generated by the trajectory z for task i. Note that in this case p_θ has no learnable parameters and is a measure of the goodness of fit of a trajectory for a given task. The *prior* $p_\theta(z)$ is now a distribution over trajectories generated under the *behavior prior* and system dynamics. Similarly the posterior $q_\phi(z|x_i)$ defines a distribution over trajectories under the *policy* and system dynamics. We will sometimes refer to the policy as the 'posterior' because of this equivalence.

3.5 Algorithmic considerations

The objective for training *behavior priors* from Equation (11) involves solving a regularized RL problem similar to Schulman et al. (2017a); Hausman et al. (2018); Haarnoja et al. (2018b), as well as an additional distillation step for optimizing the prior. For this work, we adapt the off-policy actor-critic algorithm SVG-0 of Heess et al. (2015) although as such our method can easily be incorporated into any RL algorithm. The SVG-0 algorithm proceeds by collecting trajectories from some behavior policy μ which are stored into a replay buffer (as in Mnih et al., 2015). K length tuples of experience data $(s_t, a_t, r_t, ..., s_{t+K}, a_{t+K}, r_{t+K})$ are then sampled from the buffer for learning. This data is used to train 3 components: a critic; an actor and a *behavior prior*. Additionally, we use target networks for each of these components as in Mnih et al. (2015) which we found stabilized learning.

Critic update: In order to estimate the state-action value function $Q(s, a)$, we use the retrace estimator (Munos et al., 2016) which has been shown to reduce variance in the multistep off-policy setting. For some timestep t the estimate is given by:

$$Q^R(x_t, a_t) = Q(x_t, a_t) + \sum_{s \geq t} \gamma^{s-t} \left(\prod_{i=t}^{s} c_i \right) \left(r_s + \gamma V(x_{s+1}) - Q(x_s, a_s) \right), \tag{12}$$

$$c_i = \lambda \min \left(\frac{\pi(a_i|x_i)}{\mu(a_i|x_i)}, 1 \right),$$

Algorithm 1 Learning *priors*: SVG(0) with experience replay

Policy: $\pi_\theta(a_t|x_t)$ with parameters θ
Behavioral Prior: $\pi_{0,\phi}(a_t|x_t^D)$
Q-function: $Q_\psi(a_t, x_t)$ with parameters ψ

Initialize target parameters $\theta' \leftarrow \theta, \quad \phi' \leftarrow \phi, \quad \psi' \leftarrow \psi$.
Hyperparameters: α, α_H, β_π, β_{π_0}, β_Q
Target update counter: $c \leftarrow 0$
Target update period: P
Replay buffer: $\mathcal{B}$

for $j = 0, 1, 2, \ldots$ **do**
Sample partial trajectory $\tau_{t:t+K}$ generated by behavior policy μ from replay $\mathcal{B}$:
$\tau_{t:t+K} = (s_t, a_t, r_t, ..., r_{t+K})$

for $t' = t, ...t + K$ **do**
$\hat{\mathrm{KL}}_{t'} = \mathrm{KL}\left[\pi_\theta(.|x_{t'})\|\pi_{0,\phi'}(.|x_{t'}^D)\right]$ ▷ Compute KL using target prior.
$\hat{\mathrm{KL}}_{t'}^{\mathcal{D}} = \mathrm{KL}\left[\pi_\theta(.|x_{t'})\|\pi_{0,\phi}(.|x_{t'}^D)\right]$ ▷ Compute KL for distillation with online prior.
$\hat{\mathrm{H}}_{t'} = \mathbb{E}_{\pi_\theta(a|x_{t'})}[\log \pi_\theta(a|x_{t'})]$ ▷ Compute action entropy

$\hat{V}_{t'} = \mathbb{E}_{\pi_\theta(a|x_{t'})}\left[Q_{\psi'}(a, x_{t'})\right] - \alpha\hat{\mathrm{KL}}_{t'}$ ▷ Estimate bootstrap value
$\hat{c}_{t'} = \lambda \min\left(\frac{\pi_\theta(a_{t'}|x_{t'})}{\mu(a_{t'}|x_{t'})}, 1\right)$ ▷ Estimate traces Munos et al. (2016)
$\hat{Q}_{t'}^R = Q_{\psi'}(a_{t'}, x_{t'}) + \sum_{s\geq t'} \gamma^{s-t'} \left(\prod_{i=t'}^{s} \hat{c}_i\right)\left(r_s + \gamma\hat{V}_{s+1} - Q_{\psi'}(a_s, x_s)\right)$
▷ Apply Retrace to estimate Q targets

end for

$\hat{L}_Q = \sum_{i=t}^{t+K-1} \|\hat{Q}_i^R - Q_\psi(a, x_i)\|^2$ ▷ Q-value loss

$\hat{L}_\pi = \sum_{i=t}^{t+K-1} \mathbb{E}_{\pi_\theta(a|x_i,\eta)} Q_{\psi'}(a, x_i) - \alpha\hat{\mathrm{KL}}_i + \alpha_{\mathrm{H}}\hat{\mathrm{H}}_i$ ▷ Policy loss

$\hat{L}_{\pi_0} = \sum_{i=t}^{t+K-1} \hat{\mathrm{KL}}_i^{\mathcal{D}}$ ▷ *Behavior Prior* loss

$\theta \leftarrow \theta + \beta_\pi \nabla_\theta \hat{L}_\pi \qquad \phi \leftarrow \phi + \beta_{\pi_0} \nabla_\phi \hat{L}_{\pi_0} \qquad \psi \leftarrow \psi - \beta_Q \nabla_\psi \hat{L}_Q$
Increment counter $c \leftarrow c + 1$
if $c > P$ **then**
Update target parameters $\theta' \leftarrow \theta, \quad \phi' \leftarrow \phi, \quad \psi' \leftarrow \psi$
$c \leftarrow 0$
end if
end for

where r is the task reward, π is the current policy, c_i represents a form of clipped importance sampling for off-policy correction and V is given by:

$$V(x_t) = \mathbb{E}_{a\sim\pi(.|x_t)}\left[Q(x_t, a)\right] - \alpha\mathrm{KL}\left[\pi(.|x_t)\|\pi_0(.|x_t^D)\right].$$

Note that we incorporate the KL term from Equation (11) for timestep $s > t$ through the bootstrap value V in Equation (12). This is because for the tuple (x_t, a_t, r_t) at time t, the reward r_t is the effect of the *previous* action a_{t-1} while $\mathrm{KL}\left(\pi(.|x_t)||\pi_0(.|x_t)\right)$ corresponds to the action in the *current* state s_t. The KL for timestep t is thus added to the policy update instead. With this, we can write down the objective for the critic as follows:

$$L_Q = \sum_{i=t}^{t+K-1} \|Q^R(x_i, a_i) - Q(x_i, a_i)\|^2.$$

Policy update: The SVG-0 algorithm trains a stochastic policy from the value function $Q(s, a)$ via reparameterization of the policy gradient. Our method additionally regularizes against a *behavior prior* by adding a KL term to the policy loss. Putting these together we can write down the objective for the policy as:

$$L_\pi = \sum_{i=t}^{t+K-1} \mathop{\mathbb{E}}_{\substack{\eta\sim\rho \\ a\sim\pi(.|x_i,\eta)}} [Q(x_i, a)] - \alpha \mathrm{KL}\left[\pi(.|x_i) \| \pi_0(.|x_i^D)\right] + \alpha_\mathrm{H} \mathrm{H}_i(\pi), \tag{13}$$

where η is random noise drawn from a distribution ρ and H_i is the conditional entropy of the policy evaluated in state x_i. This objective is thus similar to the one considered in Teh et al. (2017).

To isolate the effect of the regularization to the learned prior we aim to keep our baseline as similar to our main algorithm as possible. We thus consider the objective of Equation (13) but do not include the KL term to the *behavior prior*.

$$L_{\pi_{baseline}} = \sum_{i=t}^{t+K-1} \mathop{\mathbb{E}}_{\substack{\eta\sim\rho \\ a=\pi(s_i,\eta)}} [Q(s_i, a)] + \alpha_\mathrm{H} \mathrm{H}_i(\pi). \tag{14}$$

This entropy-regularized objective has also been used in Riedmiller et al. (2018) and is similar to the one considered by Haarnoja et al. (2018b); Hausman et al. (2018); Mnih et al. (2016); Williams and Peng (1991). We provide a more detailed theoretical and empirical comparison between SVG0 and other algorithms like Soft Actor Critic (SAC) (Haarnoja et al., 2018b) and Deterministic Policy Gradients (DPG) (Silver et al., 2014) in Appendix C.

Prior update: Finally we train the *behavior prior* $\pi_0(.|x_t^D)$ to match the policy distribution $\pi(.|x_t)$ with the following objective:

$$L_{\pi_0} = \sum_{i=t}^{t+K-1} \mathrm{KL}\left[\pi(.|x_i) \| \pi_0(.|x^D{}_i)\right].$$

We refer to this form of distribution matching through minimizing the KL as 'distillation' in keeping with Teh et al. (2017). The full procedure used for training is shown in Algorithm 1, where the additions for learning a *prior* are highlighted in blue. We used separate ADAM optimizers (Kingma and Ba, 2014) for training the critic, policy and *behavior prior*. A full list of hyperparameters used is presented in Appendix F.

4. Experiments

In this section, we analyze the effect of *behavior priors* experimentally on a number of simulated motor control domains using walkers from the DeepMind control suite (Tassa et al., 2018) developed using the MuJoCo physics engine (Todorov et al., 2012). The purpose of these experiments is to understand how priors with various capacity and information constraints can learn to capture general task-agnostic behaviors at different levels of abstraction, including both basic low-level motor skills as well as goal-directed and other temporally extended behavior. We consider a range of multi-task and transfer problems with overlapping solution spaces that we can model at varying levels of detail and generality.

Our range of tasks requires a simulated agent to reach various goal locations or manipulate objects in its environment. The locations of the goals, objects and walker are chosen at random for every episode. In other words, each task is a distribution over goals and targets, some of which are harder to solve than others. However all of them share a common *underlying structure*; their solutions involve consistent patterns of interaction between the agent and environment. For instance, a common element underpinning these tasks is that the gait or underlying pattern of movement of any agent that solves it is largely goal independent. More specific and less general behavior that recurs across tasks includes goal-directed locomotion or object interactions.

In Section 4.1 we show information constraints with simple Multilayer Perceptron (MLP) models can learn to model these behaviors. For instance, we demonstrate that priors without any hierarchical structure can learn to model temporally correlated patterns of movement for a complex 21 degree-of-freedom (DoF) humanoid body. Apart from information constraints, we are also interested in understanding how architectural choices can affect the kinds of behaviors modeled under the prior. With this in mind, we include problems that are specifically designed to exhibit structure at different temporal scales: In the 'Sequential Navigation' tasks an agent must visit several targets in *sequence*, with some target sequences being much more likely to occur. In this setting we show how some architectures can learn to model these dynamics.

We vary the complexity of these tasks using more complicated bodies (*Locomotion (Humanoid)*); changing the number of boxes or targets (*Manipulation (2 boxes, 2 targets)*); or by combining different objectives (*Locomotion and Manipulation*). All the tasks that we consider in this work use sparse rewards. Sparse reward tasks are typically harder to learn but are easier to specify since they do not require hand engineered per-timestep rewards. This setting provides an interesting context in which to study *behavior priors*. As the prior learns to model solutions for some instances, it can help *shape* the learning process and guide the policy to discover solutions to new ones. As we show below, this can lead to faster convergence for the policy on a range of tasks.

For convenience, we have summarized the tasks used in this section and Section 6 in Table 1. Videos of all tasks can be found on our accompanying website: `https://sites.google.com/view/behavior-priors`. The code to run these tasks can be found at `https://github.com/deepmind/deepmind-research/tree/master/box_arrangement` and the code for the agent is available at `https://github.com/deepmind/acme/tree/master/acme/agents/tf/svg0_prior`.

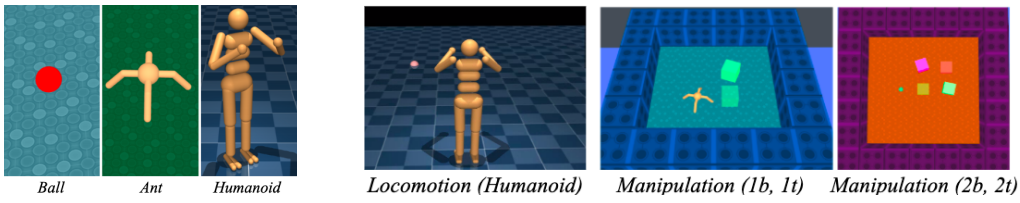
Unless otherwise specified, learning curves show returns (on the Y-axis) as a function of the number of environment steps processed by the learner (X-axis) and for each curve, we plot the mean of the best performing configuration averaged across 5 seeds with shaded areas showing the standard error. A complete list of hyperparameters considered is included in Appendix F. All experiments were run in a distributed setup using a replay buffer with 32 CPU actors and 1 CPU learner.

4.1 Effect of Information Asymmetry

The goal of this section is to both qualitatively and quantitatively understand the effect of modeling priors with information constraints.

Task Name	Description	Body	Action Dim.	Obs.	Obs. to *Prior*
Navigation Easy	Visit targets in a sequence where the sequence is history dependent.	PointMass / Ant	2/8	Proprio + Targets + next target index	Proprio + targets
Navigation Hard	Visit targets in order.	PointMass / Ant	2/8	Proprio + Targets	Proprio + Targets
Locomotion (Humanoid)	Follow a moving target	Humanoid	23	Proprio + Target location	Varies (See text)
Locomotion (Ant)	Go to one of 3 targets	Ant	8	Proprio + Targets + Index of goal	Varies (See text)
*Locomotion (Gap)**	Move forward and jump across a variable length gap.	Ant	8	Proprio + Global position + Gap length	Proprio
Locomotion and Manipulation	Move a box to a specified target and then move to another target.	Ant	8	Proprio + Box + Targets + Task encoding	Varies (See text)
*Manipulation (1 box, 1 target)**	Move a box to a target location.	Ant	8	Proprio + Box + Targets + Task encoding	Proprio + Box
*Manipulation (1 box, 3 target)**	Move a box to one of 3 target locations.	Ant	8	Proprio + Box + Targets + Task encoding	Proprio + Box
*Manipulation (2 box, 2 target)**	Move one of two boxes to one of 2 target locations.	Ball	2	Proprio + Boxes + Targets + Task encoding	Proprio + Boxes
*Manipulation (2 box gather)**	Move two boxes close together.	Ball	2	Proprio + Boxes + Targets + Task encoding	Proprio + Boxes

Table 1: **List of Tasks** A short summary of tasks used along with the information asymmetry used to train the *behavior prior*. Tasks marked with an asterisk are considered in Section 6. A more complete description is presented in Appendix E.1.



(a) All bodies considered. (b) Some of the tasks considered

Figure 1: **Visualization of tasks and bodies.** An illustration of some of the tasks and the bodies used for our experiments.

To begin with, we focus our discussion on the *Locomotion and Manipulation* task which is also considered in the hierarchical setting in Section 6.1. This task involves many behaviors like goal-directed movement and object interactions and thus provides a rich setting in which

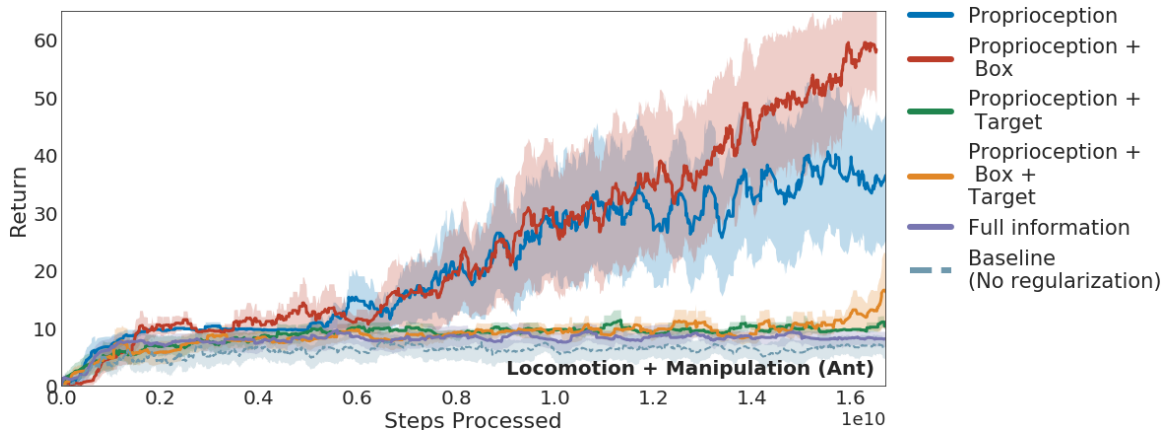


Figure 2: **Effect of information asymmetry.** Effect of learning with different levels of information asymmetry on the *Locomotion and Manipulation* task. Label shows the kinds of information made accessible to the *prior*.

to study the effects of information asymmetry in some detail. We expand our discussion to a range of other tasks in Section 4.1.2.

4.1.1 Locomotion and Manipulation Task

This task involves a walker, a box and two goal locations. In order to solve the task, the agent must move the box to one of the goals (manipulation) and subsequently move to the other goal (locomotion). The locations of the agent, box and goals are randomized at the start of each episode. The agent receives a reward of +10 for successfully accomplishing either sub-task and an additional bonus of +50 when it solves both. For this task we use an 8-DoF ant walker as shown in Figure 1a.

The agent is given proprioceptive information, the location of the goals and box and the identity of the goal it must go to. The proprioceptive information includes the relative joint positions and velocities of the agent's body (details in Appendix E.2). There are many choices for the information set sent to the prior which affect the kinds of behaviors it can learn to model. For instance the prior cannot capture the behavior of interacting with the box if it has no access to box information. On the other hand, a prior with access to all of the same information as the policy will learn to mimic the policy and lose the regularizing effect created by the information asymmetry. We examine whether this impacts performance on this task.

We present our results in Figure 2. We find a marked difference in performance based on the information that the *behavior prior* has access to. We observe that most policies on this task can easily get stuck in a local optima where the agent learns to solve the easier locomotion component of the task but never learns about the reward for moving the box. However, *behavior priors* can help regularize against this sub-optimal solution by encouraging useful patterns of behavior. For instance, a prior that has access to *only*

proprioceptive information learns to encode the space of trajectories containing primitive gaits or movements. By encouraging the policy to continue to explore in this space, the agent is more likely to continue learning and to see the full task reward. In this case we find that the *behavior prior* that has access to both proprioception as well as the location of the box learns the fastest.

To understand why this might be, we perform a qualitative analysis by generating trajectories from some of the priors trained in Figure 2. Specifically, in Figure 3, we compare the kinds of behaviors that emerge when the prior has (left) only proprioceptive information; (middle) proprioception + box information and (right) proprioception + target locations. We observe that the prior with only proprioception models a space of undirected movement behaviors. This behavior is already quite complex in that it involves specific patterns of joint actuation to move the body that can be modeled solely through information present in the proprioceptive state. In contrast to this, the prior with access to the box information learns to model behaviors related to moving the box around. The blue dotted line represents trajectories showing movement of the box as the agent interacts with it. Finally the agent with access to proprioception and the location of the targets (and not the box) also learns to model 'goal-oriented' patterns of moving towards the different targets.

It is worth pausing to understand this in more detail. While each instance of the task might involve a specific solution, like moving from point A to point B, together they all share a common behavior of 'movement'. In fact these kinds of behaviors can be seen to occur at various levels of generality. For example, moving towards a specific object or target is less general than a goal-agnostic meander. The latter is more general and may thus be useful in new scenarios like maze exploration where there may not be any objects or targets. The analysis of Figure 2 demonstrates that information constraints allow the prior to generalize the behaviors it models. Moreover, different choices for the information set affect the kinds of behaviors modeled.

