Robust Predictable Control

Benjamin Eysenbach[1 2] **Ruslan Salakhutdinov**[1] **Sergey Levine**[2 3]
[1]Carnegie Mellon University, [2]Google Brain, [3]UC Berkeley
beysenba@cs.cmu.edu

Abstract

Many of the challenges facing today's reinforcement learning (RL) algorithms, such as robustness, generalization, transfer, and computational efficiency, are closely related to compression. Prior work has convincingly argued why minimizing information is useful in the supervised learning setting, but standard RL algorithms lack an explicit mechanism for compression. The RL setting is unique because (1) its sequential nature allows an agent to use past information to avoid looking at future observations and (2) the agent can optimize its behavior to prefer states where decision making requires few bits. We take advantage of these properties to propose a method (RPC) for learning *simple* policies. This method brings together ideas from information bottlenecks, model-based RL, and bits-back coding into a simple and theoretically-justified algorithm. Our method jointly optimizes a latent-space model and policy to be *self-consistent*, such that the policy avoids states where the model is inaccurate. We demonstrate that our method achieves much tighter compression than prior methods, yielding up to $5\times$ higher reward than a standard information bottleneck. As a result of this compression, the policies learned by our method are robust and generalize well to new tasks.[1]

1 Introduction

Many areas of reinforcement learning (RL) research focus on specialized problems, such as learning invariant representations, improving robustness to adversarial attacks, improving generalization, or building better world models. These problems are often symptoms of a deeper underlying problem: autonomous agents use too many bits from their environment. For the purpose of decision making, most information about the world is irrelevant. For example, a lane keeping feature on a car may take as input high-resolution camera input (millions of bits), but only needs to extract a few bits of information about the relative orientation of the car in the lane. Agents that rely on more bits of information run the risk of overfitting to the training task.

Agents that use few bits of information enjoy a number of appealing properties. These agents can better cope with high-dimensional sensory inputs (e.g., dozens of cameras on a self-driving car) and will be forced to learn representations that are more broadly applicable. Agents that throw away most information will be agnostic to idiosyncrasies in observations, providing robustness to missing or corrupted observations and allowing for transfer to different scenarios. For example, if an agent ignores 99.9% of bits, then corrupting a random bit is unlikely to change the agent's behavior. Moreover, an agent that minimizes bits will prefer states where the dynamics are easy to predict, meaning that the agent's resulting behavior will be easier to model. Thus, compression not only changes an agent's representation, but also changes its behavior: an agent that can only use a limited number of bits will avoid risky behaviors that require more bits to execute (see Fig. 3a).

The generalization and robustness of a machine learning model is directly related to the complexity of that model. Indeed, standard techniques for reducing complexity, such as the information bottleneck [1, 43], can be directly applied to the RL setting [14, 21, 31, 42]. While these approaches make

[1]Project site with videos and code: `https://ben-eysenbach.github.io/rpc`

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

the policy's action a simple function of the state, they ignore the temporal dimension of decision making. Instead, we will focus on learning policies whose temporally-extended behavior is simple, where we use compression to measure simplicity. Our key observation is that *a policy's behavior is simple if it is predictable*.

Our method improves upon prior methods that apply an information bottleneck to RL [14, 21, 31] by recognizing two important properties of the decision making setting. **First**, because agents make a sequence of decisions, they can use salient information at one time step to predict salient information at the next time step. These predictions can decrease the amount of information that the agent needs to sense from the environment. We will show that learning a predictive model is not an ad-hoc heuristic, but rather a direct consequence of minimizing information using bits-back coding [12, 19]. **Second**, unlike supervised learning, the agent can change the distribution over states, choosing behaviors that visit states that are easier to compress. For example, imagine driving on a crowded road. Aggressively passing and tailgating other cars may result in reaching the destination faster, but requires careful attention to other vehicles and fast reactions. In contrast, a policy optimized for using few bits would not pass other cars and would leave a larger following distance (see Fig. 3b). Combined, these two capabilities result in a method that jointly trains a latent space model and a control policy, with the policy being rewarded for visiting states where that model is accurate. Unlike typical model-based methods, our method explicitly optimizes for the accuracy of open-loop planning and results in a model and policy that are *self-consistent*.

The main contribution of this paper is an RL algorithm, **robust predictable control (RPC)**, for learning policies that use few bits of information. We will refer to such policies as *compressed policies*. RPC brings together ideas from information bottlenecks, model-based RL, and bits-back coding into a simple and theoretically-justified algorithm. RPC attains up to $5\times$ higher return than a standard information bottleneck when compared at the same bitrate. Experiments demonstrate that the compressed policies learned by RPC are more robust than those learned by alternative approaches, generalize well to new tasks, and learn behaviors that can be composed for hierarchical RL.