For this task it appears that the prior shown in Figure 3b performs the best but in general which of these behaviors is better depends on the setting we would like to evaluate them in. For instance, if we were to transfer to a task where interacting with the box results in a negative reward or where the goal is to move to one of the targets, then the prior in Figure 3c might work better than the one in Figure 3b. Alternatively a task where the optimal solution favors random exploration away from the box *and* the targets would suit the prior in Figure 3a. The point here is to demonstrate that altering the information set that the prior has access to can lead to different kinds of behaviors that are modeled under it and the best choice may depend on the domain we are ultimately interested in solving.

4.1.2 Other Tasks

In this section, we expand our analysis of the effect of information asymmetry for learning to three more tasks: *Manipulation (1 box, 3 targets)*, *Locomotion (Humanoid)* and *Locomotion (Ant)* (refer to Table 1 for a brief summary).

We show our findings in Figure 4. We observe that regularizing against a *behavior prior* always provides a benefit and priors with partial information perform better than ones with full information. Additionally, *behavior priors* with access to only proprioceptive information tend to perform the best in tasks involving just locomotion.

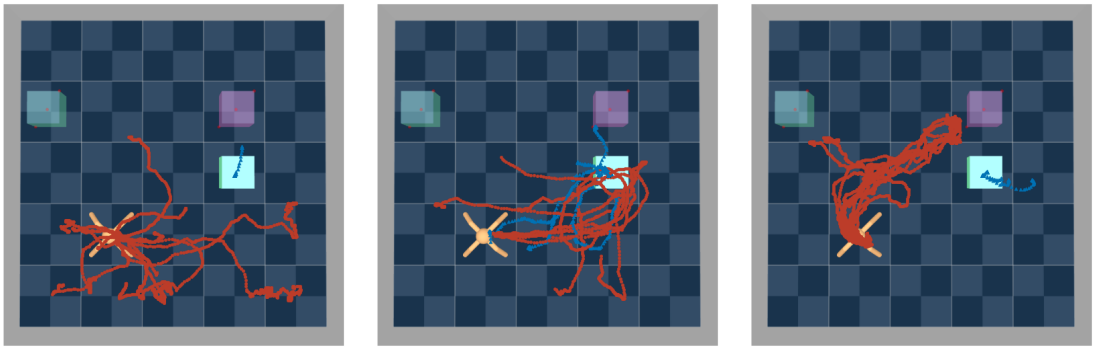


Figure 3: **Qualitative effect of information asymmetry.** Trajectories generated under *behavior priors* with access to different information sets for a fixed configuration of the box (solid blue), targets (semi-transparent blue and pink) and walker. The *behavior prior* has access to (left) proprioceptive information; (middle) proprioceptive information and the location of the box; (right) proprioceptive information and location of target. Red dotted line represents the movement of the ant and blue dotted line represents the movement of the box across multiple episodes.

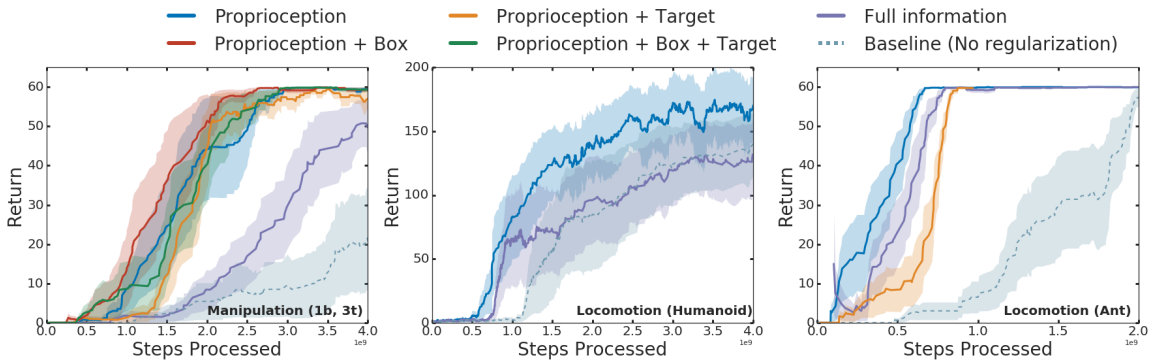
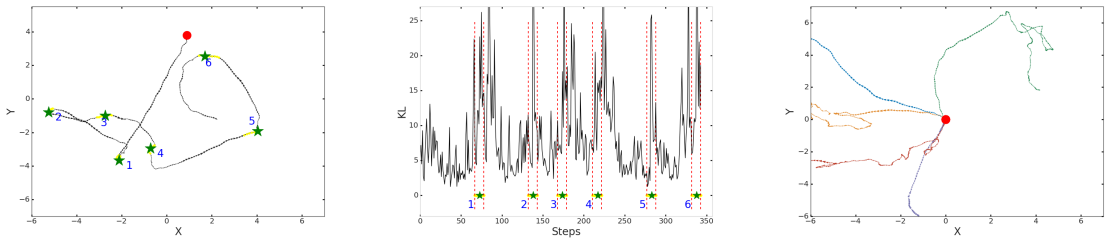


Figure 4: **Effect of information asymmetry.** Effect of learning with different levels of information asymmetry on *Manipulation (1 box, 3 targets)*, *Locomotion (Humanoid)* and *Locomotion (Ant)*.



(a) Trajectory of humanoid visiting targets in the *Locomotion (Humanoid)* task under the optimal policy.

(b) Deviation of policy from the *behavior prior* in terms of KL divergence.

(c) Trajectories generated by the *behavior prior*.

Figure 5: **Analysis of the *Locomotion (Humanoid)* task**. The spikes in KL divergence in the center align with the change in target as on the left; this is consistent with the idea that the *behavior prior* encodes primitive movement patterns. (right) Trajectories generated under the *behavior prior* result in forward movement with occasional turns.

In order to understand this in greater detail, we qualitatively analyze the *Locomotion (Humanoid)* task in Figure 5. Our analysis demonstrates that the prior with only proprioceptive information models forward movement in different directions with occasional sharp turns. In Figure 5b we plot the KL divergence between the prior and policy as the agent visits different goals. Spikes in the KL divergence align precisely with the points where new targets are chosen as shown in the trajectory in Figure 5a. As the figure illustrates, the *behavior prior* and the policy match quite closely (and so the KL is low) until the agent reaches a target. At this point, the policy changes to a new pattern of movement in order to change direction towards the new target location. As the humanoid settles into a new movement pattern, we observe that the KL once again reduces. This sequence repeats for each of the targets. Furthermore, in Figure 5c, we generate trajectories sampled from the learnt prior starting from the same initial position. The *behavior prior* captures forward movement in different directions with occasional sharp turns.

These findings are consistent with our observations from the previous section. The prior does not have access to the goal location. Therefore, it learns to mimic the experts behavior by capturing higher order statistics relating to movement that largely depends on the current configuration of the body. While this behavior is not perfect and mostly involves forward movements with occasional sharp turns, it demonstrates that simple MLP models can learn to capture general behaviors like goal-agnostic movement even for complex bodies like the 21-DoF humanoid robot.

4.2 Sequential Navigation

In the previous section we demonstrated that information constraints in the prior affect the kinds of behaviors it learns and how this in turn can affect the speed of learning. In this section, we instead analyze how model architecture can affect the kinds of behaviors that are captured by a *behavior prior*. To this end, we study 'Sequential Navigation' tasks that

are designed to exhibit structure at multiple temporal and spatial scales and thus allow us to study behaviors at a coarser timescale. In these tasks an agent must learn to visit targets in a history-dependent sequence as illustrated by Figure 6.

Specifically, sequences of targets are chosen based on 2nd order Markovian dynamics where some targets are more likely to be chosen than others given the two preceding targets that were visited. However the marginal probabilities given the last visited target are uniformly distributed among the remaining ones. In other words, a prediction about the next target to visit can be made given the last two targets but nothing can be gleaned when only the last target is known (see Appendix E for details). In this setting the behavioral structure that could usefully be modeled by the prior includes, effectively, a hierarchy of behaviors of increasing specificity: undirected locomotion behavior, goal-directed locomotion, as well as the long-horizon behavior that arises from some target sequences being more likely than others.

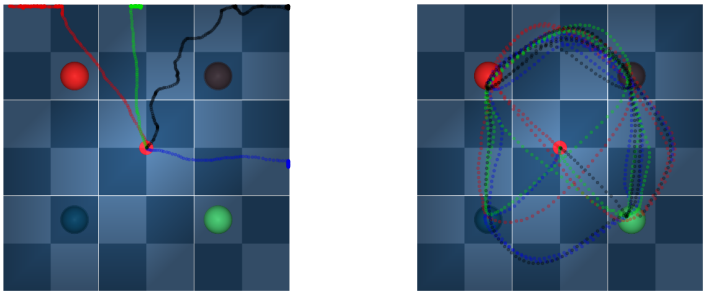
We consider two versions of this task:

- **Sequential Navigation Easy**: On each episode, a random sequence of targets is chosen based on the underlying Markovian dynamics. In this setting the agent is given the immediate next target to visit as input at each point in time.
- **Sequential Navigation Hard**: In this setting, the agent must learn to visit a single 4 target sequence (target 1 followed by target 2, 3 and then 4) and does not receive any additional information. The agent receives a bonus reward of +50 for completing the task. In order to ensure that the task is fully observable under the MLP prior, we provide it with additional information indicating how far it has solved the sequence so far (the LSTM prior does not receive this extra information).

In order to understand why the **Hard** task presents an additional challenge, consider the following scenario. Suppose in the **Easy** task, the agent were to randomly stumble upon target 2 when it is selected (an event that occurs 25% of the time). It would get a reward and have some signal to learn about the task. Now imagine the same scenario but under the conditions of the **Hard** task. The agent only ever receives a reward for visiting target 2 if it has previously visited target 1 within the episode. This requirement only gets more complex for later targets. In order to receive a reward for visiting target 3, the last two targets that the agent visited must be targets 1 and 2 in that order. This version of the task thus presents a far more complex exploration problem with sparse learning signal. And this is precisely why a *behavior prior* could be useful for learning in this setting. The target sequence in the **Hard** task is more likely to be generated under the 2nd-order dynamics that underpin the **Easy** task; therefore a prior that can capture these dynamics would provide an advantage during transfer.

We consider two kinds of models for our *behavior priors* - one that only uses a Multilayer Perceptron (MLP) architecture and another that incorporates an LSTM (Long short-term memory: Hochreiter and Schmidhuber, 1997) and can thus learn to model temporal correlations. In both cases the policy and critic include LSTMs in order to render the tasks fully observable and solve them. Our goal is to analyze the kinds of behaviors each of these models can learn and understand the effect this has on learning and transfer.

In both variants of the task, the agent receives the coordinates of all of the targets as well as its own location and velocity information. The *behavior priors* also receive this



(a) Exploration with a randomly initialized Gaussian policy.

(b) Exploration with the learnt *behavior prior* model (LSTM).

Figure 6: **Visualization of exploration using a *behavior prior*** . Each colored dotted line represents a trajectory generated by rolling out a policy in the task.

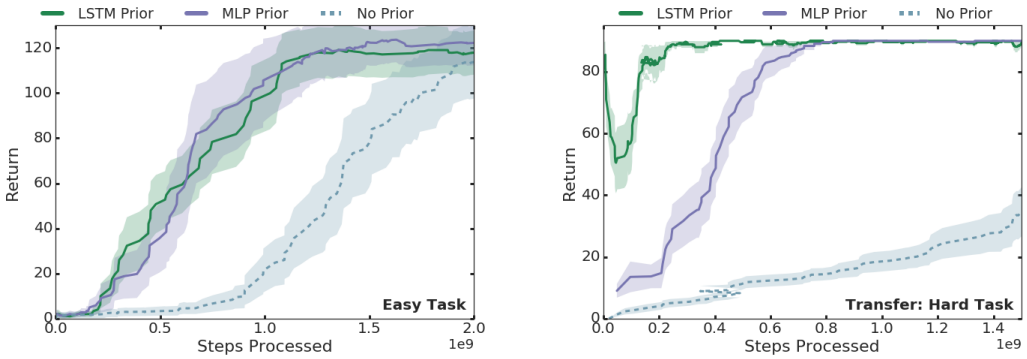
information but do not get the task specific information described above. Since the state information provided to the *behavior prior* includes velocity, even an MLP prior can learn some temporally extended behavior like visiting targets. In order to do this, the prior can use it's heading to infer the target that it is likely to visit next. It can also learn to slow down when close to that target and to randomly sample a new heading after reaching it. However since the task is setup in such a way that no prediction about the next target can be made when given only the last visited target, there is a limitation to how much of the task structure such a prior can uncover.

For this section, we consider a version of this task with the 2-DoF pointmass body. The target locations are randomized at the start of each episode and the agent is given a reward of +10 each time it visits the correct target in the sequence. For the transfer experiment, since there is no additional information beyond what the prior receives, we use the prior to initialize the weights of the agent policy. Each training curve for transfer averages across 10 seeds (2 seeds for each of the 5 seeds from training).

We illustrate our results in Figure 7. Figure 7a shows that learning with either prior leads to faster convergence on this task. This is in line with our intuition that the structure captured by the prior can guide the learning process. Results on the transfer domain are shown in Figure 7b. Our results show that learning is considerably accelerated when regularizing against either of the pretrained priors. Additionally, there is a significant improvement in learning speed when using the LSTM model. Since we do not transfer the critic and the SVG-0 algorithm relies on an informed critic to guide policy learning, we see a temporary drop in performance across all curves early in learning. In order to understand why the priors improve performance, we analyze the behaviors captured under each prior below.

4.2.1 Analysis

To understand why the *behavior priors* are useful during training and transfer we first need to clarify what behaviors they are actually modelling. In order to do this, we qualita-



(a) Learning with priors on Easy Task

(b) Transfer of priors to hard task

Figure 7: **Learning and transfer on Sequential Navigation.** Learning curves from training on the easy task and transferring to the hard task with various *behavior prior* architectures.

tively compare trajectories generated from the trained MLP prior to 'random' trajectories generated from a randomly initialized Gaussian policy as shown in Figure 6.

As the figure shows, the prior learns to model a 'goal-directed' behavior of bouncing between the different targets as opposed to the random motion of the untrained policy. As previously described, even an MLP prior is able to learn this temporally extended target visiting behavior since it can integrate over velocity information present in the state. This 'target-centric' movement behavior is useful to guide the final policy during training and transfer.

The LSTM prior exhibits goal directed behavior similar to the MLP prior but additionally models the longer-term structure of the domain. To understand this difference, we can compare the transition dynamics (the order that targets are visited) when the priors visit a specific target. In Figure 8, we plot the empirical distribution of visiting target 0 on the Y-axis against the last 2 visited targets on the X for each model. The distribution generated under the LSTM closely resembles the true underlying distribution used to generate the target sequence. The MLP prior instead learns to visit each target more or less uniformly at random. This results in a significant difference in performance during transfer where the LSTM model is more likely to generate a trajectory that visits targets in the rewarded sequence.

It is important to note that restricting the space of trajectories that the prior explores is only advantageous if that space also captures the *solution space* of the task we are ultimately interested in solving. In other words, there is a tradeoff between generality and efficacy of the prior. A prior over trajectories that is 'uniform' in that it assigns equal mass to all possible trajectories may be of limited use since it does not capture any of the underlying task structure. On the other hand, a prior that overfits to the dynamics of a specific task will not be general enough to be useful on many tasks. Understanding how the choice of model and information set affect this tradeoff is thus crucial to learn effective priors.

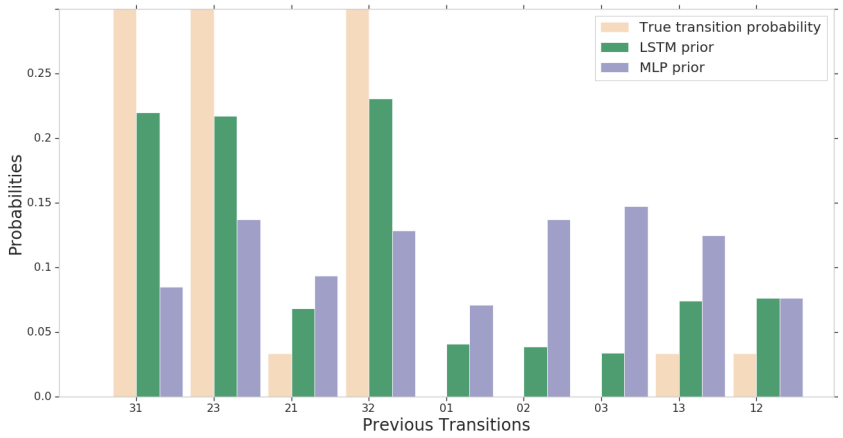


Figure 8: **Transition dynamics** generated under different *behavior prior* models to visit target 0. Each point on the x-axis indicates the indices of the last two targets visited. On the y-axis we plot the probability of visiting target 0 for each transition sequence.

5. Structured *behavior prior* models

In the discussion thus far, we have considered examples where π and π_0 use neural network models to parameterize uni-modal Gaussian distributions; a form which has shown promising results but is relatively simple in its *stochastic structure* and in the kinds of trajectory distributions it can model.

To see why such a simple model might be limiting, consider an example where the optimal action in a particular state depends on some unobserved context; for instance an agent may have to go left or right based on whether a light is on. If we were to model this on a continuous spectrum going 'left' might correspond to an action value of -1 and going 'right' to a value of +1. A prior without access to the conditioning information (the light in our example) would need to model the marginal distribution, which, in this case, is bi-modal (with one mode at -1 and the other at +1). It is impossible to capture such a distribution with a uni-modal Gaussian. Due to nature of the objective (cf. Equation 8) the prior will have to cover both modes. This might lead to undesirable solutions where the mean (and mode) of the prior is centered on the 'average' action of 0.0, which is never a good choice.

While this example may seem contrived, it does illustrate that there may be situations where it will be important to be able to model more complex distributions. In fact, as discussed in Section 3, the optimal prior in the multi-task setting is a mixture of the solutions to individual task (cf. Equation 10). This may not be modelled well by a Gaussian.

In this section we will develop the mathematical tools required to work with more complex models. Our starting point, is to once again turn to the probabilistic modeling literature to borrow ideas from the space of latent variable models. Latent or 'unobserved' variables in a probabilistic model can be used to increase capacity, introduce inductive biases and model complex distributions. In the most general form we can consider directed

latent variable models for both π_0 and π of the following form:

$$\pi_0(\tau) = \int \pi_0(\tau|y)\pi_0(y)dy, \tag{15}$$
$$\pi(\tau) = \int \pi(\tau|z)\pi(z)dz, \tag{16}$$

where the unobserved 'latents' y and z can be time varying, e.g. $y = (y_1, \dots y_T)$, continuous or discrete, and can exhibit further structure.

Above, we have motivated latent variables in the prior π_0. To motivate *policies* π with latent variables, we can consider tasks that admit multiple solutions that achieve the same reward. The KL term towards a suitable prior can create pressure to learn a distribution over solutions (instead of just a single trajectory), and augmenting π may make it easier to model these distinct solutions (e.g. Hausman et al., 2018).

Policies with latent variables have also been considered within the RL community under the umbrella of hierarchical reinforcement learning (HRL). While the details vary, the motivation is often to model higher level abstract actions or 'options' (z_ts in our formulation). These are often temporally consistent, i.e. the abstract actions may induce correlations between the primitive actions. This may be beneficial, for instance for exploration and credit assignment on long horizon tasks like the ones we considered in Section 4.2. The focus of the present work is for modeling *behavior priors* π_0, i.e. we are interested in models that can provide *richer descriptions of behavior*. We discuss the relations between our method and some work from HRL in Section 7.

Unfortunately, a direct use of the formulation from Equations (15) and (16) can be problematic. It is often difficult to compute the KL term in Equation (5): $\mathrm{KL}[\pi(\tau)||\pi_0(\tau)]$ when π and π_0 are marginal distributions of the form outlined in Equation (16) and Equation (15) respectively. This results in the need for approximations and a number of choices and practical decisions stem from this consideration. We focus the discussion here on a simple and practical formulation that is the focus of our experimental analysis in Section 6. A more detailed discussion of the effects of various modeling choices is deferred to Section 7.

5.1 Simplified Form

In this work we focus on a formulation that allows for continuous latent variables in *both* π_0 and π which is both tractable and practical to implement. The formulation allows π and π_0 to model richer (state-conditional) action distributions, to model temporal correlations, and provides flexibility for partial parameter sharing between π_0 and π.

We consider a continuous latent variable z_t which has the same dimension and semantics in π and π_0. We divide π into higher level (HL) $\pi^H(z_t|x_t)$ and lower level (LL) $\pi^L(a_t|z_t, x_t)$ components. We can then derive the following bound for the KL (see Appendix D.5 for proof):

$$\begin{aligned}\mathrm{KL}[\pi(a_t|x_t)||\pi_0(a_t|x_t)] \leq \mathrm{KL}[\pi^H(z_t|x_t)||\pi_0^H(z_t|x_t)] \\ + \mathbb{E}_{\pi^H(z_t|x_t)}[\mathrm{KL}[\pi^L(a_t|z_t, x_t)||\pi_0^L(a_t|z_t, x_t)]],\end{aligned} \tag{17}$$

which can be approximated via Monte Carlo sampling and we now define x_t such as to contain previous z_ts as well. This upper bound effectively splits the KL into two terms - one between the higher levels $\pi^H(z_t|x_t)$ and $\pi_0^H(z_t|x_t)$ and the other between the lower levels $\pi^L(a_t|x_t, z_t)$ and $\pi_0^L(a_t|x_t, z_t)$.

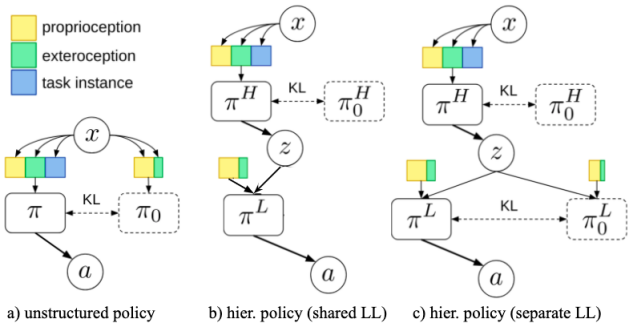


Figure 9: **Training setup for structured and unstructured models.**

5.2 Modeling considerations

In the remainder of this section, we describe modeling considerations that are needed in order to implement the KL-regularized objective using Equation (17).

Regularizing using information asymmetry In Section 4.1.1, we demonstrated that information constraints can force the prior to generalize across tasks or different parts of the state space. This intuition also applies to the different levels of a hierarchical policy. Information asymmetry between the higher level prior and policy in Equation (17) has two effects: it regularizes the higher level action space making it easier to sample from; and it introduces an information bottleneck between the two levels. The higher level thus pays a 'price' for every bit it communicates to the lower level. This encourages the lower level to operate as independently as possible to solve the task. By introducing an information constraint on the lower level, we can force it to model a general set of skills that are modulated via the higher level action z in order to solve the task. Therefore, the tradeoff between generality and specificity of *behavior priors* mentioned in Section 4 would now apply to the different layers of the hierarchy. We will explore the effect of this empirically in Section 5.2.

Partial parameter sharing An advantage of the hierarchical structure is that it enables several options for partial parameter sharing, which when used in the appropriate context can make learning more efficient. For instance, sharing the lower level controllers between the agent and default policy, i.e. setting $\pi^L(a_t|z_t, x_t) = \pi_0^L(a_t|z_t, x_t)$ reduces the number of parameters to be trained and allows skills to be directly reused. This amounts to a hard constraint that forces the KL between the lower levels to zero. With that, the objective from Equation (11) now becomes:

$$\mathcal{L}(\pi, \pi_0) \geq \mathbb{E}_\tau \left[\sum_{t \geq 1} \gamma^t r(s_t, a_t) - \alpha \gamma^t \mathrm{KL}[\pi(z_t|x_t) || \pi_0(z_t|x_t)] \right], \tag{18}$$

where the KL divergence is between the high-level prior and policy defined only on abstract actions z_t. We illustrate the various approaches to training structured and unstructured models with shared and separate lower levels in Figure 9.

5.3 Algorithmic considerations

Off-policy training of hierarchical policies adds complexity in the estimation of the state-action value $Q(x_t, a_t, z_t)$, the bootstrap value $V(x_t, z_t)$ and for learning the policy itself. We present a modified version of Algorithm 1 for hierarchical policies in Algorithm 2, where the modifications required for learning *behavior priors* are in blue and those for training latent variable models are in purple. The algorithm assumes the latent z is sampled at every timestep. There are additional complexities when the latent is held fixed across multiple timesteps (e.g. similar to Heess et al., 2016), which we describe in greater detail below.

Critic update: We adapt the retrace algorithm from Munos et al. (2016) for off-policy correction to hierarchical policies. At timestep t, the update is given by:

$$Q(x_t, a_t, z_t) = Q(x_t, a_t, z_t) + \sum_{s \geq t} \gamma^{s-t} \left(\prod_{i=t}^{s} c_i \right) \left(r_s + \gamma V_{s+1} - Q(x_s, a_s, z_s) \right),$$

$$c_i = \lambda \min \left(\frac{\pi^H(z_i|x_i)\pi^L(a_i|x_i, z_i)}{\mu(a_i|x_i)}, 1 \right).$$

Note that the state-action value function Q also depends on the latent z_t here. This is because in the general case, z_t may be sampled infrequently or held constant across multiple timesteps. In other words, the value of *future* states x_s where $s > t$ could depend on the current latent z_t and must be accounted for.

This may also hold for the state-value function based on whether z_t is sampled in state x_t. Specifically, in states where z_t is sampled:

$$V(x_t) = \mathbb{E}_{z \sim \pi^H(.|x_t), a \sim \pi^L(.|x_t, z)} Q(x_t, a, z),$$

and in other states:

$$V(x_t, z_t) = \mathbb{E}_{a \sim \pi^L(.|x_t, z_t)} Q(x_t, a, z_t).$$

Additionally, in our adaptation of the retrace estimator, we do not consider the latent z_t in the behavior policy μ that was used to generate the data on the actors. Since the latent does not affect the environment directly, we can estimate the importance weight as $\mu(a_t|x_t)$ (as a way to reduce the variance of the estimator). Finally we include the KL term in the computation of the value for bootstrapping as described in Section 3.

Policy update: In the hierarchical formulation there are different ways to adapt the policy update from the SVG-0 algorithm (Heess et al., 2015).

One way to compute the update for π^H is through the partial derivative of Q:

$$\nabla_{\theta^H} \mathbb{E}_\pi \left[Q(x, a, z) \right] = \mathbb{E}_{\zeta(\eta)} \left[\frac{\partial Q}{\partial z} \nabla_{\theta^H} \pi^H(z|x, \eta) \right],$$

where $z \sim \pi^H(z|x)$ is reparameterized as $z = \pi^H(z|x, \eta)$, for some noise distribution $\eta \sim \zeta(.)$. In practice we found that this leads to worse performance and opted instead to compute the policy update as:

$$\nabla_\theta \mathbb{E}_\pi \left[Q(x, a, z) \right] = \mathbb{E}_{\zeta(\eta), \rho(\epsilon)} \left[\frac{\partial Q}{\partial a} \nabla_{\theta^H} \pi(z|x, \eta) \nabla_{\theta^L} \pi(a|s, z, \epsilon) \right],$$

where we introduce two sources of noise $\eta \sim \zeta(.)$ and $\epsilon \sim \rho(.)$. Finally the policy update also includes the KL terms from the objective as in Algorithm 1 using the bound from Equation (18).

Prior update: The update for the *behavior prior* is exactly as in Algorithm 1 except that here we use the bound from Equation (17) to compute the KL used for distillation. Intuitively, this can be thought of as two separate distillation steps - one for the HL and the other for the LL. The objective for the prior can thus be written as:

$$L_{\pi_0} = \sum_{i=t}^{t+K-1} \mathrm{KL}\left[\pi^H(.|x_i) \| \pi_0^H(.|x_i^D)\right] + \mathrm{KL}\left[\pi^L(.|x_i, z_i) \| \pi_0^L(.|x_i^D, z_i)\right]. \tag{19}$$

Note that when the parameters for the lower level are shared between the policy and *prior*, the objective only contains terms pertaining to the higher level π_0^H.

6. Experiments with structured priors

In this section, we analyze the performance of structured and unstructured *behavior priors* on a range of control domains. Our goal is to understand if hierarchy offers any advantages over unstructured architectures. In particular, we study the potential benefits of richer distributions and partial parameter sharing (between π and π_0) during transfer. We begin with an analysis similar to the one in Section 4.1.1 to study how information asymmetry in modular policies affect the kinds of behaviors learnt. Then we compare training and transfer performance of structured and unstructured models across a range of 'Locomotion' and 'Manipulation' tasks in Section 6.1 and the 'Sequential Navigation' tasks in Section 6.4.

Unless otherwise specified, all the structured priors we consider for this part include a shared lower level policy as described in Section 5. During transfer, this enables the reuse of skills captured within the lower level and restricts the parameters that are trained to those just in the higher level policy: π^H.

We primarily consider two models for the higher level prior π_0^H: **Independent isotropic Gaussian.** $\pi_0^H(z_t|x_t) = \mathcal{N}(z_t|0, 1)$, where abstract actions are context independent or **AR(1) process.** $\pi_0^H(z_t|x_t) = \mathcal{N}(z_t|\alpha z_{t-1}, \sqrt{1-\alpha^2})$, a first-order auto-regressive process with a fixed parameter $0 \leq \alpha < 1$ chosen to ensure the variance of the noise is marginally distributed according to $\mathcal{N}(0, I)$. We include this model since it showed promising results in Merel et al. (2019) and could allow for temporal dependence among the abstract actions since the choice of action is conditioned on the previously sampled one. For the 'Sequential Navigation' tasks considered in Section 6.4, we consider two architectural variants of the prior with and without memory: one with an MLP (*Hier. MLP*); and one with an LSTM (*Hier. LSTM*).

6.1 Effect of Information Asymmetry

As described in Section 5.2, we expect information asymmetry between the different levels of the hierarchy to play an important role in shaping the learning dynamics; an idea which we demonstrate empirically in this section.

Consider the *Locomotion and Manipulation* task (see Table 1) from Section 4.1.1 where an agent must move a box to a target (*Manipulation*) and then move to another target

Algorithm 2 SVG(0) with experience replay for hierarchical policy

Flat policy: $\pi_\theta(a_t|\epsilon_t, x_t)$ with parameter θ
HL policy: $\pi^H_\theta(z_t|x_t)$, where latent is sampled by reparameterization $z_t = f^H_\theta(x_t, \epsilon_t)$
Behavioral Priors: $\pi^H_{0,\phi}(z_t|x_t)$ and $\pi^L_{0,\phi}(a_t|z_t, x_t)$ with parameter ϕ
Q-function: $Q_\psi(a_t, z_t, x_t)$ with parameter ψ

Initialize target parameters $\theta' \leftarrow \theta$, $\phi' \leftarrow \phi$, $\psi' \leftarrow \psi$.
Hyperparameters: α, α_H, β_π, β_{π_0}, β_Q
Target update counter: $c \leftarrow 0$
Target update period: P
Replay buffer: $\mathcal{B}$

for $j = 0, 1, 2, \ldots$ **do**
Sample partial trajectory $\tau_{t:t+K}$ generated by behavior policy μ from replay B:
$\tau_{t:t+K} = (s_t, a_t, r_t, ..., r_{t+K})$,

for $t' = t, \ldots t+K$ **do**
$\epsilon_{t'} \sim \rho(\epsilon)$, $z_{t'} = f^H_\theta(x_{t'}, \epsilon_{t'})$
▷ Sample latent via reparameterization
$\hat{\mathrm{KL}}_{t'} = \mathrm{KL}\left[\pi^H_\theta(z|x_{t'}) \| \pi^H_{0,\phi'}(z|x_{t'})\right] + \mathrm{KL}\left[\pi^L_\theta(a|z_{t'}, x_{t'}) \| \pi^L_{0,\phi'}(a|z_{t'}, x_{t'})\right]$ ▷ Compute KL
$\hat{\mathrm{KL}}^{\mathcal{D}}_{t'} = \mathrm{KL}\left[\pi^H_\theta(z|x_{t'}) \| \pi^H_{0,\phi}(z|x_{t'})\right] + \mathrm{KL}\left[\pi^L_\theta(a|z_{t'}, x_{t'}) \| \pi^L_{0,\phi}(a|z_{t'}, x_{t'})\right]$
▷ Compute KL for Distillation
$\hat{\mathrm{H}}_{t'} = \mathbb{E}_{\pi_\theta(a|\epsilon_{t'}, x_{t'})}[\log \pi_\theta(a|\epsilon_{t'}, x_{t'})]$ ▷ Compute action entropy

$\hat{V}_{t'} = \mathbb{E}_{\pi_\theta(a|\epsilon_{t'}, x_{t'})}\left[Q_{\psi'}(a, z_{t'}, x_{t'})\right] - \alpha \hat{\mathrm{KL}}_{t'}$ ▷ Estimate bootstrap value
$\hat{c}_{t'} = \lambda \min\left(\frac{\pi_\theta(a_{t'}|\epsilon_{t'}, x_{t'})}{\mu(a_{t'}|x_{t'})}\right)$ ▷ Estimate traces Munos et al. (2016)
$\hat{Q}^R_{t'} = Q_{\psi'}(a_{t'}, z_{t'}, x_{t'}) + \sum_{s \geq t'} \gamma^{s-t'} \left(\prod_{i=s}^{t'} \hat{c}_i\right)\left(r_s + \gamma \hat{V}_{s+1} - Q_{\psi'}(a_s, z_s, x_s)\right)$
▷ Apply Retrace to estimate Q targets Munos et al. (2016)

end for

$\hat{L}_Q = \sum_{i=t}^{t+K-1} \|\hat{Q}^R_i - Q_\psi(a, z_i, x_i)\|^2$ ▷ Q-value loss

$\hat{L}_\pi = \sum_{i=t}^{t+K-1} \mathbb{E}_{\pi_\theta(a|\epsilon_i, x_i)} Q_{\psi'}(a, z_i, x_i) - \alpha \hat{\mathrm{KL}}_i + \alpha_{\mathrm{H}} \hat{\mathrm{H}}_i$ ▷ Policy loss

$\hat{L}_{\pi^H_0} = \sum_{i=t}^{t+K-1} \hat{KL}^{\mathcal{D}}_i$ ▷ *Prior* loss

$\theta \leftarrow \theta + \beta_\pi \nabla_\theta \hat{L}_\pi$ $\quad \phi \leftarrow \phi + \beta_{\pi^H_0} \nabla_\phi \hat{L}_{\pi^H_0}$ $\quad \psi \leftarrow \psi - \beta_Q \nabla_\psi \hat{L}_Q$
Increment counter $c \leftarrow c + 1$
if $c > P$ **then**
Update target parameters $\theta' \leftarrow \theta$, $\phi' \leftarrow \phi$, $\psi' \leftarrow \psi$
$c \leftarrow 0$
end if
end for

(*Locomotion*). Here we will consider structured models with a shared lower level policy and an isotropic Gaussian higher level prior, and we further ensure that the network architectures used across all priors are of comparable size. [7]

7. Adding hierarchical structure automatically increases the depth and complexity of the models being considered. Complex models in turn can affect the relative speeds of the agent when acting and for learning which could affect the distribution of data used for training. In order to account for these discrepancies, whenever a comparison is made between structured and non-structured models, as far as possible we ensure that the network sizes of both are of comparable size and complexity. This is important as it allows us to tease apart the relative advantage of the structured formulation while keeping all other factors constant. A full description of the models and training setup used for all experiments can be found in Appendix F.

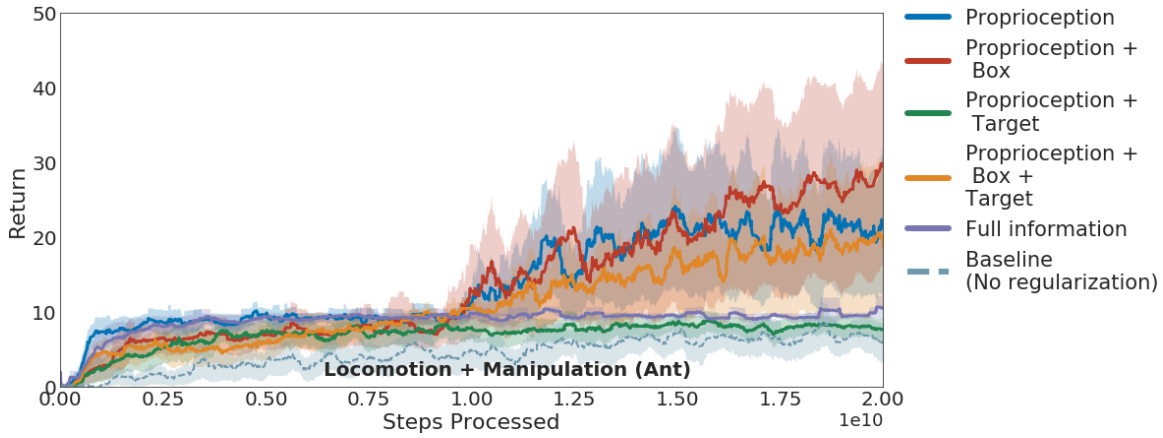


Figure 10: **Effect of information asymmetry** with structured models on the 'Locomotion and Manipulation' task.

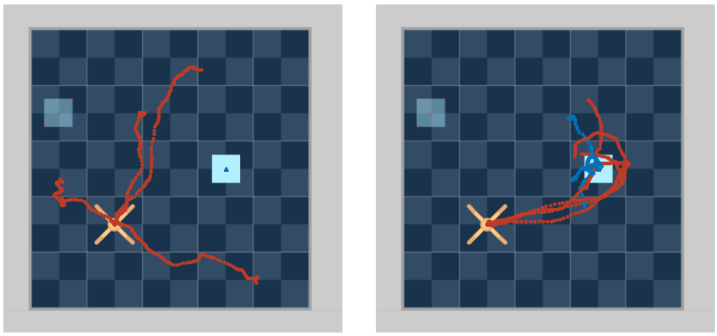


Figure 11: **Effect of information asymmetry on exploration** A visualization of the motion of the ant (red) and the box (blue) under random latent samples fed to a lower level policy with a) just proprioceptive information and b) proprioceptive and box information.

As Figure 10 illustrates, we see a similar effect of information asymmetry here as in Section 4.1.1. Specifically, *behavior priors* with partial information perform better than those with access to full information. As discussed in Section 4, this task includes a local optimum where agents that learn to solve the easier 'Locomotion' component of the task may not go on to solve the full task. Priors with access to only partial information can help regularize against this behavior.

To qualitatively understand these results better, we perform an analysis similar to the one considered in Section 4 (see Figure 3). Recall that there we visualized the behaviors by generating trajectory rollouts of the different priors. In the present discussion, we consider

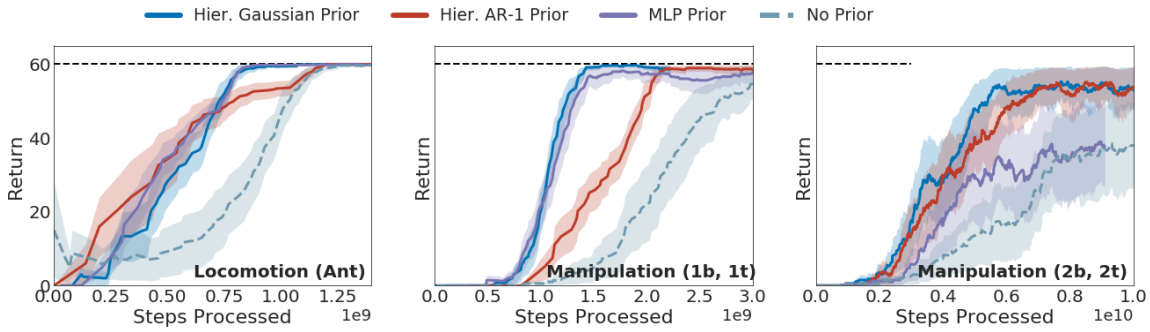


Figure 12: **Learning performance** with different *behavior priors* on the *Locomotion (Ant)*, *Manipulation (1b, 1t)* and *Manipulation (2b, 2t)* tasks.

the information asymmetry that exists between the higher and lower level of the policy. The higher level sees all the information pertaining to the task while the lower level has access to a subset of this information and the latent variable z. We can thus generate trajectories from the lower level by sampling different values for the latent variable z from some starting state.