2 Related Work

Our work builds on a large body of prior work, both in supervised learning and reinforcement learning, that argues that compression is a good objective for learning representations and improving robustness [4, 19, 27, 30, 38, 39, 42, 44]. We measure compression by the mutual information between its inputs and outputs [4, 43], a metric that has been used to study the generalization properties [43] and representations learned by neural networks [1]. Our work extends these results to the RL setting by observing that the agent can change its behavior (i.e., the data distribution) to be more easily compressed. In the RL community, prior work has used the variational information bottleneck (VIB) [2] to minimize communication between agents in a multi-agent setting [45] and to improve exploration in a goal-reaching setting [14]. The most related RL methods are those that use an information bottleneck in RL to improve generalization [21, 31]. Whereas these prior methods compress observations individually, we will compress *sequences* of observations, similar to a sequential VAE [8, 23]. This difference, which corresponds to learning a latent-space model, improves compression and increases robustness on downstream tasks.

Prior work in RL has sought robustness, transfer, and generalization in many ways. For example, prior RL methods often use explicit representation learning objectives to accelerate learning of image-based tasks [13, 26, 28, 36]. The presentations learned by our method, like those learned by prior contrastive learning methods [26, 34, 36], do not require reconstructing observations. Prior work has also studied numerous strategies for learning policies that are resilient to perturbations in the environment, a problem known as robust RL [20, 22, 33, 41]. While prior robust RL methods typically involve solving a two-player game, we show that compression is a simpler mechanism for achieving some robustness benefits. Finally, the problem of learning RL policies that generalize has been studied by many prior papers [9, 10].

3 Reinforcement Learning with Fewer Bits

This section introduces the idea that predicting the future allows RL policies to operate with fewer bits. We derive this idea from first principles, develop it into a complete RL method, then discuss connections with model-based RL, bits-back coding, and other related topics.

3.1 Notation and Preliminaries

An agent interacts in an MDP defined by states $\mathbf{s_t}$ and actions $\mathbf{a_t}$; we will use "states" and "observations" interchangeably. The agent samples actions from a policy $\pi_\theta(\mathbf{a_t} \mid \mathbf{s_t}, \mathbf{s_{t-1}}, \mathbf{s_{t-2}}, \cdots)$.[2] We will construct this policy by learning an encoder $\phi(\mathbf{z_t} \mid \mathbf{s_t})$ (which *encodes* a state $\mathbf{s_t}$ into a representation $\mathbf{z_t}$), and a high-level policy $\pi^z_\theta(\mathbf{a_t} \mid \mathbf{z_t})$, which *decodes* a representation $\mathbf{z_t}$ into an action $\mathbf{a_t}$. The environment dynamics are defined by an initial state $\mathbf{s_1} \sim p_1(\mathbf{s_1})$ and a transition function $p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})$. The RL objective is to maximize the expected γ-discounted sum of rewards: $\max_\pi \mathbb{E}_\pi \left[\sum_{t=1}^\infty \gamma^t r(\mathbf{s_t}, \mathbf{a_t})\right]$. The discount factor can be viewed as saying that the episode terminates at every time step with probability $(1-\gamma)$, an interpretation we will use in Sec. 3.2.

A model is simpler if it expresses a simpler input-output relationship [4, 43]. We will measure the complexity of a function using an *information bottleneck*, which is the mutual information $I(\mathbf{x}; \mathbf{y})$ between an input $\mathbf{x}$ and an output $\mathbf{y}$ [1, 43]. The *variational information bottleneck* provides a tractable upper bound on mutual information [1, 2]:

$$I(\mathbf{x}; \mathbf{y}) \leq \mathbb{E}_{p(\mathbf{x},\mathbf{y})}\left[\log\left(\frac{p(\mathbf{y} \mid \mathbf{x})}{m(\mathbf{y})}\right)\right],$$

where $m(\mathbf{y})$ is an arbitrary prior distribution. Applying the information bottleneck to an intermediate layer $\mathbf{z} = \phi(\mathbf{x})$ is sufficient for bounding the mutual information between input x and output y.

Following prior work on compression in RL [21, 31], we aim to maximize rewards while minimizing the number of bits. While there are many ways to apply compression to RL (e.g., compressing actions, goals, or individual observations), we will focus on compressing *sequences* of states. The input is a sequence of states, $\mathbf{s_{1:\infty}} \triangleq (\mathbf{s_1}, \mathbf{s_2}, \cdots)$; the output is a sequence of actions, $\mathbf{a_{1:\infty}} \triangleq (\mathbf{a_1}, \mathbf{a_2}, \cdots)$. The objective is to learn a representation $\phi_\theta(\mathbf{z_t} \mid \mathbf{s_t})$ and policy $\pi_\theta(\mathbf{a_t} \mid \mathbf{z_t})$ that maximize reward, subject to the constraint that the policy uses on average $C > 0$ bits of information per episode:

$$\max_\theta \mathbb{E}_{\pi,\phi}\left[\sum_{t=1}^\infty \gamma^t r(\mathbf{s_t}, \mathbf{a_t})\right] \qquad \text{s.t.} \quad \mathbb{E}_\pi[I(\mathbf{s_{1:\infty}}; \mathbf{z_{1:\infty}})] \leq C, \tag{1}$$

While prior work on compression in RL [21, 31] has applied the VIB to states *independently*, we will aim to compress entire *sequences* of observations. Applying the VIB to sequences allows us to use more expressive choices of the prior $m(\mathbf{z_{1:\infty}})$. Our prior will use previous representations to *predict* the prior for the next representation, allowing us to obtain a tighter bound on mutual information.