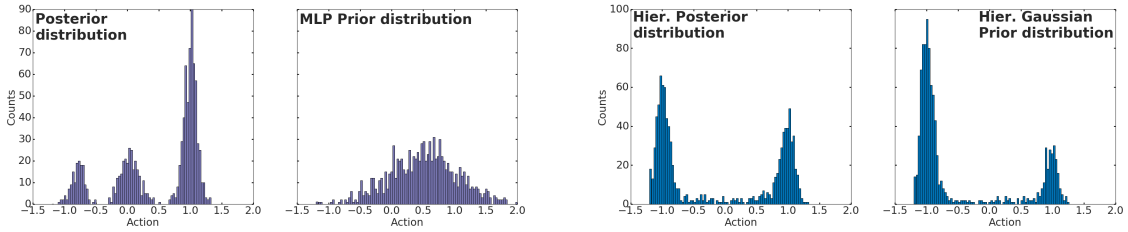
We present our results in Figure 11. Qualitatively, the trajectories are very similar to those observed in Figure 3. The prior with only proprioceptive information learns to model movement behaviors in different directions. In contrast, the prior with access to the box location shows a more goal-directed behavior of pushing the box. However, compared to the unstructured priors of Section 4, here the behavior can be modulated through z. This presents a new approach to utilising the prior - rather than distilling knowledge through the KL-regularized objective, we can instead learn a controller in latent space to directly reuse the lower level. We analyze the advantage of this approach in the next section.

6.2 Empirical comparison of structured and unstructured priors

Next, we empirically compare training and transfer performance of structured and unstructured priors on the tasks of Section 4.1.2.

Training: We compare performance on three tasks: *Locomotion (Ant)*, *Manipulation (1 box, 1 target)* and *Manipulation (2 boxes, 2 targets)* (refer to Table 1 for a short summary and the information asymmetry used).

We illustrate our results in Figure 12. They are consistent with Section 4: *behavior priors* with partial information accelerate learning across all tasks. Furthermore, we find that hierarchical models tend to perform better than the unstructured ones on more complex tasks (*Manipulation (2b, 2t)*). In contrast to the findings of Merel et al. (2019), we find that the AR-1 prior performs worse than the Gaussian prior. The AR-1 prior allows to model correlations across time. However, this added flexibility does not seem to offer an advantage in our setting.



(a) (b)

Figure 13: **Empirical distributions** for different goals under the posterior and prior distributions for a) MLP and b) Hier. Gaussian on the *Manipulation (1 box, 1 target)* task. Each plot represents the empirical distribution for an action dimension marginalized over different goals x^G for a fixed goal-agnostic state x^D. The prior only has access to x^D.

A key motivation behind the introduction of the structured models was their ability to better capture complex distributions. In order to understand whether this effect might explain some of the performance differences we observed, we computed the average KL divergence between the policy and *behavior prior* on the *Manipulation (1 box, 1 target)* task across 5 seeds and 100 episodes. Values of 2.64 and 11.35 for the Hier. Gaussian and MLP prior are consistent with the idea that the latent variable model provides a better prior over the trajectory distribution and thus allows for a more favorable trade-off between KL and task reward.

Figure 13 helps to demonstrate this point qualitatively. In the figure, for a fixed x^D (where x^D is the 'goal agnostic' information subset from Section 3), we sample different values of x^G (the 'goal specific' information) from the initial state distribution and generate samples under the posterior policy. This represents the empirical action distribution for the policy marginalized over different goals, i.e. the distribution that the prior is required to model. We show this distribution for a particular action dimension with the MLP model in Figure 13a.

We observe multiple modes in the empirical action marginal, presumably corresponding to different settings of the goal x^G. This marginal cannot be modeled by a unimodal Gaussian prior. We observe a mode-covering behavior where the prior spreads probability mass across the different modes in line with the intuition described in Section 3. In contrast, the hierarchical prior shown in Figure 13b does a better job of capturing such multi-modal distributions since the action distribution π_0^L is conditioned on samples from the higher level prior π_0^H. These results are consistent with the lower KL between prior and posterior for structured priors. And as we discussed at the start of Section 5, there may be cases where this property is desirable and may significantly affect performance.

Transfer: Next, we study the effectiveness of priors for transferring learnt behaviors. A policy is trained for the transfer task while regularizing against a fixed pre-trained prior. We consider two transfer tasks: *Manipulation (1box, 3targets)* where we transfer from a

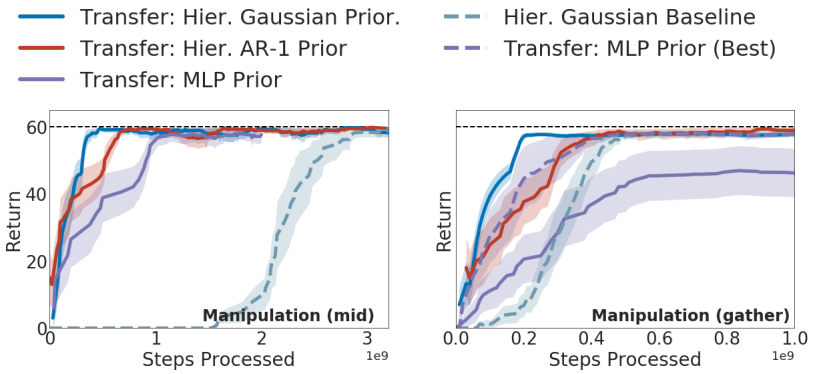


Figure 14: **Transfer performance** with various *behavior priors* on *Manipulation (1b, 3t)* and *Manipulation (2b, gather)*.

task with fewer targets ((*Manipulation (1box, 1targets)*) and *Manipulation (2box gather)* where we transfer from a task with a different objective (*Manipulation (2box, 2targets)*).

We show our results in Figure 14. We observe a significant boost in learning speed when regularizing against structured *behavior priors* across all transfer tasks. The advantage of hierarchical priors can be explained by the fact that the behaviors captured by the lower level policies are directly shared during transfer and do not need to be distilled through the KL term in the objective.[8]

6.3 Separate low level modules

All of the structured models we have considered so far used a lower level policy that was shared between the *behavior prior* π_0 and policy π. This can be advantageous for transfer since it allows us to directly reuse the lower level prior. It does amount, however, to a hard constraint on the lower level which remains unchanged during transfer and cannot adapt to the demands of the target task. As an alternative, we can use a separate lower level controller for the policy as shown in Figure 9c. This corresponds to the full KL decomposition from Equation (17). In this section, we consider an example where the separate lower level provides an advantage.

We consider a transfer task *Locomotion (Gap)* where an agent must learn to *jump* over a variable sized gap that is always at the same location in a straight corridor. The agent is rewarded at each time step for its forward velocity along the corridor. Locomotive behaviors from the *Locomotion (Ant)* task are sufficient to jump across small gaps, but will not fare well on the larger gaps.

We demonstrate this point in Figure 15. The model with a separate lower level performs asymptotically better than one with a shared lower level. Unlike a shared controller, the

8. Surprisingly, we observed that the *MLP prior* slowed learning on the *Manipulation (2b, gather)* task. Further analysis showed that some of the priors used for this task were of poor quality; this is visible from the performance on the training task in Figure 12c. Discarding these leads to improvement in performance shown by the dotted purple line in Figure 14.

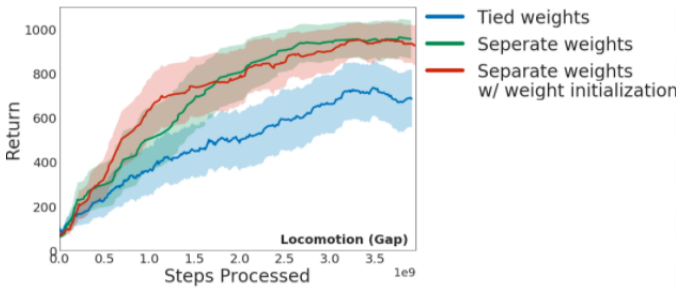


Figure 15: **Skill adaptation for hierarchical models** using separate lower level controllers on the *Locomotion (Gap)* task.

separate lower level controller adapts to the task by learning 'jumping' behaviors for all gap lengths.

6.4 Sequential Navigation

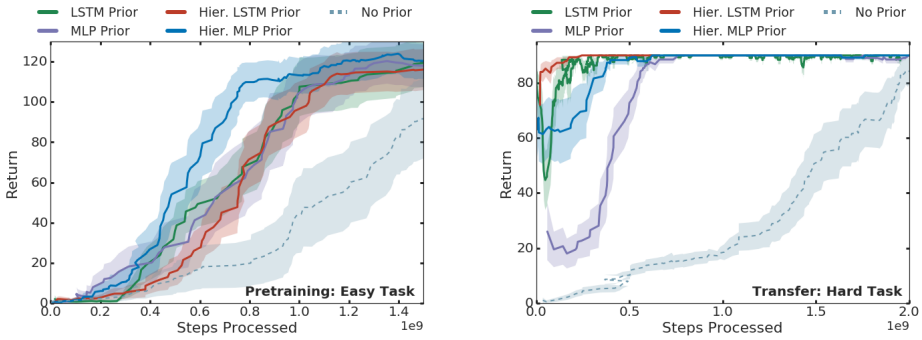
In Section 4.2, we discussed how *behavior priors* can model behaviors at multiple temporal scales and how modeling choices affect the nature of the behaviors learnt. Now, we will expand this analysis and consider hierarchical model architectures.

We revisit the Sequential Navigation task introduced in Section 4.2 which requires an agent to visit a sequence of targets in a history-dependent order. This task involves structure at various levels - goal-directed locomotion behavior and long-horizon behavior due to some target sequences being more likely to be generated than others. We consider two versions of this task with two different bodies: the 2-DoF pointmass and the 8-DoF ant walker. The latter increases the complexity of the low-level control problem while the high-level task structure remains unchanged.

The policy consists of the higher level component $\pi^H(z|x)$ which operates in the space of latent z's and the lower level component $\pi^L(a|x,z)$ which produces primitive actions. We share the lower level policy and lower level prior as described in Section 5.

It is worth considering the information asymmetry used in the design of these experiments. As described in Section 4.2 the information communicated to the agent can be split into: task-specific information (like the location of the target) and body specific proprioceptive information (including the location and velocity of the agent). In our setup, the higher level policy has access to all information while the higher level prior and the shared lower level are conditioned on just proprioceptive information. There is thus an information asymmetry between the higher and lower level policy and also between the higher level policy and prior.

The asymmetry between the higher and lower level policy allows the latent space z to capture high level information present in the task; for instance which target to reach next. The precise control in the primitive action space of the body can be left to the lower level. Similarly, the higher level prior only needs to model task structure present in the latent space z. Depending on the nature of the prior, we can capture task level information at various levels of granularity (for example, a prior with memory may be able to capture the



(a) Learning with structured priors on the easy Task

(b) Transfer of structured priors to hard task

Figure 16: **Learning and transfer** with structured *behavior priors* on the 'Sequential Navigation' task.

long term dynamics present in the task). The potential advantage of this separation is that it could make it easier to learn temporal structure present in the task, especially for more complex bodies like the Ant walker.

Since we would like to study various priors that can model temporal correlations across time, we compare two higher level priors in our experiments: one with an MLP (*Hier. MLP*) and one with an LSTM (*Hier. LSTM*). Note that unlike the isotropic Gaussian and AR-1 priors considered in previous sections, here the higher level is a parameteric prior trained according to Equation (19). Full details with the networks and training setup used for these experiments can be found in Appendix F.

Training: We first consider the effect of different priors during training. Results are shown in Figure 16. As in the previous section, we find that the structured priors improve learning speed on this task. We also observe a slight advantage of using the *Hier. MLP* prior in this case. This may be explained by a tradeoff between the richness of the prior and the speed at which it can be learned.

Transfer: We compare the performance of structured priors during transfer in Figure 16b. We find that the priors that model temporal structure in the task (LSTM and Hier. LSTM) perform better here as they are more likely to generate the rewarding sequence. We analyze this point quantitatively in Table 2. The table records the average returns generated on the transfer task over a 100 episodes when rolling out the trained priors. Priors with memory (LSTM and Hier. LSTM) are more likely to visit the rewarding sequence and thus generate much higher returns during early exploration. The distribution of target sequences visited when executing these priors also has a lower KL divergence against the true underlying dynamics of the task which is illustrated qualitatively in Figure 17 and quantitatively in Table 2. We also find that structured priors slightly outperform their unstructured counterparts. As argued above this may be explained by the fact that the hierarchical model allows to transfer the low-level behavior directly.

Model	Hier. LSTM	Hier. MLP	LSTM	MLP
Return	**81.28**	43.82	**82.00**	33.94
KL	**0.262**	0.760	**0.191**	0.762

Table 2: **Average return and KL**. For each pre-trained model, we compute statistics by generating trajectories under the *prior* on the hard task (Averaged across a 100 episodes for 5 seeds).

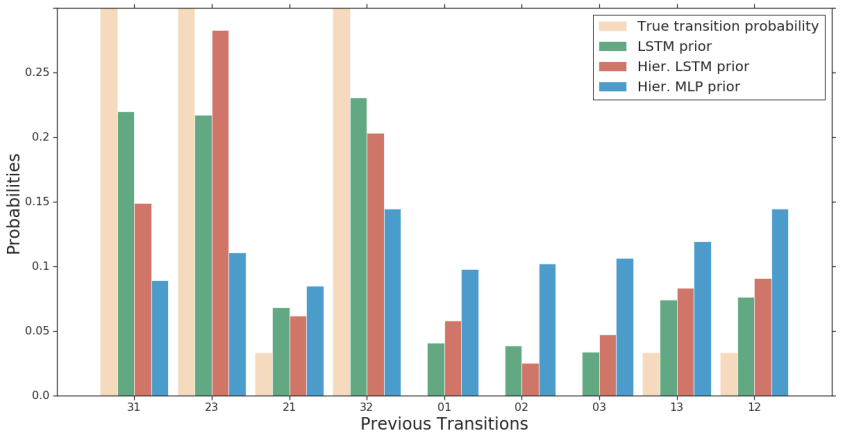
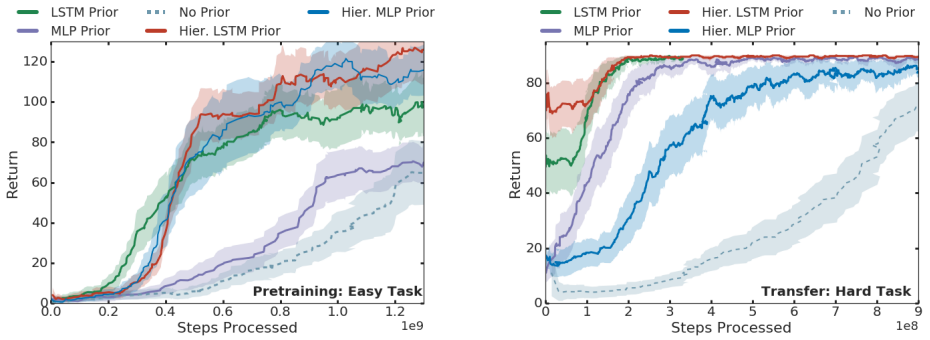


Figure 17: **Transition dynamics for first target** generated under structured *behavior prior* models. Each point on the x-axis indicates the indices of the last two targets visited. On the y-axis we plot the probability of visiting the target for each transition sequence.



(a) Learning with structured priors on easy task (Ant)

(b) Transfer of structured priors to hard task (Ant)

Figure 18: **Learning and transfer** on *Sequential Navigation (Ant)*

A possible advantage of structured models is their ability to model different aspects of the behavior in different layers of the hierarchy. The *Sequential Navigation (Ant)* task increases the complexity of the motor control problem but keeps the high-level task requirements the same. This version of the task tests whether a split between longer term task behaviors in the higher level and primitive movement behaviors in the lower level could prove advantageous. We restrict the targets to be generated on a smaller grid surrounding the agent in order to speed up training. We show our results in Figure 18. The trend that is consistent across both training and transfer is that priors with memory (LSTM and Hier. LSTM) learn faster than those without (MLP and Hier. MLP). During training, the Hier. MLP prior seems to learn faster than the MLP prior but the reverse holds true for transfer. We did not find an advantage of using structured priors on this task and the reasons for this are not immediately clear. In general we found that training with parameteric latent priors (Hier. MLP and Hier. LSTM) can sometimes be unstable and care must be taken when initializing the weights for the policies (more details in Appendix F).

Overall through the results of this section and Section 4, we have demonstrated that *behavior priors* with information and modeling constraints can improve learning speed on a range of tasks. Moreover, structured *priors* can provide an additional benefit when transferring to new domains. More generally, *behavior priors* are a way to introduce inductive biases into the learning problem; a property that is likely to become increasingly important for many real world RL tasks.

7. Discussion and Related work

In this section we discuss the connections between our work and various related strands of research. We will aim to tie the discussion to the objective introduced in Equation (5) and show how various algorithmic choices and modeling assumptions for π and π_0 relate to established models in the literature.

7.1 Entropy regularized RL and EM policy search

The entropy regularized RL objective, also known as maximum entropy RL, which was considered by Todorov (2007) and later by Toussaint (2009) (also see Ziebart, 2010; Kappen et al., 2012) has recently gained resurgence in deep reinforcement learning (e.g. Fox et al., 2016; Schulman et al., 2017a; Nachum et al., 2017; Haarnoja et al., 2017; Hausman et al., 2018; Haarnoja et al., 2018b). As noted in Section 3, this objective is a special case of the KL regularized objective of Equation (5) where we set the prior π_0 to be a uniform distribution over actions. This encourages the policy π to be stochastic which is often good for exploration, has been shown to have a beneficial effect on the optimization landscape (Ahmed et al., 2019) and may lead to more robust policies in practice (Haarnoja et al., 2018b). As discussed in Section 4, this choice of reference distribution is general in some sense but unlikely to be good choice in all cases.

A related family of algorithms fall under the category of *expectation maximization* (EM) policy search algorithms (e.g. Peters et al., 2010; Toussaint and Storkey, 2006; Rawlik et al., 2013; Levine and Koltun, 2013; Levine et al., 2016; Montgomery and Levine, 2016; Chebotar et al., 2016; Abdolmaleki et al., 2018b). The fundamental idea behind this line of work is to cast policy search as an alternating optimization problem, similar to the classic EM algorithm (Dempster et al., 1977). In some cases this can be motivated from the objective in Equation (5) by considering the same form for both π and π_0. Then, since the form of π_0 is not restricted (e.g. via model capacity or information constraints), the KL term in the objective effectively acts as a trust region that constrains the optimization procedure in each step but *does not* constrain the form of the final solution. This has been shown to be very successful for a large number of problems (e.g Schulman et al., 2015, 2017b; Wang et al., 2017a; Abdolmaleki et al., 2018b). Yet, even though there are algorithmic similarities and even though our results may also benefit from an effect very similar to a trust region, the motivation behind this line of work is often quite different from the ideas presented in the present paper.