3.2 Using Fewer Bits by Predicting the Future

Predicting the future allows agents to observe fewer bits from future observations. To further decrease the number of bits used, the agent can change its behavior to visit states that are more easily compressed. States where the dynamics are hard to predict will require more bits, so the agent will prefer visiting states where its learned model can accurately predict the next state.

This intuition corresponds to solving the optimization problem in Eq. 1 with a prior that is factored autoregressively: $m_{1:\infty}(\mathbf{z_{1:\infty}} \mid \mathbf{a_{1:\infty}}) = m(\mathbf{z_1}) \prod_t m_\theta(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})$. Note that the prior has learnable parameters θ. We apply the VIB to obtain an upper bound on the constraint in Eq. 1:

$$\mathbb{E}_\pi[I(\mathbf{s_{1:\infty}}; \mathbf{z_{1:\infty}})] \leq \mathbb{E}_\pi\left[\log\left(\frac{p(\mathbf{z_{1:\infty}} \mid \mathbf{s_{1:\infty}})}{m(\mathbf{z_{1:\infty}})}\right)\right] = \mathbb{E}_\pi\left[\sum_{t=1}^\infty \gamma^t (\log \phi_\theta(\mathbf{z_t} \mid \mathbf{s_t}) - \log m_\theta(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t}))\right]. \tag{2}$$

This objective is different from prior work that applies the VIB to RL [14, 21, 31] because the prior $m_\theta(\mathbf{z_t})$ is *predicted*, rather than fixed to be a unit Normal distribution. The discount factor γ reflects the assumption from Sec. 3.1 that the episode terminates with probability $(1-\gamma)$ at each time step; of course, no bits are used after the episode terminates. Our final objective optimizes the policy π, the encoder ϕ, and the prior m to maximize reward and minimizing information:

$$\max_\theta \mathbb{E}_{\pi_\theta,\phi_\theta,m_\theta}\left[\sum_{t=1}^\infty \gamma^t r(\mathbf{s_t}, \mathbf{a_t})\right] \qquad \text{s.t.} \quad \mathbb{E}_{\pi_\theta,\phi_\theta,m_\theta}\left[\sum_t \gamma^t \left(\log \phi_\theta(\mathbf{z_t} \mid \mathbf{s_t}) - \log m_\theta(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})\right)\right] \leq C. \tag{3}$$

For the encoder, this objective looks like a modification of the information bottleneck where the prior is *predicted* from the previous representation and action. Of course, we cannot predict the first

[2] We condition the policy on *sequences* of states because, even if the task is Markovian, the agent's choice to omit certain bits from the representation can turn an otherwise fully-observed task into a partially-observed task.

representation, so we substitute use a unit Normal distribution for the prior at initial time step. For the policy, this objective looks like the standard RL objective but with a different reward function, one that includes an additional *information cost*:

$$\tilde{r}_\lambda(\mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+1}}) \triangleq r(\mathbf{s_t}, \mathbf{a_t}) + \lambda \underbrace{(\log m_\theta(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t}) - \log \phi_\theta(\mathbf{z_{t+1}} \mid \mathbf{s_{t+1}}))}_{\text{(negative of the) information cost}}, \tag{4}$$

where λ is the cost per bit. The term $\log \phi_\theta(\mathbf{z_t} \mid \mathbf{s_t})$ corresponds to the number of bits required to represent the representation $\mathbf{z_t}$. The term $\log m_\theta(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})$ reflects how well the agent can predict the next representation. In effect, the agent has to "pay" for bits of the observation, but it gets a "refund" on bits that it predicted from the previous time step, analogous to bits-back coding [12, 19]. Said in other words, we want to minimize the *excess* bits required to infer $\mathbf{z_t}$ from the current state, relative to the number of bits required to predict $\mathbf{z_t}$ from the previous $(\mathbf{z_{t-1}}, \mathbf{a_{t-1}})$. Note that the standard VIB approaches to RL [14, 21, 31] do not receive this refund because they compress observations independently, rather than compressing *sequences* of observations.

Note that the policy, encoder, and model are all *jointly optimized* with respect to the same objective, so updates to any component increases the objective for all components. Importantly, the agent optimizes not only its representation but also its behavior to minimize this information cost: the agent learns a representation that is easily predictable and learns to visit states where that representation is easily predictable. We therefore call our method **robust predictable control (RPC)**.

4 A Practical Algorithm

Our method for optimizing (Eq. 3) is an actor-critic method applied to the information-augmented reward function in Eq. 4. We introduce a Q function,

$$Q_\psi(\mathbf{s_t}, \mathbf{a_t}) = \mathbb{E}\Big[\sum_t \gamma^t \tilde{r}(\mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+1}})\Big],$$

corresponding to the augmented reward function $\tilde{r}$. We optimize this Q function using standard temporal difference learning:

$$\mathcal{L}(\psi) = \frac{1}{2}(Q_\psi(\mathbf{s_t}, \mathbf{a_t}) - \mathbf{y_t})^2 \tag{5}$$

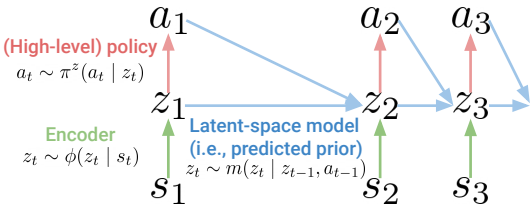


Figure 1: **Robust Predictable Control (RPC)**: Our method learns three components: an encoder $\phi(\mathbf{z_t} \mid \mathbf{s_t})$, a latent-space model $m(\mathbf{z_t} \mid \mathbf{z_{t-1}}, \mathbf{a_{t-1}})$, and a policy $\pi^z(\mathbf{a_t} \mid \mathbf{z_t})$. A conventional VIB [21, 31] omits the blue arrows.

where $\mathbf{y_t} = \lfloor \tilde{\mathbf{r}}_\mathbf{t} + \gamma Q_\psi(\mathbf{s_{t+1}}, \mathbf{a_{t+1}}) \rfloor_{\text{sg}}$ and $\lfloor \cdot \rfloor_{\text{sg}}$ is the stop-gradient operator [6, 11]. Since our overall objective (Eq. 1) only entails compressing the policy, not the Q function, we condition the Q function directly on the state. To optimize the encoder, prior, and policy, we reexpress our objective (Eq. 1) in terms of the immediate reward plus the Q function (see derivation in Appendix A.1):

$$\mathcal{L}(\theta; \mathbf{s_t}) = \mathbb{E}_{\substack{\mathbf{z_{t-1}} \sim \phi_\theta(\mathbf{s_{t-1}}), \mathbf{s_t} \sim p(\mathbf{s_t} \mid \mathbf{s_{t-1}}, \mathbf{a_{t-1}}) \\ \mathbf{z_t} \sim \phi(\mathbf{s_t}), \mathbf{a_t} \sim \pi^z_\theta(\mathbf{a_t} \mid \mathbf{z_t})}} [Q_\psi(\mathbf{s_t}, \mathbf{a_t}) + \lambda (\log m_\theta(\mathbf{z_t} \mid \mathbf{z_{t-1}}, \mathbf{a_{t-1}}) - \log \phi_\theta(\mathbf{z_t} \mid \mathbf{s_t}))]. \tag{6}$$

Since the encoder is stochastic, we compute gradients using the reparametrization trick [25]. The fact that all three components are jointly optimized with respect to the same objective makes implementation of RPC surprisingly simple. Unlike sequential VAEs and RNNs, training agents with RPC does not require sampling from $m_\theta(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})$ and does not require backpropagation through time. In our implementation, we instantiate the encoder $\phi_\theta(\mathbf{s_t})$, the prior $m_\theta(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})$, and the high-level policy $\pi^z_\theta(\mathbf{a_t} \mid \mathbf{z_t})$ as neural networks with parameters θ. The Q function $Q_\psi(\mathbf{s_t}, \mathbf{a_t})$ is likewise represented as a neural network with parameters ϕ. We update the dual parameter $\lambda \geq 0$ using dual gradient descent. We use standard tricks such as target networks and taking the minimum over two Q values. We refer the reader to Appendix B and the open-sourced code for details.

5 Connections and Analysis

In this section we discuss the connections between robust predictable control and other areas of RL. We then prove that RPC enjoys certain robustness guarantees.

5.1 Connections

This section explains connections between RPC and related ideas in the RL literature, with the aim of building intuition into how RPC works and providing an explanation for why RPC should learn robust policies and useful representations. We include a further discussion of the relationship to MaxEnt RL, the expression for the optimal encoder, and the value of information in Appendix A.3.