The KL regularized objective has also seen application in the completely offline or Batch Reinforcement Learning (Batch RL) setting. This is often the case in applications where some prior data exists and the collection of new online data is prohibitively expensive or unsafe. Conventional off policy algorithms like DDPG or SVG typically do not fare well in this regime as demonstrated e.g. by Fujimoto et al. (2018); Kumar et al. (2019) since these algorithms still rely to some extent on the ability to collect new experience under the current policy. This is required to generalize beyond the data for instance, to correct for value estimation errors for previously unobserved state-action pairs. One way to mitigate this problem is to (approximately) restrict the state-action distribution to the data contained in the batch. This can be achieved by regularizing against a *behavior prior* for instance as described e.g. by Siegel et al. (2020); Wu et al. (2019); Jaques et al. (2019); Laroche et al. (2017); Wang et al. (2020); Peng et al. (2020).

7.2 Information bottleneck

One view of the prior touched upon in Section 3 is that it encodes a 'default' behavior that can be independent of some aspect of the state such as the task identity. An alternate view is that the KL-regularized objective from Equation (11) implements an (approximate) in-

formation bottleneck where different forms of the prior and policy penalize the information flow between the agent's interaction history and future actions in different ways. This intuition that agents should exhibit similar behaviors in different contexts that only need to be modified or adjusted slightly e.g. as new information becomes available, and more generally the notion that information processing in the agent is costly and subject to constraints has a significant history in the neurosciences and cognitive sciences (e.g. Simon, 1956; Kool and Botvinick, 2018).

Formally, this idea can be expressed using an objective similar to the following:

$$\mathcal{L}_{\mathcal{I}} = \mathbb{E}_{\pi}[\sum_{t} \gamma^t r(s_t, a_t) - \alpha\gamma^t \mathrm{I}[x_t^G; a_t | x_t^D]], \tag{20}$$

where, at time t, x_t^G and x_t^D are the goal-directed and goal-agnostic information subsets; a_t is the action; α is a weighting factor and I is the conditional mutual information between x_t^G and a_t given x_t^D. This objective can be understood as attributing a cost for processing information contained in x_t^G to choose action a_t, or, as expressing a (soft-)constraint on the capacity of the channel between x_t^G and a_t. As we show in Appendix D.6, this term can be upper bounded by:

$$\mathcal{L}_{\mathcal{I}} \geq \mathbb{E}_{\pi}[\sum_{t} \gamma^t r(s_t, a_t) - \alpha\gamma^t \mathrm{KL}[\pi(a_t|x_t)||\pi_0(a_t|x_t)]],$$

where the RHS is exactly the KL-regularized objective from Equation (11). Therefore, we can see our work as a particular implementation of the information bottleneck principle, where we penalize the dependence of the action on the information that is hidden from the default policy π_0. The models with latent variables presented in Section 5 can be motivated in a similar way. Interestingly a similar information theoretic view of the ELBO has recently also found its way into the probabilistic modeling literature (e.g. Alemi et al., 2016, 2017).

Such constraints on the channel capacity between past states and future actions have been studied in a number of works (e.g. Tishby and Polani, 2011; Ortega and Braun, 2011; Still and Precup, 2012; Rubin et al., 2012; Ortega and Braun, 2013; Tiomkin and Tishby, 2017; Goyal et al., 2019). This view also bears similarity to the formulation of Strouse et al. (2018) where the goal is to learn to hide or reveal information for future use in multi-agent cooperation or competition.

7.3 Hierarchical RL

In this work, we focus on probabilistic models of behaviors and consider hierarchical variants as a means to increase model flexibility, and to introduce modularity that facilitates partial re-use of model components. Practically, the use of hierarchical models closely resembles, for instance, that of Heess et al. 2016; Hausman et al. 2018; Haarnoja et al. 2018a; Merel et al. 2019, in that a HL controller modulates a trained LL controller through a continuous channel and the trained LL can be subsequently reused on new tasks. Our analysis in Section 5 demonstrates that the use of low-level controllers can be motivated via the use of (hierarchical) behavior priors, thus providing a more general context for some ideas in the HRL literature from the perspective of modeling reusable behaviors.

Our work, however, also emphasizes a distinction between *model hierarchy* and *modeling hierarchically structured* (or otherwise complex) behavior. While hierarchical models have

been introduced as a way to introduce temporally consistent behaviors (e.g Dayan and Hinton, 1993; Parr and Russell, 1998; Sutton et al., 1999), as we have seen with the LSTM priors in Sections 4 and 6, model hierarchy is not a prerequisite, for instance, for modeling temporally correlated, structured behavior. Furthermore, even though the modularity of our hierarchical model formulation facilitates re-use, it is not a requirement as our investigation in Section 4 demonstrates.

The perspective of hierarchical priors has not been explored explicitly in the context of HRL. The discussion in Section 5 focuses on a specific formulation when the latent variables present in π_0 and π have the same dimensions; experimentally we explore a small set of particular models in Section 6. Yet, the general perspective admits a much broader range of models. Latent variables in π_0 and π may be continuous or discrete, and can be structured e.g. to model temporal or additional hierarchical dependencies, necessitating alternative exact or approximate inference schemes, thus giving rise to different models and algorithms. Below we illustrate how the KL regularized objective connects several strands of research. We present the main results here and leave the derivations to Appendix D.

Latent variables in π_0 Consider the case where only π_0 contains latent variables. An application of Jensen's inequality leads us to the following lower bound to $\mathcal{L}$

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_\pi[\textstyle\sum_t r(s_t, a_t)] - \mathrm{KL}[\pi(\tau)||\pi_0(\tau)] && (21)\\ &\geq \mathbb{E}_\pi\left[\textstyle\sum_t r(s_t, a_t) + \mathbb{E}_f\left[\log \pi_0(\tau|y)\right]\right.\\ &\qquad\qquad \left. - \mathrm{KL}[f(y|\tau)||\pi_0(y)]\right] + \mathrm{H}[\pi(\tau)]. && (22)\end{aligned}$$

Equation (22) is an application of the evidence lower bound to $\log \pi_0(\tau)$ and $f(y|\tau)$ is the approximate posterior, just like, for instance, in variational auto-encoders (VAE; Kingma and Welling 2013; Rezende et al. 2014). This emphasizes the fact that π_0 can be seen as a model of the trajectory data generated by π. Note however, as explained in Section 3 the model extends only to the history-conditional action distribution (the policy) and not the system dynamics. If y is discrete and takes on a small number of values exact inference may be possible. If it is conveniently factored over time, and discrete, similar to a hidden Markov model (HMM), then we can compute $f(y|\tau)$ exactly (e.g. using the forward-backward algorithm in (Rabiner, 1989) for HMMs). For temporally factored continuous y a learned parameterized approximation to the true posterior (e.g. Chung et al., 2015), or mixed inference schemes (e.g. Johnson et al., 2016) may be appropriate.

Variants of this formulation have been considered, for instance, in the imitation learning literature where the authors model trajectories generated by one or multiple experts using a parameteric decoder $\pi_0(\tau|y)$ and encoder $f(y|\tau)$. In this view π is fixed (it may correspond to one or more RL policies, or, for instance, a set of trajectories generated by a human operator) and we optimize the objective only with respect to π_0. Fox et al. (2017); Krishnan et al. (2017) learn a sequential discrete latent variable model of trajectories (an *option model*) with exact inference. Merel et al. (2019) and Wang et al. (2017b) model trajectories generated by one or multiple experts using a parameteric encoder $f(y|\tau)$ and decoder $\pi_0(\tau|y)$. The resulting models can then be used to solve new tasks by training a higher level controller to modulate the learnt decoder $\pi_0(\tau|y)$ similar to our transfer experiments in Section 6 (e.g. Merel et al., 2019).

Latent variables in π On the other hand, if only π contains latent variables z_t this results in the following approximation that can be derived from Equation (11) (see Appendix D.4 for proof):

$$\mathcal{L} \geq \mathbb{E}_\pi[\sum_t r(s_t, a_t)] + \mathbb{E}_\pi \left[\log \pi_0(\tau) + \log g(z|\tau)\right] + \mathrm{H}[\pi(\tau|Z)] + \mathrm{H}[\pi(Z)], \tag{23}$$

where g is a learned approximation to the true posterior $\pi(z|\tau)$. This formulation is also discussed in Hausman et al. (2018). It is important to note that the role of g here is only superficially similar to that of a variational distribution in the ELBO since the expectation is still taken with respect to π and *not* g. However, the optimal g, which makes the bound tight, is still given by the true posterior $\pi(z|\tau)$. The effect of g here is to exert pressure on $\pi(\tau|z)$ to make (future) trajectories that result from different zs to be 'distinguishable' under g.

Hausman et al. (2018) considers the special case when z is sampled once at the beginning of the episode. In this case, g effectively introduces an intrinsic reward that encourages z to be identifiable from the trajectory, and thus encourages a solution where different z lead to different outcomes. This formulation has similarities with diversity inducing regularization schemes based on mutual information (e.g. Gregor et al., 2017; Florensa et al., 2017; Eysenbach et al., 2019), but arises here as an approximation to trajectory entropy. More generally this is related to approaches based on empowerment (Klyubin et al., 2005; Salge et al., 2013; Mohamed and Rezende, 2015) where mutual information is used to construct an intrinsic reward that encourages the agent to visit highly entropic states.

This formulation also has interesting connections to auxiliary variable formulations in the approximate inference literature (Salimans et al., 2014; Agakov and Barber, 2004). A related approach is that of Haarnoja et al. (2018a), who consider a particular parametric form for policies with latent variables for which the entropy term can be computed analytically and no approximation is needed.

From an RL perspective, g can be thought of as a learned *internal reward* which is dependent on the choice of z i.e. we can think of it as a goal-conditioned internal reward where the internal goal is specified by z. As noted above, since the expectation in Equation (23) only considers samples from π, we are free to condition g on subsets of the information set, albeit at the expense of loosening the bound. We can use this as another way of injecting prior knowledge by choosing a particular form of g. For instance, it is easy to see that for $z, s \in \mathbb{R}^D$ and $g(z|s) = N(z|s, \sigma^2)$ we can effectively force z to play the role of a goal state, and g to compute an internal reward that is proportional to $||z - s||^2$. This view draws an interesting connection between the auxiliary variable variational formulations to other 'subgoal' based formulations in the RL literature (e.g. Dayan and Hinton, 1993; Vezhnevets et al., 2017; Nachum et al., 2018).

The options framework proposed by Sutton et al. (1999); Precup (2000) and more recently studied e.g. by Bacon et al. (2016); Fox et al. (2017); Frans et al. (2018) introduces discrete set of (sub-)policies ('options') that are selected by a (high-level) policy and then executed until an option-specific termination criterion is met (at which point a new option is chosen for execution). Options thus model recurrent temporal correlations between actions and can help improve exploration and credit assignment.

From a probabilistic perspective, options are a special case of our framework where a discrete latent variable or option c_t is held constant until some termination criterion is satisfied. The model consists of three components: a binary switching variable b_t which is drawn from a Bernoulli distribution $\beta(.|x_t, c_{t-1})$ that models option termination i.e. whether to continue with the current option or switch to a new one; a high level distribution over options $\pi(c_t|x_t)$ and an option specific policy $\pi_{c_t}(a_t|x_t)$. Together with these components, we have the following model:

$$\begin{aligned}\pi(a_t, z_t|x_t, z_{t-1}) &= \sum_{b\in\{0,1\}} \beta(b_t|x_t, c_{t-1}) \left[\pi(c_t|x_t)^{b_t} + \delta(c_t, c_{t-1})^{1-b_t}\right] \pi_{c_t}(a_t|x_t) \\ &= \beta(0|x_t, c_{t-1})\delta(c_t, c_{t-1})\pi_{c_t}(a_t|x_t) + \beta(1|x_t, c_{t-1})\pi(c_t|x_t)\pi_{c_t}(a_t|x_t),\end{aligned}$$

where $c_t \in \{1 \dots K\}$ indexes the current option and $z_t = (c_t, b_t)$. The presence of the time-correlated latent variable allows the model to capture temporal correlations that are not mediated by the state. A similar probabilistic model for options has also been considered by Daniel et al. (2016a,b) and more recently by Wulfmeier et al. (2020b) although they do not consider a notion of a prior in those cases.

One important question in the options literature and more generally in hierarchical reinforcement learning is how useful lower level policies or options can be learned (Barto and Mahadevan, 2002; Schmidhuber, 1991; Wiering and Schmidhuber, 1997; Dietterich, 1999; Boutilier et al., 1997). From a probabilistic modeling perspective options are beneficial when a domain exhibits temporal correlations between actions that cannot be expressed in terms of the option policies' observations. This can be achieved, for instance by introducing information asymmetry into the model, and a suitable prior can further shape the learned representation (and achieve e.g. effects similar to Harb et al. (2017) where a 'deliberation cost' for switching between options can be introduced through an information bottleneck as discussed in Section 5 in the form of a prior on the behavior of the option variable b_t).

Our results have also shown, however, that latent variables (and options in particular) are not necessary to model such temporal correlations. They can be similarly captured by unstructured auto-regressive models and then be used, for instance, for exploration (e.g. Sections 4.2 and 6.4). This observation is in keeping with the results of Riedmiller et al. (2018) who consider a setting where data is shared among a set of related tasks that vary in difficulty. The skills learnt by solving the easier tasks results in improved exploration on the hard tasks. They demonstrate that this exploration benefit is enough to solve challenging real world tasks even with non-hierarchical policies. In principle, this idea of reusing shared data across tasks has also been explored by Andrychowicz et al. (2018) in the context of goal-conditioned policies where the goal is relabelled in hindsight to include goals that were not sampled when acting. These ideas are orthogonal to exploiting structure in a set of related tasks that we consider in this work.. Similarly, (Nachum et al., 2019) empirically demonstrated that the advantages of sub-goal based (Nachum et al., 2018; Levy et al., 2017) and options based (Bacon et al., 2016; Precup, 2000) HRL approaches can largely be attributed to better exploration. Finally structured representations learnt by hierarchical option policies do hold additional computational (rather than representational) benefits, which could be useful for planning (e.g. Sutton et al., 1999; Precup, 2000; Krishnan et al., 2017). However this is not the focus on the present work.

8. Conclusion

We can summarize the findings of our work as follows:

- ***Behavioral Priors* can speed up learning**: We have demonstrated a method to learn *behavior priors* , which are policies that model trajectories that are shared amongst the solutions to many tasks. We have shown that using these priors during learning can lead to significant speedups on complex tasks. This improvement in performance can be attributed to the behaviors learnt by the priors. We have shown that *behavior priors* can model long term correlations in action space like movement patterns for complex bodies or complex task structure in the case of 'Sequential Navigation' tasks. While we have only considered one simple algorithmic application of this idea by extending the SVG-0 algorithm, the methods introduced here can theoretically be incorporated into any RL algorithm.

- **Information regularization can lead to generalization**: We have explored how restricting processing capacity through information asymmetry or modeling capacity through architectural constraints can be used as a form of regularization to learn general behaviors. We have demonstrated that this idea can be extended to structured models to train modular hierarchical polices. In contrast to other methods, in our approach, this behavior emerges during learning without explicitly building it into the task or architecture.

- **Structured Priors can be advantageous**: We have shown how models that incorporate structure in the form of latent variables can model more complex distributions; a feature that can be important in some domains.

- **General framework for HRL**: We discussed how our formulation can be related to a larger set of ideas within RL. The latent variable formulation presented in Section 5 can be connected to a number of ideas in HRL and more generally we have shown the relation between the KL-regularized RL objective and objectives based on mutual information and curiosity.

All of the methods that we have discussed in this work draw from a rich, mature literature on probabilistic modeling. Most of the ideas here largely stem from a simple principle: policies and task solutions define distributions over a space of trajectories. This perspective, which comes quite naturally from the KL-regularized objective, allows us to reason about priors which cover a broader manifold of trajectories. While we have considered a specific form of this objective where the prior is learnt jointly in a multi-task scenario; more generally the prior introduces a means to introduce inductive biases into the RL problem.

For instance in applications like robotics, real world interaction can often be expensive and we may only have access to some other form of data. In other cases where safety is a factor, exploration often needs to be limited to a manifold of 'safe' trajectories that may be learnt or hand defined in some way. As RL methods gain wider application to real world domains, these needs are bound to become increasingly important. The real world is often challenging to work with, impossible to perfectly simulate and quite often constrained in the kinds of solutions that can be applied. In such settings, the ability to introduce

prior knowledge about the problem in order to constrain the solution space or for better exploration is likely to be of importance. We hope that the ideas introduced here can be extended and improved upon to devise methods that feasibly apply RL to the real world.

Acknowledgments

We would like to acknowledge Gregory Wayne and Siddhant Jayakumar for immensely useful discussion. We would also like to thank Josh Merel and Tom Schaul for their valuable comments and guidance on earlier drafts of this work. We would also like to acknowledge David Szepesvari for their input on the role of hierarchy in RL. A large part of this work is the fruit of these discussions. Finally, we are indebted to the valuable contributions of Courtney Antrobus for her support, wisdom and organizational wizardry without which this work would have taken far longer to produce.

Appendix A. Probabilistic modeling and variational inference

In this section, we briefly describe core ideas from probabilistic machine learning and their connection with the KL-regularized objective as described in the text. We refer the reader to Bishop (2006); Koller and Friedman (2009); Murphy (2012) for introductory textbooks on probabilistic learning.

Probabilistic learning is framed in the context of learning a generative model p_θ which is a joint distribution of some data $x_{1:N} = \{x_1, \dots x_N\}$, parameterised by θ, potentially involving latent variables z, and including contextual information or covariates.

For simplicity, though the framework is more general, we will assume that the data is *independent and identically distributed (iid)* under the model, i.e.

$$p_\theta(x_{1:N}) = \prod_{i=1}^{N} p_\theta(x_i) = \prod_{i=1}^{N} \int p_\theta(x_i|z_i) p_\theta(z_i) dz_i, \tag{24}$$

where we have also introduced one iid latent variable z_i corresponding to each observed datum x_i. For intuition, each datum can be an image while the latent variable describes the unobserved properties of the visual scene. $p_\theta(z)$ is a prior distribution, e.g. over visual scenes, and $p_\theta(x|z)$ is an observation model, e.g. of the image given the scene.

In this framework, *inference* refers to computation or approximation of the posterior distribution over latent variables given observed data given by Bayes' rule, i.e.,

$$p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)} = \frac{p_\theta(x|z)p_\theta(z)}{\int p_\theta(x|z)p_\theta(z)dz}, \tag{25}$$

while *learning* refers to estimation of the model parameters θ from data. Typically this is achieved by maximising the log *likelihood*, i.e. the log marginal probability of the data as a function of the parameters θ:

$$\theta^{\text{MLE}} = \arg\max_\theta \log p_\theta(x_{1:N}) = \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(x_i). \tag{26}$$

In this paper we will only be interested in maximum likelihood learning. It is possible to extend the framework to one of Bayesian learning, i.e. to treat the parameters themselves as random variables, whose posterior distribution $p(\theta|x)$ captures the knowledge gained from observing data $x_{1:N}$.

The core difficulty of the probabilistic learning framework is in the computation of the log marginal and posterior probabilities (as well as their gradients). These are intractable to compute exactly for most models of interest.

A common approach is to employ a variational approximation. This stems from the following lower bound on the log marginal probability obtained by applying Jensen's in-

equality. For any conditional distribution $q(z|x)$ and any x, we have:

$$\log p_\theta(x) = \log \int p_\theta(x,z)dz = \log \int q(z|x)\frac{p_\theta(x,z)}{q(z|x)}dz, \tag{27}$$

$$= \log \mathbb{E}_q\left[\frac{p_\theta(x,z)}{q(z|x)}\right] \geq \mathbb{E}_q\left[\log \frac{p_\theta(x,z)}{q(z|x)}\right], \tag{28}$$

$$= \mathbb{E}_q\left[\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z|x)}\right] = \mathbb{E}_q\left[\log p_\theta(x|z)\right] - \mathrm{KL}[q(z|x)||p_\theta(z)], \tag{29}$$

$$= \mathbb{E}_q\left[\log p_\theta(x) + \log \frac{p_\theta(z|x)}{q(z|x)}\right] = \log p_\theta(x) - \mathrm{KL}[q(z|x)||p_\theta(z|x)], \tag{30}$$

where we use Jensen's inequality for lower bound in (28). (29) and (30) show two equivalent ways of writing the resulting bound, the first of which is tractable and the second of which is intractable but gives a different way to see that the expression gives a lower bound on the log marginal probability, since the KL divergence is non-negative.