Model-Based RL. The prior $m_\theta(\mathbf{z_t} \mid \mathbf{z_{t-1}}, \mathbf{a_{t-1}})$ learned by RPC can be viewed as a dynamics model. Rather than predicting what the next state will be, this model predicts what the representation of the next state will be. While the model is trained to make accurate predictions, *the policy is also trained to visit states and actions where the model will be more accurate.*

Representation learning. What, precisely, does the representation $\mathbf{z_t}$ represent? Given a prior $m(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})$ and latent-conditioned policy $\pi^z(\mathbf{a_t} \mid \mathbf{z_t})$, we can use the current representation $\mathbf{z_t}$ to predict good actions at both the current time step and (by unrolling the prior) at future time steps. Thus, *the representation $\mathbf{z_t}$ can be thought of as a compact representation of open-loop action sequences*. Our experiments demonstrate that this compact representation of action sequences, once learned on one task, can be used to quickly learn a range of downstream tasks.

We can also view RPC as *learning a new action space for the MDP*, where the reward function is now $\tilde{r}$ (Eq. 4). What we previously called the encoder, $\phi(\mathbf{z_t} \mid \mathbf{s_t})$, is now the policy for selecting the actions $\mathbf{z_t}$ in this new MDP. The $-\log \phi(\mathbf{z_t} \mid \mathbf{s_t})$ term in from $\tilde{r}$ is the entropy of this encoder, so the encoder is performing MaxEnt RL on this new MDP. This new MDP encodes a strong prior for open-loop policies: sampling actions from the prior yields high reward.

Open-loop control. RPC learns a model $m(\mathbf{z_t} \mid \mathbf{z_{t-1}}, \mathbf{a_{t-1}})$ that predicts the state representation at the next time step. Thus, we can unroll our policy in an open-loop manner, without observing transitions from the true system dynamics. Because the model and policy are trained to be self-consistent, we expect that the highly compressed policies learned by RPC will perform well in this open-loop setting, as compared to uncompressed policies (see experiments in Sec. 6).

5.2 Theoretical Guarantees

We now present analysis showing that the compressed policies learned by RPC generalize better. We also discuss a close connection between compression and open-loop control, proving that RPC learns policies and models that are self-consistent in a way that guarantees good performance when even when the agent cannot observe new observations. We do not intend to to analyze all benefits deriving from model compression (see, e.g., [3, 4]). All proofs are in Appendix C.

In supervised learning, it is well known that compression results in models that generalize better from the training set to the testing set. Our main result applies this same reasoning to RL. We assume that the policy has been trained on an empirical distribution of MDPs, and will be evaluated on a new MDP sampled from that sample distribution. For the following result, we assume that the given stochastic MDP is a mixture of deterministic MDPs, each described by a b-bit random string (see Ng and Jordan [35]).

Lemma 5.1. *Let stochastic MDP $\mathcal{M}$ and policy π be given. Define $R^\pi(\mathcal{M})$ to be the expected reward on MDP $\mathcal{M}$. Then the probability that the policy's expected return on an observed (deterministic) MDP $\hat{\mathcal{M}}$ is much different than the policy's expected return on the stochastic MDP is bounded by the policy's bitrate:*

$$P_{\hat{\mathcal{M}}}[|R^\pi(\hat{\mathcal{M}}) - R^\pi(\mathcal{M})| > \epsilon] \leq \frac{C+1}{2b\epsilon^2 - 1}.$$

The proof is a direct application of Bassily et al. [4, Theorem 8]. This result may be of interest in the offline RL setting, where observed trajectories effectively constitute a deterministic MDP; that policies that use fewer bits will be less likely to overfit to the offline dataset.

Not only do compressed policies generalize to new MDPs, they are also robust to missing observations. Intuitively, a policy that uses zero bits of information will perform identically in the open-loop setting. The following result shows that our model compression objective corresponds to maximizing a lower bound on the expected return of the open-loop policy. Let π^{open} be the open-loop policy corresponding to the composition of the prior $m(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})$ and the high-level policy $\pi^z(\mathbf{a_t} \mid \mathbf{z_t})$. Let π^{reactive}

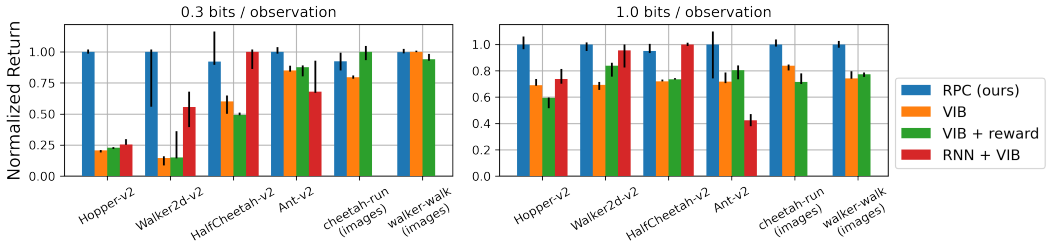


Figure 2: **Learning Compressed Policies.** We measure the return achieved by policies constrained to have a fixed bit rate. Especially at low bit rates, RPC achieves higher return than alternative methods.

be the reactive policy corresponding to the composition of the encoder $\phi(\mathbf{z_t} \mid \mathbf{s_t})$ and the high-level policy $\pi^z(\mathbf{a_t} \mid \mathbf{z_t})$.

Lemma 5.2. *Let encoder $\phi(\mathbf{z_t} \mid \mathbf{s_t})$, policy $\pi^z(\mathbf{a_t} \mid \mathbf{z_t})$, prior $m(\mathbf{z_t} \mid \mathbf{z_{t-1}}, \mathbf{a_{t-1}})$, and reward function $r(\mathbf{s_t}, \mathbf{a_t}) > 0$ be given. Then our model compression objective (Eq. 3) with reward function $(1-\gamma)\log r(\mathbf{s_t}, \mathbf{a_t})$ is a lower bound on the expected return of the open-loop policy.*

$$\mathbb{E}_{\pi^{open}(\tau)}\left[\sum_{t=1}^{\infty}\gamma^t r(\mathbf{s_t}, \mathbf{a_t})\right] \geq f\left(\mathbb{E}_{\pi^{reactive}(\tau)}\left[\sum_{t=1}^{\infty}\gamma^t\Big((1-\gamma)\log r(\mathbf{s_t}, \mathbf{a_t}) + \log m(\mathbf{z_t} \mid \mathbf{z_{t-1}}, \mathbf{a_{t-1}}) - \log \phi(\mathbf{z_t} \mid \mathbf{s_t})\Big)\right]\right),$$

where $f(x) = \frac{\gamma}{1-\gamma}e^{\frac{x}{\gamma}}$ is a monotone increasing function of x.

Moreover, the policies learned by RPC will receive similar returns in settings where they can and cannot observe observations, a result that may be useful for quantifying the regret incurred when using the learned representation $\mathbf{z_t}$ as an action space of temporally-extended behaviors. Let $R_{\text{max}} = \max_\tau R(\tau)$ be the maximum return of any trajectory.

Lemma 5.3. *The expected return of the open-loop policy π^{open} is at most $R_{max}\sqrt{C/2}$ worse than the expected return of the reactive policy $\pi^{reactive}$:*

$$\mathbb{E}_{\pi^{open}}\left[\sum_t \gamma^t r(\mathbf{s_t}, \mathbf{a_t})\right] \geq \mathbb{E}_{\pi^{reactive}}\left[\sum_t \gamma^t r(\mathbf{s_t}, \mathbf{a_t})\right] - R_{max}\sqrt{C/2}.$$

In summary, our theoretical results suggest that compressed policies generalize well and, by virtue of their open-loop guarantees, may be useful for planning. We emphasize that the theoretical benefits of model compression have been well studied in the supervised learning literature. As our method is likewise performing model compression, we expect that it will inherit a wide range of additional guarantees, such as guarantees about sample complexity.

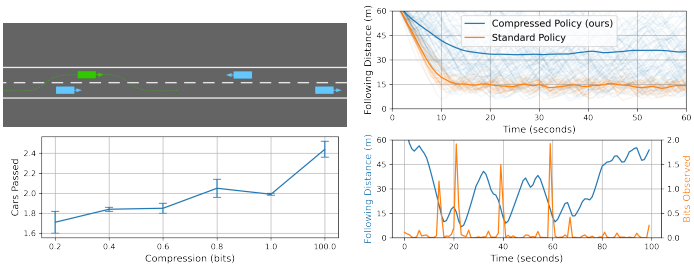
6 Experiments

Our experiments have two aims. First, we will demonstrate that RPC achieves better compression than alternative approaches, obtaining a higher reward for the same number of bits. Second, we will study the empirical properties of compressed policies learned by our method, such as their robustness and ability to learn representations suitable for hierarchical RL. We do not intend this section to exhaustively demonstrate every possible benefit from compression; we acknowledge that there are many purported benefits of compression, such as exploration and sample efficiency, which we do not attempt to study here. We include additional experiments in Appendix B.

6.1 Evaluating Compression

Our first experiment studies whether RPC outperforms alternative approaches to compression, which we summarize in the inline table. We compare against a standard VIB [21] and an extension that adds the information cost to the reward, "VIB+reward" [31]. This baseline can be viewed as a special case of RPC where the blue arrows in Fig. 1 are removed. Finally, since the predictive prior in RPC is similar to a recurrent network, we compare against an extension of the VIB [21] that uses an LSTM ("RNN + VIB"). Unlike RPC, this baseline requires training on entire trajectories (rather than individual transitions) and requires backpropagating gradients through time. We evaluate all methods on four tasks from

	feedforward architecture	predicted prior	augmented reward
RPC (ours)	✓	✓	✓
VIB [21]	✓	✗	✗
VIB+reward [31]	✓	✗	✗
VIB+RNN	✗	✗	✓



(a) Driving with Traffic (b) Active Cruise Control

Figure 3: **Behavior of compressed policies**: On two driving tasks, we observe that highly compressed policies *(Left)* avoid passing other cars and *(Right)* leave a larger following distance between cars. The passing and tailgating that our method forgoes would require more bits of information about the precise locations of the other cars.