This bound is often referred to as evidence lower bound or *ELBO*. As the last line shows the bound is tight when q matches the true posterior $p_\theta(z|x)$. For most models the true posterior is as intractable as the log marginal probability[9] and q is thus chosen to belong to some restricted family parameterised by variational parameters ϕ.

Plugging this lower bound into (26), we get the following objective to be maximized with respect to model parameters θ and variational parameters ϕ:

$$\sum_{i=1}^{N} \log p_\theta(x_i) \geq \sum_{i=1}^{N} \mathbb{E}_{q_\phi(z_i|x_i)}\left[\log p_\theta(x_i|z_i) + \log p_\theta(z_i) - \log q_\phi(z_i|x_i)\right]. \tag{31}$$

Appendix B. 2-D Task Results

In this section we present additional results to help build an intuition for the joint optimization scheme of the policy and *behavior prior* from Equation (11). We consider a 2-D setting where each task consists of a continuous probability distribution function (pdf) and the reward for a given 2-D action corresponds to the log-likelihood under the distribution. For most of our tasks we use a multivariate Gaussian distribution for which the reward for each task is maximal at the mean. We also consider one special task whose underlying distribution is a mixture of two Gaussians.

We consider a multi-task setup consisting of 4 different tasks shown as colored contours in Figure 19. The distributions for each task are: (green) a gaussian with mean (-3., 0.), scale (0.5, 0.5) and correlation ρ 0.5; (blue) a gaussian with mean (0, 2.), scale (0.2, 0.2) and correlation ρ 0.8; (orange) a gaussian with mean (3., 3.), scale (0.5, 0.5) and correlation ρ 0.5; (red) a mixture of two equally likely gaussians with mean (0, 0.) and (-4, -1) and an equal of scale (0.2, 0.2) with correlation 0.8. Each plot shows task specific posteriors (colored dots) and *priors* (red dots) for different prior models. We consider three models which, in increasing order of expressivity are: isotropic Gaussian; Gaussian with full covariance and a mixture of Gaussians. All posteriors are multivariate Gaussians.

9. This is not surprising considering that $p_\theta(z|x) = p_\theta(x,z)/p_\theta(x)$, i.e. the normalization constant of the posterior is exactly the intractable marginal probability.

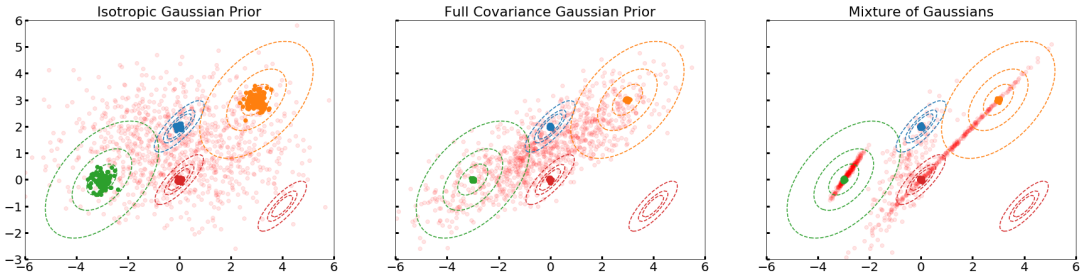


Figure 19: **Training on 2D task.** Each colored dotted contour represents a reward function for the different tasks. Each figure shows samples from the task specific posteriors (colored dots) and *behavior priors* (red) for different models.

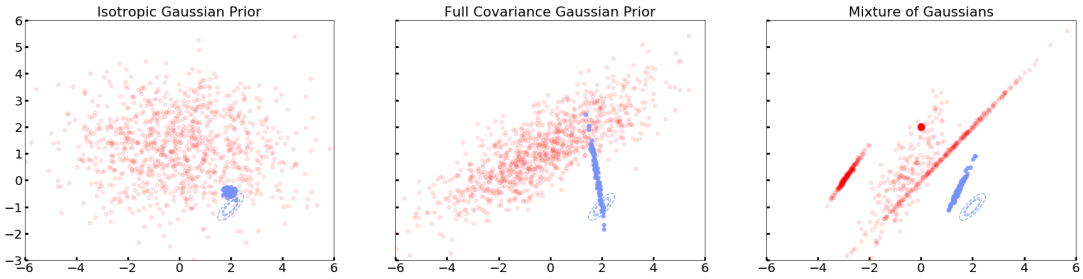


Figure 20: **Transfer to new 2D task.** Each dotted contour represents a reward function with each color representing a different tasks. Each figure plots samples from task specific posteriors (colored) and *behavior priors* (red) for various *prior* models. Posteriors were only allowed to train for 10 iterations.

In Figure 19 we observe that the task specific posteriors generate samples close to the center of the reward contours (where the reward is maximal). For the mixture task, the posteriors learn to model the solution closer to the distribution modeled under the *prior* which is exactly as expected from the KL in Equation (5). The *priors* capture a more general distribution containing *all* the task solutions. Moreover, the choice of *prior* affects the kinds of solutions learnt. The least expressive model - an isotropic Gaussian (Figure 19a), cannot capture the joint solution to all tasks perfectly. In contrast, a mixture of Gaussians *prior* (Figure 19c) almost perfectly models the mixture distribution of the posteriors. The joint optimization of Equation (6) has an interesting effect on the posterior solutions. More general *priors* lead to noisy posteriors as in Figure 19a) since the objective trades off task reward against minimizing the KL against the *prior*.

In Figure 20 we consider the effect of *priors* on a held-out transfer task. We consider a new task defined by a gaussian distribution centered at (2, -1) with a scale of (0.1, 0.1) and a correlation of 0.8. We consider a few-shot transfer setting where policies are trained for

only 10 iterations using the objective from Equation (5). This highlights the advantage of a more general *prior*.The more general trajectory space captured by the isotropic Gaussian is better suited for a transfer task as illustrated in Figure 20. In contrast the mixture of Gaussians *prior* from Figure 20c does not fare as well.

Appendix C. Algorithms

In this section we discuss our choice of baseline, the SVG-0 algorithm with entropy bonus (Heess et al., 2015; Riedmiller et al., 2018) and compare it against other state-of-the-art algorithms for continuous control including Soft Actor Critic (SAC) (Haarnoja et al., 2018b) and Deep Deterministic Policy Gradients (DPG/DDPG) (Silver et al., 2014; Lillicrap et al., 2015).

C.1 Theoretical Comparison

To begin, we reintroduce the KL-regularized RL objective from Section 3:

$$\mathcal{L} = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) - \alpha\gamma^t \mathrm{KL}[\pi(a_t|x_t)||\pi_0(a_t|x_t)]]. \tag{32}$$

As discussed previously, this objective trades off maximizing returns with staying close to some reference trajectory: π_0. If we consider π_0 to be a uniform distribution over actions (assuming a bounded action space), then we recover the entropy-regularized RL objective that has been considered by a number of previous works including SAC (Haarnoja et al., 2018b; Hausman et al., 2018; Riedmiller et al., 2018; Mnih et al., 2016; Williams and Peng, 1991), and is given by:

$$\mathcal{L} = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) + \alpha\gamma^t \mathrm{H}(\pi)]. \tag{33}$$

where $\mathrm{H}(\pi)$ is the entropy of the policy π. Equation (33) is thus a special case of the general KL-regularized objective which we focus on in this work and SAC is a particular choice of algorithm to optimize it.

The policy improvement step considered by SAC reparameterizes the policy gradient (exactly as proposed in SVG-0), resulting in a policy update that is given by:

$$L_\pi = \sum_{i=t}^{t+K-1} \mathop{\mathbb{E}}_{\substack{\eta\sim\rho \\ a=\pi(s_i,\eta)}} Q(s_i, a) + \alpha_{\mathrm{H}} \mathrm{H}_i(\pi), \tag{34}$$

where η is random noise drawn from a distribution ρ.

The gradient for this objective is presented in Haarnoja et al. (2018b) and is given by:

$$\nabla_\theta Ł_\pi = \sum_{i=t}^{t+K-1} \mathbb{E}_{\rho(\eta)} \left[\nabla_a Q(s_i, a) \nabla_\theta \pi_\theta(a|s_i, \eta)\right] - \nabla_\theta \alpha_{\mathrm{H}} \log \pi_\theta(a|s_i, \eta). \tag{35}$$

This gradient is in fact, exactly the one being optimized by our baseline from Section 3.5 (observe that the policy update from Equation (34) is identical to the policy update we

introduced for our baseline in Equation (14)). This is not surprising since the core ideas being exploited in both SAC and SVG-0 with entropy are identical: reparameterization to compute the gradient of a stochastic policy and an objective that trades off return against policy entropy.

While the two algorithms are thus strongly related, there are nevertheless algorithmic and implementation differences between them that we highlight below. Note that while each of these differences may provide benefits in some settings, when taken together there is no clear reason why one algorithm would perform better than the other in general. In fact, our empirical evaluation in Section C.2 indicate that both SVG-0 and SAC perform similarly on many control tasks.

Policy Evaluation On each step, SAC samples a single transition from a replay buffer and updates the critic according to:

$$\mathbb{E}_{(s_t,a_t,r_t,s_{t+1})}\left[(Q_\psi(s_t,a_t)-y(r,s_{t+1}))^2\right], \tag{36}$$

where the target y is given by:

$$y(r,s_{t+1}) = r+\gamma\left[\min_{j=1,2} Q_{\psi'}(s_{t+1},a_{t+1}) - \alpha\log\pi_\theta(a_{t+1}|s_{t+1},\eta)\right], \tag{37}$$

$$a_{t+1}\sim\pi_\theta(.|s_{t+1},\eta). \tag{38}$$

This version of SAC was described in Haarnoja et al. (2019) and makes use of two Q-functions (van Hasselt et al., 2015) in order to estimate the critic. Each critic is a 'soft' Q function that estimates not just the return but also the average trajectory entropy.

This update is different from the one we use for our baseline in Section 3.5. Our critic estimate uses the retrace algorithm (Munos et al., 2016) for importance weighted off-policy correction allowing us to sum rewards from multiple steps of the trajectory. In addition, we do not model a soft critic like SAC and instead only estimate the return of the policy. In practice, we found this makes little difference in performance.

Policy Improvement The version of SAC from Haarnoja et al. (2019) learns the temperature parameter that trades off the entropy bonus in the maximum entropy objective (similar to Abdolmaleki et al. (2018b)). Our baseline instead specifies a fixed tradeoff between the returns and entropy which we found worked quite well for the domains considered in this work.

Other details Finally SAC incorporates a number of ideas for learning that are not directly related to the algorithm per se but could nevertheless impact performance. For instance, for early exploration SAC optionally generates trajectories under a uniformly random policy over the action space (which for most domains is bounded). This allows the algorithm to potentially bootstrap learning early on with an exploration strategy that may provide advantages in some domains.

The parameteric form of the policy considered in SAC is a squashed Gaussian - a transformed distribution that applies a 'tanh' bijection to a Gaussian to ensure actions are in the range [-1, 1]. In contrast, our baseline agent uses a 'tanh' non-linearity on the mean of a Gaussian policy but does not bound the variance in any way.

In summary, while both SAC and our baseline are similar in terms of their core principles, there are nevertheless several important implementation differences between them. In Section C.2, we compare both algorithms on a standard open-sourced set of tasks and demonstrate that the performance of SVG-0 is not very different from SAC. Note that while this study may be of practical interest to RL practitioners, ultimately, these ideas are unrelated to *behavior priors* which are the primary focus of this work.

Comparison to DPG Alternatively in the formulation of Equation (14), if we were to instead choose a *deterministic* policy (and only learn the mean value μ_θ), while setting the entropy term to zero ($\alpha_{\mathrm{H}} = 0.$), then the gradient from Equation (35) would instead look like:

$$\nabla_\theta \mathbb{E}_{\pi_\theta}\left[Q(s,a)\right] = \left[\nabla_a Q(s,a) \nabla_\theta \mu_\theta(a|s)\right]. \tag{39}$$

This is similar to the formulation of DDPG that was considered by (Lillicrap et al., 2015) and originally introduced by (Silver et al., 2014) which is used extensively in RL for continuous control (Andrychowicz et al., 2018; Eysenbach et al., 2018; Henderson et al., 2019). Our baseline SVG-0 is thus also closely related to DPG and in fact has been shown to outperform DPG on a set of standard benchmark tasks (see Figure 12 in Abdolmaleki et al. (2018a)).

Finally it is important to note that the methods to learn *behavior priors* described in Sections 3 and 5 are orthogonal to the choice of underlying algorithm.

C.2 Empirical Comparison

In what follows we perform an empirical comparison between the baseline used in this work (SVG-0 with entropy) against SAC. We perform this comparison in two parts: first on a standard benchmark, the DeepMind Control Suite and then on some of the domains considered in this work.

Comparison on Control Suite Figure 21 compares the performance of our implementation SVG-0 with entropy against the SAC implementation from Yarats and Kostrikov (2020) on the DeepMind Control Suite (Tassa et al., 2018). For this comparison we use a single-actor replay setup similar to the one used in Haarnoja et al. (2018b) and Yarats and Kostrikov (2020). For SAC we directly plot the data from Yarats and Kostrikov (2020). Yarats and Kostrikov (2020) regularly evaluate the policy for 10 episodes every 10000 steps. In contrast, our implementation evaluates the policy for 10 episodes asynchronously every 10 seconds throughout training. The standard deviation of the plotted results is thus not directly comparable but the mean is a good indicator of performance.

As the figure shows, both SVG-0 and SAC perform similarly on most domains including high-dimensional tasks like Humanoid-walk. While there are small differences on some domains (notably 'acrobot', 'hopper' and 'swimmer'), neither algorithm is clearly superior and the performance is about the same on most tasks. The main takeaway from this set of results is to illustrate that SVG-0 with entropy is a strong baseline and is comparable in performance to other algorithms like DDPG and SAC (a similar discussion can be found in Abdolmaleki et al. (2018a); see Figure 12). Note that all results used a single set of hyperparameters averaged over 3 seeds as detailed in Appendix F.

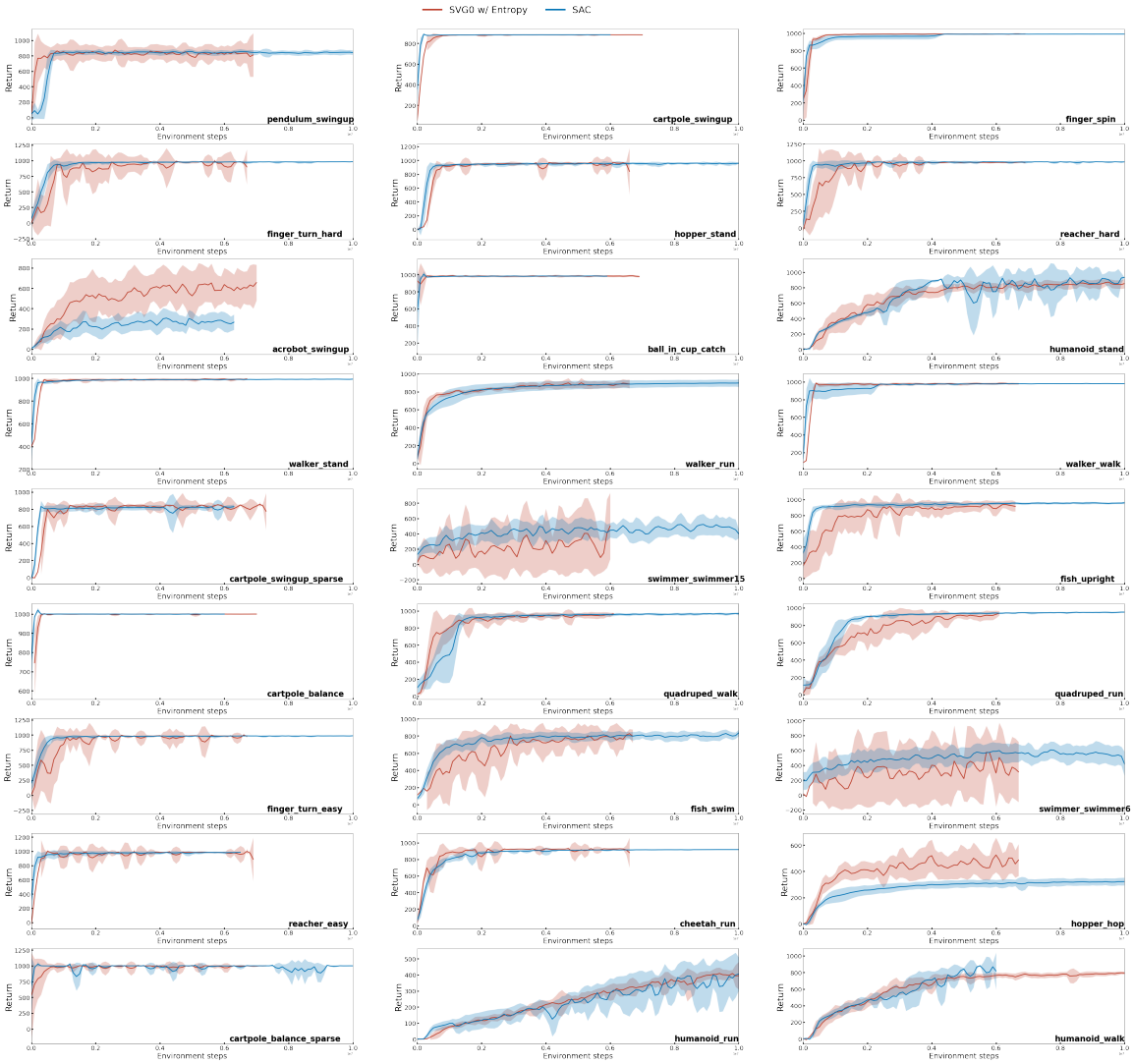


Figure 21: **Comparison:** Performance comparison between SVG-0 with entropy and Soft Actor Critic (SAC)

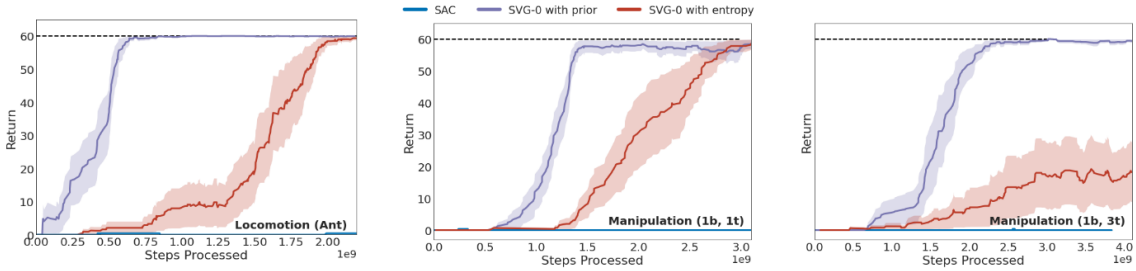


Figure 22: **SAC on locomotion and manipulation tasks:** Comparison of SAC on the *Locomotion(Ant)*, *Manipulation(1b, 1t)* and *Manipulation(1b, 3t)* tasks against SVG-0 with entropy and SVG-0 with priors.

Comparison on locomotion and manipulation task We further evaluated SAC on the *Locomotion(Ant), Manipulation(1 box, 1 target)* and *Manipulation(1 box, 3 targets)* tasks considered in Sections 4 and 6. In order to be able to compare algorithms on an equal footing we implemented our own version of SAC and confirmed that our algorithmic implementation matches state-of-the-art performance using the architecture and setup described in Haarnoja et al. (2019).