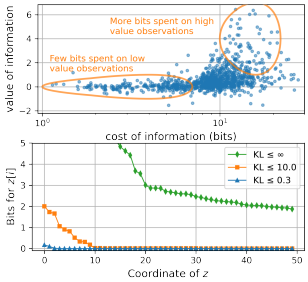


Figure 4: **Representations**: *(Top)* Compressed policies observe more bits from observations that have a large value of information. *(Bottom)* Compressed policies learn sparse representations.

OpenAI-Gym [5] and two *image-based* tasks from dm-control [40]. Because of computational constraints, we omit the RNN+VIB baseline on the image-based tasks.

We plot results in Fig. 2. To make the rewards comparable across tasks, we normalize the total return by the median return of the best method. On almost all tasks, RPC achieves higher returns than prior methods for the same bitrate. For example, when learning the Walker task using 0.3 bits per observation, RPC achieves a return of 3,562, the VIB+RNN baseline achieves a return of 1,978 (-44%), and the other VIB baselines achieve a return of around 530 (-85%). We include full results on a wider range of bitrates in Appendix Fig. 10. While, in theory, the VIB+RNN baseline could implement RPC internally, in practice it achieved lower returns, perhaps because of optimization challenges associated with training LSTMs [7]. Even if the VIB+RNN baseline could implement the strategy learned by RPC, RPC is simpler (it does not require training on trajectories) and trains about 25% times faster (it does not require backpropagation through time).

6.2 Visualizing Compressed Policies

Behavior of compressed policies. To visualize how compression changes behavior, we applied RPC to two simulated driving tasks shown in Fig. 3a (top left), which are based on prior work [29]. In the first task, the agent can pass cars by driving into the lane for oncoming traffic; the second task uses the same simulated environment but restricts the agent to remain in its own lane to study active cruise control in isolation. The rewards for these tasks corresponds to driving to the right as quickly as possible, without colliding with any other vehicles. In the first task, we observe that compressed policies passed fewer cars than uncompressed policies. Intuitively, a passing maneuver requires many bits of information about the relative positions of other cars, so we expect that compressed policies to engage in fewer passing maneuvers. Fig. 3a (bottom) shows that the bitrate of RPC is directly correlated with the number of cars passed. In the second task, we observe that compressed policies leave a larger following distance from the leading car (Fig. 3b *(top)*). Fig. 3b *(bottom)* shows that the number of bits used per observation increases when the car is within 15m of another car. Thus, by maintaining a following distance of more than 30m, the compressed policy can avoid these situations where it would have to use more bits. See the project website for videos.[3]

Representations of compressed policies. A policy learned by compression must trade off between maximizing reward and paying to receive bits of information. Using the `HalfCheetah-v2` task, we plot the value of information versus cost of information for many sampled states. See Appendices A.4 and B.5 for details. As shown in Fig. 4 *(top)*, the RPC uses more bits from observations that are more valuable for maximizing future returns. Then, to visualize the learned representation $\mathbf{z_t}$, we sample the representation from randomly sampled states, and plot the numbers of bits used for each coordinate. Fig. 4 *(bottom)* shows that RPC learns sparse representations. Whereas the uncompressed policy uses all coordinates, a policy compressed with bitrate 10 uses only $10/50$ coordinates and a policy compressed with bitrate 0.3 uses only $2/50$ coordinates.

Intuitively, the ways in which compression changes the behavior and representations of policies should make these policies more robust to disturbances. Our next experiments explicitly test this hypothesis, which would validate the analysis in Sec. 5.2.

[3]Project site with videos and code: `https://ben-eysenbach.github.io/rpc`

6.3 Robustness

The connection between compression and robustness has been well established in the literature (e.g., [16, 47]). Our next set of experiments test the robustness of compressed policies to different types of disturbances: missing observations (i.e., open-loop control), adversarial perturbations to the observations, and perturbations to the dynamics (i.e., robust RL). We emphasize that, because these experiments focus on robustness, the policy is trained and tested in different environments.

Robustness to missing observations and open-loop control. Since compressed policies rely on fewer bits of input from the observations, we expect that they not only will be less sensitive to missing observations, but also will actively modify their behavior to adopt strategies that require fewer bits from the observation. In this experiment, we drop each observation independently with probability $p \in [0, 1]$, where $p = 1$ corresponds to using a fully open-loop policy. For RPC, we handle missing observations by predicting the representation from the previous time step. Our two main baselines take a policy used by standard RL and learn either a latent-space model or a state-space model. When the observation is missing, these baselines make predictions using the learned model. We also compare against RNN+VIB, the strongest baseline from Fig. 2. When observations are missing, the LSTM's input for that time step is sampled from the prior. Fig. 5 shows that all methods perform similarly when no observations are dropped, but RPC achieves a higher reward than baselines when a larger fraction of observations are dropped. This experiment shows that more effective compression, as done by RPC, yields more robust policies. This robustness may prove useful in real-world environments where sensor measurements are missing or corrupted.