Keeping this network configuration fixed, but using the distributed training setup described in Section 4, we then evaluated the performance of SAC on a number of locomotion and manipulation tasks. As Figure 22 shows, SAC was unable to learn on any of these extremely sparse reward task with the same network configuration that works on OpenAI gym.

Note that we did not perform an exhaustive search of network and algorithmic parameters on this task and only swept over learning rates and the batch size [10]. This is consistent with reports that SAC does not work well on extremely sparse reward tasks like the one considered here (personal correspondence: Tuomas Haarnoja).

However this study does illustrate that deep Reinforcement Learning agents can be sensitive to a number of hyper-parameter choices that may vary across domains. For this reason, as far as possible in all the experiments considered in Sections 4 and 6 we have tried to ensure various factors such as architectural parameters, training speed and network complexity are consistent across the plots being compared.

In summary, we have shown that our baseline performs on-par with a public implementation of SAC. We have tried to match algorithms as much as possible but found SAC performed poorly on the subset of tasks considered in our work.

Appendix D. Derivations

Here, we present derivations for the lower bounds and proofs presented in the main text.

10. We also tried to use networks similar to those used by our baseline in Section 4 on the 'Locomotion (Ant)' task, but did not manage to get SAC to work.

D.1 Decomposition of KL per-timestep

In this part we demonstrate that the KL over trajectory distributions decomposes conveniently to a sum over per timestep KLs. We start with the general KL formulation over trajectories:

$$\mathcal{L} = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)] - \mathrm{KL}[\pi(\tau)||\pi_0(\tau)].$$

If we choose π_0 and π to be distributions over trajectories that arise as a composition of the true system dynamics $p(s_{t+1}|s_t, a_t)$ and state-conditional distributions over actions ($\pi_0(a_t|s_t)$ and $\pi(a_t|s_t)$) we obtain the following factorization:

$$\pi_0(\tau) = p(s_0) \prod_t \pi_0(a_t|s_t) p(s_{t+1}|s_t, a_t), \tag{40}$$

$$\pi(\tau) = p(s_0) \prod_t \pi(a_t|s_t) p(s_{t+1}|s_t, a_t), \tag{41}$$

in which case we recover exactly the expected reward objective in Section 3, with a per-step regularization term that penalizes deviation between π_0 and π:

$$\begin{aligned}\mathcal{L}(\pi_0, \pi) &= \mathbb{E}_{\pi}[\sum_t (\gamma^t r(s_t, a_t) + \gamma^t \log \frac{\pi_0(a_t|s_t)}{\pi(a_t|s_t)})] \\ &= \mathbb{E}_{\pi}[\sum_t \gamma^t (r(s_t, a_t) - \mathrm{KL}[\pi||\pi_0])].\end{aligned}$$

D.2 Optimal prior in multi-task DISTRAL

In this part we derive the form for the optimal prior π_0 that minimizes the multi-task objective from Section 3.2 (Teh et al., 2017). Recall that in this setting, we are given a distribution over tasks $p(w)$ where tasks are indexed by $w \in \mathcal{W}$.

We reproduce the KL-regularized objective with task-specific policies π_w and 'shared' prior π_0 below:

$$\mathcal{L} = \sum_w p(w) \mathbb{E}_{\pi_w} \left[\sum_t \gamma^t r_w(s_t, a_t) - \gamma^t \mathrm{KL}[\pi_w(a_t|x_t)||\pi_0(a_t|x_t)] \right], \tag{42}$$

When given a set of task-specific policies, we would like to find the optimal prior that minimizes this loss; that is, we would like to find π_0^* such that:

$$\pi_0^* = \arg\min_{\pi_0} \sum_w p(w) \mathbb{E}_{\tau \sim \pi_w} \sum_t \gamma^t [\mathrm{KL}[\pi_w(\cdot|x_t)||\pi_0(\cdot|x_t)]],$$

where trajectory sequences are generated under the task-specific state visitation distribution:

$$p(\tau|\pi_w) = p(s_0) \prod_{t'=0}^{t-1} \pi_w(a_{t'}|s_{t'}) p(s_{t'+1}|s_{t'}, a_t).$$

Since we are interested in optimizing for the prior, the term we are interested in is:

$$\begin{aligned}\pi_0^* &= \arg\min_{\pi_0} \sum_w p(w)\mathbb{E}_{\tau\sim\pi_w} \sum_t \gamma^t \mathbb{E}_{a_t\sim\pi_w(.|x_t)} - \log[\pi_0(a_t|x_t)], \\ &= \arg\max_{\pi_0} \sum_w p(w)\mathbb{E}_{\tau\sim\pi_w} \sum_t \gamma^t \mathbb{E}_{a_t\sim\pi_w(.|x_t)} \log[\pi_0(a_t|x_t)].\end{aligned}$$

This corresponds to first picking a task $w \sim p(w)$ and then generating trajectories under the corresponding policy π_w for which we would like to maximize the (log-) likelihood (MLE) of actions under the prior π_0.

Instead, we could flip the order of the expectations and first consider the marginal distribution over trajectories given by:

$$p(\tau) = \sum_w p(w)p(\tau|\pi_w), \tag{43}$$

to optimize:

$$\pi_0^* = \arg\max_{\pi_0} \mathbb{E}_{\tau\sim p(\tau)} \sum_t \sum_w p(w|x_t)\gamma^t \mathbb{E}_{a_t\sim\pi_w(.|x_t)} \log[\pi_0(a_t|x_t)], \tag{44}$$

where $p(w|x_t)$ is the posterior distribution over tasks given the trajectory snippet x_t. Importantly, we observe that this posterior is independent of the task-specific policy π_w given x_t.

It follows that the optimal prior that maximizes the objective from Equation (44) for a given state-history x_t is given by:

$$\pi_0^*(a_t|x_t) = \sum_w p(w|x_t)\pi_w(a_t|x_t).$$

D.3 Bound when π_0 contains latent

In this part, we derive the bound presented in Equation (22). In this setting we consider the case where π_0 contains latent variables y.

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_\pi[\textstyle\sum_t \gamma^t (r(s_t, a_t)] - \gamma^t \mathrm{KL}[\pi(\tau)||\pi_0(\tau)] \\ &\geq \mathbb{E}_\pi \left[\textstyle\sum_t \gamma^t \left(r(s_t, a_t) + \mathbb{E}_f \left[\log \frac{\pi_0(\tau, y)}{f(y|\tau)}\right]\right)\right] + \gamma^t \mathrm{H}[\pi(\tau)] \\ &= \mathbb{E}_\pi \left[\textstyle\sum_t \gamma^t (r(s_t, a_t) + \mathbb{E}_f \left[\log \pi_0(\tau|y)\right]\right) \\ &\qquad\qquad - \gamma^t \mathrm{KL}[f(y|\tau)||\pi_0(y)]\right] + \gamma^t \mathrm{H}[\pi(\tau)].\end{aligned} \tag{45}$$

D.4 Bound when π contains latent

In what follows we expand on the derivation for the bound introduced in Equation (23). We begin with the objective for $\mathcal{L}$ from Equation (5) and consider the case where only π contains latent variables z_t which could be continuous. This results in the approximation

as discussed in Hausman et al. (2018):

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_\pi[\sum_t \gamma^t r(x_t, a_t)] - \gamma^t \mathbb{E}_\pi[\mathrm{KL}[\pi(\tau)||\pi_0(\tau)]] \\ &\geq \mathbb{E}_\pi[\sum_t \gamma^t r(x_t, a_t)] + \gamma^t \mathbb{E}_\pi\left[\log \frac{\pi_0(\tau) g(z|\tau)}{\pi(\tau|Z)\pi(Z)}\right] \\ &= \mathbb{E}_\pi[\sum_t \gamma^t r(x_t, a_t)] + \gamma^t \mathbb{E}_\pi\left[\log \pi_0(\tau) + \log g(z|\tau)\right] + \gamma^t \mathrm{H}[\pi(\tau|Z)] + \gamma^t \mathrm{H}[\pi(Z)]. \end{aligned} \tag{46}$$

D.5 Bound when both π_0 and π contain latents

Here we present the proof for the main result presented in Section 5.

$$\begin{aligned}\mathrm{KL}[\pi(a_t|x_t)||\pi_0(a_t|x_t)] \leq& \mathrm{KL}[\pi(a_t|x_t)||\pi_0(a_t|x_t)] + \mathbb{E}_{\pi(a_t|x_t)}[\mathrm{KL}[\pi^H(z_t|x_t)||\pi_0^H(z_t|x_t)]] \\ &= \mathbb{E}_{\pi(a_t|x_t)}[\log \tfrac{\pi(a_t|x_t)}{\pi_0(a_t|x_t)}] \\ &\qquad + \mathbb{E}_{\pi(a_t|x_t)}[\mathbb{E}_{\pi^H(z_t|a_t,x_t)}[\log \tfrac{\pi^H(z_t|a_t,x_t)}{\pi_0^H(z_t|a_t,x_t)}] \\ &= \mathbb{E}_{\pi(a_t,z_t|x_t)}[\log \tfrac{\pi(a_t,z_t|x_t)}{\pi_0(a_t,z_t|x_t)}] = \mathrm{KL}(z_t, a_t|x_t) \\ &= \mathbb{E}_{\pi^H(z_t|x_t)}[\log \tfrac{\pi^H(z_t|x_t)}{\pi_0^H(z_t|x_t)}] \\ &\qquad + \mathbb{E}_{\pi^H(z_t|x_t)}[\mathbb{E}_{\pi^L(a_t|z_t,x_t)}[\log \tfrac{\pi^L(a_t|z_t,x_t)}{\pi_0^L(a_t|z_t,x_t)}] \\ &= \mathrm{KL}[\pi^H(z_t|x_t)||\pi_0^H(z_t|x_t)] \\ &\qquad + \mathbb{E}_{\pi^H(z_t|x_t)}[\mathrm{KL}[\pi^L(a_t|z_t,x_t)||\pi_0^L(a_t|z_t,x_t)]]. \end{aligned} \tag{47}$$

D.6 Upper Bounding Mutual Information

We begin with the objective for mutual information that was introduced in Section 7:

$$\mathcal{L}_\mathcal{I} = \mathbb{E}_\pi[\sum_t \gamma^t r(s_t, a_t) - \alpha\gamma^t \mathrm{I}[x_t^G; a_t|x_t^D]].$$

The mutual information term can be upper bounded as follows:

$$\begin{aligned}\mathrm{I}[x_t^G; a_t|x_t^D] &= \int \pi(x_t^G, a_t|x_t^D) \log \frac{\pi(x_t^G, a_t|x_t^D)}{\pi(x_t^G|x_t^D)\pi(a_t|x_t^D)} \\ &= \int \pi(x_t^G|x_t^D)\pi(a_t|x_t^G, x_t^D) \log \frac{\pi(x_t^G|x_t^D)\pi(a_t|x_t^G, x_t^D)}{\pi(x_t^G|x_t^D)\pi(a_t|x_t^D)} \\ &= \int \pi(x_t^G|x_t^D)\pi(a_t|x_t) \log \frac{\pi(a_t|x_t)}{\pi(a_t|x_t^D)} \\ &\leq \int \pi(x_t^G|x_t^D)\pi(a_t|x_t) \log \frac{\pi(a_t|x_t)}{\pi_0(a_t|x_t^D)} \\ &= \mathbb{E}_\pi[\mathrm{KL}[\pi(a_t|x_t)||\pi_0(a_t|x_t^D)]], \end{aligned} \tag{48}$$

since

$$\begin{aligned}\mathrm{KL}[\pi(a_t|x_t)||\pi_0(a_t|x_t^D)] &\geq 0 \\ \mathbb{E}_\pi[\log \frac{\pi(a_t|x_t)}{\pi_0(a_t|x_t^D)}] &\geq 0 \\ \mathbb{E}_\pi[\log \pi(a_t|x_t)] &\geq \mathbb{E}_\pi[\log \pi_0(a_t|x_t^D)].\end{aligned}$$

We can extend the bound in Equation (48) to the hierarchical setting by combining it with the bound from Equation (47). In the case when the LL is shared between π and π_0 (and thus the lower KL is exactly 0), this reduces to:

$$\mathrm{I}[x_t^G; a_t|x_t^D] \leq \mathrm{KL}[\pi^H(z_t|x_t)||\pi_0^H(z_t|x_t^D)].$$

Appendix E. Environments

In this section, we describe the detailed configuration of the continuous control tasks and bodies used. Videos depicting these tasks can be found at the accompanying website: `https://sites.google.com/view/behavior-priors`. The code to run these tasks can be found at: `https://github.com/deepmind/deepmind-research/tree/master/box_arrangement`.

E.1 Tasks

Toy Domain: PointMass on Grid The 1×1 arena consists of the agent at the center and 4 targets placed uniformly at random on the square. On each episode a sequence of targets is generated according to 2nd order Markovian dynamics based on the previous 2 targets visited based on the distribution given in Table 3. Note that the dynamics are chosen in such a way such that the marginal probabilities are uniformly distributed when conditioned on a single target. We consider two versions of the task: *easy task* where the agent is given the next target to visit and the sequence continues until the episode terminates (after 375 steps). For the *hard task*, the agent must visit the 4 targets in order. The agent receives a reward of +10 for every correct target and an additional bonus of +50 for completing the *hard task*. For the *easy task* we generate sequences with $p = 0.9$ while for the hard task we generate sequences with $p = 1.0$ (refer to Table 3). The policy and *priors* receive the location of all the targets and the agent's global position and velocity. Additionally, for the *easy task*, the policy gets the next target in sequence while for the *hard task* it receives the 6-length sequence as well as the index of the next target.

Locomotion (Humanoid) We use the open-source version of the go_to_target task from the DeepMind control suite (Tassa et al., 2018). We use an arena of 8×8 with a moving target and a humanoid walker. The humanoid is spawned in the center of the arena and a target is spawned uniformly at random on the square on each episode. The agent receives a reward of 1 when is within a distance of 1 unit from the target for up to 10 steps. At the end of this period, a new target is spawned uniformly at random in a 1.5×1.5 area around the walker. New targets continue to be spawned in this manner for the duration of the episode (400 steps). The agent receives proprioceptive information and the egocentric location of the target relative to its root position.

Sequence	Target 0	Target 1	Target 2	Target 3
10	0.00	0.00	p	1-p
20	0.00	1-p	0.00	p
30	0.00	p	1-p	0.00
01	0.00	0.00	p	1-p
21	1-p	0.00	0.00	p
31	p	0.00	1-p	0.00
12:	1-p	0.00	0.00	p
32:	p	1-p	0.00	0.00
02:	0.00	p	0.00	1-p
03:	0.00	p	1-p	0.00
13:	1-p	0.00	p	0.00
23:	p	1-p	0.00	0.00

Table 3: **Markov Transition Dynamics** used to generate the target sequence for the point mass task. p indicates the probability of visiting that target and the row indices indicate the previous two targets visited.

Locomotion (Ant) On a fixed 8×8 arena, an ant walker and 3 targets are placed at a random location and orientation at the beginning of each episode. In each episode, one of the 3 targets is chosen uniformly at random to be the *goal*. The agent receives a reward of +60 if its root is within a 0.5×0.5 area around the target. The episode terminates when the agent reaches the target or after 400 timesteps. The agent receives proprioceptive information, the location of all the targets relative to its current location and a onehot encoding of the *goal* target index as input.

Locomotion and Manipulation On a fixed 3×3 arena, an ant walker, 2 targets and a cubic box of edge 0.5 are placed at a random location and orientation at the beginning of each episode. In each episode, one of the 2 targets is chosen uniformly at random to be the *agent goal* and the other becomes the *box goal*. There are 2 components to this task: the agent receives a reward of +10 if its root is within a 0.5×0.5 area around the *agent goal* and a +10 if the center of the box is within *box goal*. If the agent completes *both* components of the tasks, it receives a bonus reward of +50. The episode terminates when the agent and box are at their goals or after 400 timesteps. The agent receives proprioceptive information, the location of all the targets relative to its current location, the location of the box relative to its current location and a onehot encoding of the *agent goal*.

Manipulation (1 box, k targets) On a fixed 3×3 arena, an ant walker, k targets and a cubic box of edge 0.5 are placed at a random location and orientation at the beginning of each episode. In each episode, one of the k targets is chosen uniformly at random to be the *box goal*. The agent receives a reward of +60 if the center of the box is within *box goal*. The episode terminates when box is at the goal or after 400 timesteps. The agent receives proprioceptive information, the location of all the targets relative to its current location,

the location of the box relative to its current location and a onehot encoding of the *box goal*. In our experiments we have two tasks where $k = 1$ and $k = 3$.

Manipulation (2 box, k targets) On a fixed 3×3 arena, a ball walker, k targets and a cubic box of edge 0.5 are placed at a random location and orientation at the beginning of each episode. In each episode, one of the k targets is chosen uniformly at random to be the *first box goal* and another is chosen to be the *second box goal* . The agent receives a reward of +10 if the center of the first box is within the *first box goal* or the center of the second box is within the *second box goal*. If both boxes are at their goals, the agent gets a bonus reward of +50. The episode terminates when both boxes are at their goals or after 400 timesteps. The agent receives proprioceptive information, the location of all the targets relative to its current location, the location of the boxes relative to its current location and a onehot encoding of the *first box goal*. In our experiments we have $k = 2$.

Manipulation (2 box gather) This is a variation of the Manipulation (2 box, 2 targets) as described above where the agent receives a +60 for bring both boxes together such that their edges touch. The episode terminates when both boxes touch or after 400 timesteps.

E.2 Bodies

We use three different bodies: Ball, Ant, and Humanoid as defined by the DeepMind control suite (Tassa et al., 2018). These have been all been used in several previous works (Heess et al., 2017; Xie et al., 2018; Galashov et al., 2019) The **Ball** is a body with 2 actuators for moving forward or backward, turning left, and turning right. The **Ant** is a body with 4 legs and 8 actuators, which moves its legs to walk and to interact with objects. The **Humanoid** is a body with a torso and 2 legs and arms and 23 actuators. The *proprioceptive* information provided by each body includes the body height, position of the end effectors, the positions and velocities of its joints and sensor readings from an accelerometer, gyroscope and velocimeter attached to its torso.

Appendix F. Experiment details

Throughout the experiments, we use 32 actors to collect trajectories and a single learner to optimize the model. For the transfer experiments in Section 4.2 and Section 6.4 however we use 4 actors per learner. We plot average episode return with respect to the number of steps processed by the learner. Note that the number of steps is different from the number of agent's interaction with environment. By steps, we refer to the batches of experience data that are samples by a centralized learner to update model parameters. When learning from scratch we report results as the mean and standard deviation of average returns for 5 random seeds. For the transfer learning experiments, we use 5 seeds for the initial training, and then two random seeds per model on the transfer task. Thus, in total, 10 different runs are used to estimate mean and standard deviations of the average returns. Hyperparameters, including KL cost and action entropy regularization cost, are optimized on a per-task basis. More details are provided below.

F.1 Hyperparameters used for experiments

For most of our experiments we used MLP or LSTM torso networks with an final MLP layer whose output dimensionality is twice the action space for the policy and *prior*. The output of this network is used as the mean and the log scale to parameterize an isotropic Gaussian output. For stability and in keeping with prior work (e.g. Heess et al., 2015), we pass the log scale output through a softplus layer and add an additional fixed noise of 0.01 to prevent collapse. The hierarchical policies described in Section 6 use a similar approach where the output of the higher level layer is used to paramterize a Gaussian from which we sample the latent z. We also use target networks for actor, *prior* and critic for increased stability.

Below we provide the default hyperparameters used across tasks followed by modifications made for specific tasks as well as the range of parameters swept over for each task. All MLP networks used *ELU* activations. The output of the Gaussian actions was capped at 1.0.

F.1.1 Default parameters

Actor learning rate: β_{pi} = 1e-4. *Critic learning rate:* β_{pi} = 1e-4. *Prior learning rate:* β_{pi} = 1e-4.

Target network update period: = 100 *HL policy Torso:* MLP with sizes (200, 10) *LL policy Torso:* MLP with sizes (200, 100) *Critic Torso:* MLP with sizes (400, 300).

Batch size: 512 *Unroll length:* 10

Entropy bonus: λ = 1e-4. *Distillation cost:* α = 1e-3. *Posterior entropy cost:* α = 1e-3. *Number of actors:* 32

F.1.2 Task specific parameters

For each task we list any parameters that were changed from the default ones and specify sweeps within square brackets ([]). The best hyperparameters for each configuration (which were used for the results plotted in the paper) are highlighted in **bold** font. When the best hyperparameters differ across different types of information asymmetry experiments, we explicitly make a note of what worked best in each setting.

PointMass: Easy task *MLP/LSTM Actor torso:* (64, 64) *MLP/LSTM Prior torso:* (64, 64) *Critic torso:* LSTM with size (128, 64)

Entropy bonus: [1e-4, **0.0**] *Distillation cost:* [0.0, 1e-1, **1e-2**, 1e-3] *Unroll length:* 40

Actor/Critic/Prior learning rate: [5e-5, 1e-4, **5e-4**]

Hier. MLP/Hier. LSTM Prior HL torso: [(64, 4), (64, 10), **(64, 100)**] *Hier. MLP/Hier. LSTM Posterior HL torso:* [(64, 4), (64, 10), **(64, 100)**] *Hier. MLP/Hier. LSTM LL torso:* [(64, 64)]

PointMass: Hard task *MLP/LSTM Actor torso:* (64, 64) *MLP/LSTM Prior torso:* (64, 64) *Critic torso:* LSTM with size (128, 64)

Entropy bonus: [**1e-4**, 0.0] *Distillation cost:* [0.0, **1e-1**, 1e-2, 1e-3] *Unroll length:* 40

Actor/Critic/Prior learning rate: [5e-5, 1e-4, **5e-4**]

Hier. MLP Prior HL torso: [(64, 4), **(64, 10)**, (64, 100)] *Hier. MLP Posterior HL torso:* [(64, 4), **(64, 10)**, (64, 100)] *Hier. LSTM Prior HL torso:* [(64, 4), (64, 10), **(64,**

100)] *Hier. LSTM Posterior HL torso:* [(64, 4), (64, 10), **(64, 100)**] *Hier. MLP/Hier. LSTM LL torso:* [(64, 64)] *Number of actors:* 4

Locomotion (Humanoid) *MLP Actor torso:* (200, 100) *MLP Prior torso:* (200, 100) *Distillation cost:* [1e-1, 1e-2, 1e-3, 1e-4] *Best distillation cost (proprio):* (**0.1**) *Best distillation cost (full):* (**0.01**)

Actor learning rate: [5e-5, 1e-4, **5e-4**] *Critic learning rate:* [5e-5, 1e-4, **5e-4**] *Prior learning rate:* [5e-5, 1e-4, **5e-4**]

Target network update period: [50, 100] *Best target network update period (baseline):* (**100**) *Best target network update period (proprio):* (**50**) *Best target network update period (full):* (**100**)

Locomotion (Ant) *MLP Actor torso:* (200, 100) *MLP Prior torso:* (200, 100) *HL Policy torso:* [(200, 4), (200, 10), **(200, 100)**] *Comparable MLP Prior torso (Section 6):* [**(200, 4, 200, 100)**, (200, 10, 200, 100), (200, 100, 200, 100)]

Distillation cost: [1e-1, **1e-2**, 1e-3, 1e-4]

Actor/Critic/Prior learning rate: [5e-5, **1e-4**, 5e-4] *Best learning rate for unstructured prior with full info (Section 4):* **5e-4** *Best learning rate for prior of comparable size with Proprio (Section 6): Best learning rate for Gaussian prior (Section 6):* **5e-4**

Parameter for AR-1 process: [**0.9**, 0.95]

Entropy bonus: [**1e-4**, 0.0] *Best entropy bonus for unstructured prior with proprioception+target and full IA:* **0.0** *Best entropy bonus for unstructured prior with comparable nets and proprioception (Section 6):* **0.0**

Locomotion and Manipulation *MLP Actor torso:* (200, 100) *Encoder for proprioception:* (200, 100) *Encoder for target:* (50) *Encoder for box:* (50) *Encoder for task encoding:* (50)

Actor/Critic/Prior learning rate: [5e-5, **1e-4**, 5e-4] *Best learning rate for MLP Proprio prior:* **5e-5**

HL Policy torso: [(200, 4), **(200, 10)**, (200)] *Best torso size for proprio prior:* **(200, 4)**

Distillation cost: [1e-1, **1e-2**, 1e-3] *Best distillation cost for Hier. Proprio+Box prior:* **1e-3** *Best distillation cost for MLP baseline:* **0.1**

Entropy bonus: [**1e-4**, 0.0] *Best entropy bonus for baseline:* [**0.0**]

Manipulation (1 box, 1 target) *HL Policy torso:* [(200, 4), (200, 10), **(200, 100)**] *Best HL Policy torso for baseline:* [**(200, 10)**] *Best HL Policy torso for AR-1 prior:* [**(200, 10)**]

LL Policy torso: [(200, 4), (200, 10), (200, 100)] *Policy/MLP Prior Torso:* [(200, 4, 200, 100), (200, 10, 200, 100), **(200, 100, 200, 100)**]

Parameter for AR-1 process: [**0.9**, 0.95]

Distillation cost: [1e-2, **1e-3**, 1e-4] *Actor/Critic/Prior learning rate:* [1e-4, **5e-4**] *Best learning rate for baseline:* [**1e-4**]

Entropy bonus: [**1e-3**, 1e-4, 0.0] *Best entropy bonus for Gaussian and AR-1 prior:* [**0.0**]

Manipulation (1 box, 3 targets) *Distillation cost:* [**1e-1**, 1e-2, 1e-3] *Best distillation cost for proprio+target:* [**1e-2**]

Actor/Critic/Prior learning rate: [1e-4, **5e-4**] *Best learning rate for baseline:* [**1e-4**]
Target update period: [**50**, 100] *Best target update period for proprio and full:* [**100**]
Entropy bonus: [**1e-3**, 1e-4] *Best entropy bonus for baseline and full:* [**1e-4**]

Manipulation (2 box, 2 targets) *Distillation cost:* [1e-2, 1e-3, **1e-4**] *Best distillation cost for AR1 prior:* [**1e-2**]

HL Policy torso: [(200, 4), (200, 10), **(200, 100)**] *Policy/MLP Prior Torso:* [(200, 4, 200, 100), (200, 10, 200, 100), **(200, 100, 200, 100)**]

Actor/Critic/Prior learning rate: [5e-5, **1e-4**, 5e-4] *Best learning rate for baseline and MLP prior:* **5e-5** *Parameter for AR-1 process:* [**0.9**, 0.95]

Manipulation (2 box gather) *Distillation cost:* [**1e-2**, 1e-3, 1e-1] *Best distillation cost for MLP prior:* [**1e-1**]

HL Policy torso: [(200, 4), (200, 10), **(200, 100)**] *Best HL Policy torso for AR-1:* [**(200, 4)**] *Policy/MLP Prior Torso:* [(200, 4, 200, 100), (200, 10, 200, 100), **(200, 100, 200, 100)**]

Actor/Critic/Prior learning rate: [1e-4, **5e-4**]
Parameter for AR-1 process: [**0.9**, 0.95]

F.1.3 Parameters for Control Suite tasks

Actor learning rate: 1e-4.
Critic learning rate: 1e-4.
Batch size: 32
Unroll length: 20
Entropy bonus: 1e-4
Target network update period: 250
Maximum replay size: 2e6
Actor network: MLP with LayerNorm of size (256) and tanh activation followed by MLP of size (256, 256) with elu. Actor head Gaussian uses minimum scale of 0.3 and tanh on mean output.
Critic network: MLP with LayerNorm of size (256) and tanh activation followed by MLP of size (256, 256) with elu. 'tanh' applied to action input for critic.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Jonas Degrave, Steven Bohez, Yuval Tassa, Dan Belov, Nicolas Heess, and Martin A. Riedmiller. Relative entropy regularized policy iteration. *CoRR*, abs/1812.02256, 2018a. URL `http://arxiv.org/abs/1812.02256`.

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018b.

Felix V. Agakov and David Barber. An auxiliary variational method. In *Neural Information Processing, 11th International Conference, ICONIP 2004, Calcutta, India, November 22-25, 2004, Proceedings*, pages 561–566, 2004.

Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 151–160, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. *CoRR*, abs/1612.00410, 2016. URL `http://arxiv.org/abs/1612.00410`.

Alexander A. Alemi, Ben Poole, Ian Fischer, Joshua V. Dillon, Rif A. Saurous, and Kevin Murphy. Fixing a broken elbo. *CoRR*, abs/1711.00464, 2017. URL `http://arxiv.org/abs/1711.00464`.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay, 2018.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. *CoRR*, abs/1609.05140, 2016. URL `http://arxiv.org/abs/1609.05140`.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

André Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Žídek, and Rémi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement, 2019.

Andrew Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications*, 13, 12 2002. doi: 10.1023/A:1025696116075.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Craig Boutilier, Ronen Brafman, and Christopher Geib. Prioritized goal decomposition of markov decision processes: Toward a synthesis of classical and decision theoretic planning. *Proc. UAI*, 2, 06 1997.

Yevgen Chebotar, Mrinal Kalakrishnan, Ali Yahya, Adrian Li, Stefan Schaal, and Sergey Levine. Path integral guided policy search. *CoRR*, abs/1610.00529, 2016. URL `http://arxiv.org/abs/1610.00529`.

Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model, 2016.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *CoRR*, abs/1506.02216, 2015. URL `http://arxiv.org/abs/1506.02216`.

Ignasi Clavera, David Held, and Pieter Abbeel. Policy transfer via modularity and reward guiding. In *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, September 2017.

Christian Daniel, Gerhard Neumann, Oliver Kroemer, and Jan Peters. Hierarchical relative entropy policy search. *Journal of Machine Learning Research*, 17(93):1–50, 2016a. URL `http://jmlr.org/papers/v17/15-188.html`.

Christian Daniel, Herke van Hoof, Jan Peters, and Gerhard Neumann. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 08 2016b. doi: 10.1007/s10994-016-5580-x.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, 1993.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. doi: 10.1111/j.2517-6161.1977.tb01600.x.

Thomas G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition, 1999.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning, 2016.

Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=S1vuO-bCW`.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2017.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pages 202–211. AUAI Press, 2016.

Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. Multi-level discovery of deep options. *CoRR*, abs/1703.08294, 2017. URL `http://arxiv.org/abs/1703.08294`.

Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. In *International Conference on Learning Representations*, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration, 2018.

Alexandre Galashov, Siddhant Jayakumar, Leonard Hasenclever, Dhruva Tirumala, Jonathan Schwarz, Guillaume Desjardins, Wojtek M. Czarnecki, Yee Whye Teh, Razvan Pascanu, and Nicolas Heess. Information asymmetry in KL-regularized RL. In *International Conference on Learning Representations*, 2019.

Anirudh Goyal, Riashat Islam, DJ Strouse, Zafarali Ahmed, Hugo Larochelle, Matthew Botvinick, Sergey Levine, and Yoshua Bengio. Transfer and exploration via the information bottleneck. In *International Conference on Learning Representations*, 2019.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. In *International Conference on Learning Representations*, 2017.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, 2018a.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1861–1870, 2018b.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications, 2019.

Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option: Learning options with a deliberation cost. *arXiv preprint arXiv:1709.04571*, 2017.

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

Nicolas Heess, Gregory Wayne, David Silver, Tim Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, 2015.

Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

Nicolas Heess, Dhruva Tirumala, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, Ali Eslami, Martin Riedmiller, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters, 2019.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A. Ortega, Yee Whye Teh, and Nicolas Heess. Meta reinforcement learning as task inference, 2019.

Auke J. Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1547–1554. MIT Press, 2003. URL `http://papers.nips.cc/paper/2140-learning-attractor-landscapes-for-learning-motor-primitives.pdf`.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog, 2019.

Matthew J Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2946–2954. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6379-composing-graphical-models-with-neural-networks-for-structured-representations-and-`
`pdf`.

Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

A. S. Klyubin, D. Polani, and C. L. Nehaniv. Empowerment: a universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 128–135 Vol.1, 2005.

Jens Kober and Jan R. Peters. Policy search for motor primitives in robotics. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 849–856. Curran Associates, Inc., 2009. URL `http://papers.nips.cc/paper/3545-policy-search-for-motor-primitives-in-robotics.pdf`.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Wouter Kool and Matthew Botvinick. Mental labour. *Nature Human Behaviour*, 2, 09 2018. doi: 10.1038/s41562-018-0401-9.

Sanjay Krishnan, Roy Fox, Ion Stoica, and Ken Goldberg. DDCO: discovery of deep continuous options forrobot learning from demonstrations. *CoRR*, abs/1710.05421, 2017. URL `http://arxiv.org/abs/1710.05421`.

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction, 2019.

Romain Laroche, Paul Trichelair, and Rémi Tachet des Combes. Safe policy improvement with baseline bootstrapping, 2017.

Sergey Levine and Vladlen Koltun. Variational policy search via trajectory optimization. In *Advances in Neural Information Processing Systems*, pages 207–215, 2013.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight, 2017.

Timothy Lillicrap, Jonathan Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, 09 2015.

Rudolf Lioutikov, Gerhard Neumann, Guilherme Maeda, and Jan Peters. Learning movement primitive libraries through probabilistic segmentation. *The International Journal of Robotics Research*, 36(8):879–894, 2017. doi: 10.1177/0278364917713116. URL `https://doi.org/10.1177/0278364917713116`.

Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. Neural probabilistic motor primitives for humanoid control. In *International Conference on Learning Representations*, 2019.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner, 2017.

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, 2015.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016. URL `http://arxiv.org/abs/1602.01783`.

Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning, 2015.

William Montgomery and Sergey Levine. Guided policy search as approximate mirror descent. *CoRR*, abs/1607.04614, 2016. URL http://arxiv.org/abs/1607.04614.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, 2016.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

Ofir Nachum, Shixiang (Shane) Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3303–3313. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7591-data-efficient-hierarchical-reinforcement-learning.pdf.

Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning?, 2019.

OpenAI. Openai five. https://blog.openai.com/openai-five/, 2018.

OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

Daniel Alexander Ortega and Pedro Alejandro Braun. Information, utility and bounded rationality. In Jürgen Schmidhuber, Kristinn R. Thórisson, and Moshe Looks, editors, *Artificial General Intelligence*, pages 269–274, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-22887-2.

Pedro A Ortega and Daniel A Braun. Thermodynamics as a theory of decision-making with information-processing costs. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 469(2153):20120683, 2013.

Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2616–2624. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5177-probabilistic-movement-primitives.pdf.

Ronald Parr and Stuart J Russell. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems*, 1998.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage weighted regression: Simple and scalable off-policy reinforcement learning, 2020. URL https://openreview.net/forum?id=H1gdF34FvS.

Jan Peters, Katharina Mülling, and Yasemin Altün. Relative entropy policy search. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, page 1607–1612. AAAI Press, 2010.

Doina Precup. *Temporal abstraction in reinforcement learning*. PhD thesis, University of Massachusetts, Amherst, USA, 2000.

Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables, 2019.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference (extended abstract). In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, page 3052–3056. AAAI Press, 2013. ISBN 9781577356332.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models, 2014.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom van de Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing solving sparse reward tasks from scratch. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Jonathan Rubin, Ohad Shamir, and Naftali Tishby. Trading value and information in mdps. *Decision Making with Imperfect Decision Makers*, pages 57–74, 2012.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks, 2016.

Christoph Salge, Cornelius Glackin, and Daniel Polani. Empowerment - an introduction. *CoRR*, abs/1310.1863, 2013. URL `http://arxiv.org/abs/1310.1863`.

T. Salimans, D. P. Kingma, and M. Welling. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. *ArXiv e-prints*, October 2014.

Jürgen Schmidhuber. Neural sequence chunkers. Technical report, Institut für Informatik, Technische Universität München, 1991.

John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017a.

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

Noah Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rke7geHtwH`.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 387–395, Bejing, China, 22–24 Jun 2014. PMLR. URL `http://proceedings.mlr.press/v32/silver14.html`.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

H.A. Simon. Rational choice and the structure of the environment. *Psychological Review*, 63(2):129–138, 1956. doi: 10.1037/h0042769.

Susanne Still and Doina Precup. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131(3):139–148, 2012.

Daniel Strouse, Max Kleiman-Weiner, Josh Tenenbaum, Matt Botvinick, and David J Schwab. Learning to share and hide intentions using information regularization. In *Advances in Neural Information Processing Systems*, 2018.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1–2): 181–211, August 1999. ISSN 0004-3702. doi: 10.1016/S0004-3702(99)00052-1. URL `https://doi.org/10.1016/S0004-3702(99)00052-1`.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind control suite. Technical report, DeepMind, January 2018. URL `https://arxiv.org/abs/1801.00690`.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

Stas Tiomkin and Naftali Tishby. A unified bellman equation for causal information and value in markov decision processes. *arXiv preprint arXiv:1703.01585*, 2017.

Naftali Tishby and Daniel Polani. Information theory of decisions and actions. *Perception-Action Cycle*, pages 601–636, 2011.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

Emanuel Todorov. Linearly-solvable markov decision problems. In *Advances in Neural Information Processing Systems*, 2007.

Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 945–952, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143963. URL `https://doi.org/10.1145/1143844.1143963`.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015. URL `http://arxiv.org/abs/1509.06461`.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John Agapiou, Max Jaderberg, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575, 11 2019. doi: 10.1038/s41586-019-1724-z.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn, 2016.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *International Conference on Learning Representations*, 2017a.

Ziyu Wang, Josh Merel, Scott E. Reed, Greg Wayne, Nando de Freitas, and Nicolas Heess. Robust imitation of diverse behaviors. *CoRR*, abs/1707.02747, 2017b.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Jost Tobias Springenberg, Scott Reed, Bobak Shahriari, Noah Siegel, Josh Merel, Caglar Gulcehre, Nicolas Heess, and Nando de Freitas. Critic regularized regression, 2020.

Marco Wiering and Jürgen Schmidhuber. Hq-learning. *Adaptive Behavior*, 6(2):219–246, 1997. doi: 10.1177/105971239700600202. URL `https://doi.org/10.1177/105971239700600202`.

RONALD J. Williams and JING Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991. doi: 10.1080/09540099108946587. URL https://doi.org/10.1080/09540099108946587.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning, 2019.

Markus Wulfmeier, Abbas Abdolmaleki, Roland Hafner, Jost Tobias Springenberg, Michael Neunert, Noah Siegel, Tim Hertweck, Thomas Lampe, Nicolas Heess, and Martin Riedmiller. Compositional transfer in hierarchical reinforcement learning. *Robotics: Science and Systems XVI*, Jul 2020a. doi: 10.15607/rss.2020.xvi.054. URL http://dx.doi.org/10.15607/rss.2020.xvi.054.

Markus Wulfmeier, Dushyant Rao, Roland Hafner, Thomas Lampe, Abbas Abdolmaleki, Tim Hertweck, Michael Neunert, Dhruva Tirumala, Noah Siegel, Nicolas Heess, and Martin Riedmiller. Data-efficient hindsight off-policy option learning, 2020b.

Saining Xie, Alexandre Galashov, Siqi Liu, Shaobo Hou, Razvan Pascanu, Nicolas Heess, and Yee Whye Teh. Transferring task goals via hierarchical reinforcement learning, 2018. URL https://openreview.net/forum?id=S1Y6TtJvG.

Denis Yarats and Ilya Kostrikov. Soft actor-critic (sac) implementation in pytorch. https://github.com/denisyarats/pytorch_sac, 2020.

Brian D Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, 2010.