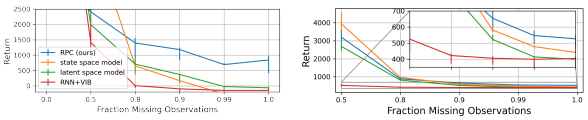


Figure 5: **Robustness to missing observations**: RPC is more robust to missing observations than prior methods, including those that learn dynamics models. We show `HalfCheetah-v2` on left and `Walker2d-v2` on right.

Adversarial Robustness. Compressed policies extract fewer bits from each observation, so we expect that compressed policies will be more robust to adversarial perturbations to the observation. While prior work has proposed purpose-designed methods for achieving robustness [33, 41], here we investigate whether compression is simple yet effective means for achieving a modicum of robustness. We do not claim that compression is the best method for learning robust policies.

We first study adversarial perturbations to the **dynamics** on the `Ant-v2` environment. Given a policy $\pi(\mathbf{a_t} \mid \mathbf{s_t})$ and the current state $\mathbf{s_t}$, the adversary aims to apply a small perturbation to that state to make the policy perform as poorly as possible. We implement the adversary using projected gradient descent [32]; see Appendix B for details. Fig. 6 *(left)* shows the expected return as we increase the magnitude of the attack. The compressed policy is more resilient to larger attacks than the uncompressed policy. Our next experiment looks at perturbations to **observations**. Unlike the previous experiment, we let the adversary perturb *every* step in an episode; see Appendix B for full details. Fig. 6 (right) shows that policies that use fewer bits achieve higher returns in this adversarial setting. When the `Ant-v2` agent uses too many bits (3 or more), the adversarial perturbations to the dynamics flip the agent over, whereas agents that use fewer bits remain upright.

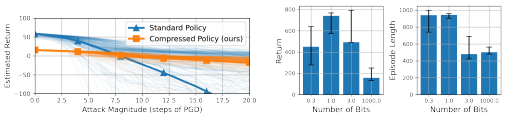


Figure 6: **Adversarial robustness**: Compressed policies are more robust against adversarial attacks to the *(Left)* dynamics and *(Right)* observations.

Robust RL. Our final set of experiments look at higher-level perturbations to the dynamics, as are typically studied in the robust RL community [33, 41]. Using the same `Ant-v2` environment as before, we *(1)* increase the mass of each body element by a fixed multiplier, or *(2)* decrease the friction of each body geometry by a fixed multiplier. These experiments test whether the learned policies are robust to more massive robots or more "slippery" settings. Fig. 7 shows that compressed policies generalize to larger masses and smaller frictions more effectively than an uncompressed policy. Comparing the policies learned

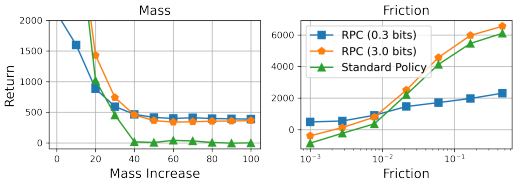


Figure 7: **Robust RL**: Compressed policies are more robust to increases in mass and decreases in friction.

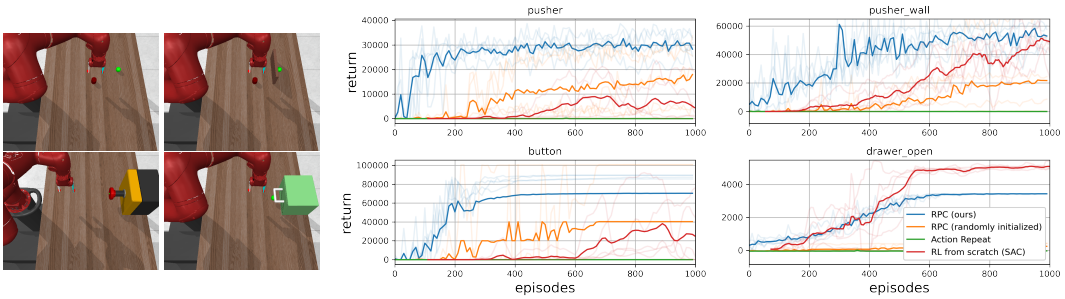


Figure 8: **Hierarchical RL**: We apply RPC to the pushing task shown in the top-left. We then use the learned representation of action sequences as the action space for solving three new tasks. On all tasks, the representations learned by RPC accelerate learning. All episodes have a fixed length.

by RPC with a bit rate of 0.3 bits versus 3.0 bits, we observe that the bit rate effectively balances performance versus robustness. See Appendix Fig. 11 for a larger version of this plot with error bars.

6.4 Hierarchical RL using the Learned Representation

RPC learns a representation of temporally-extended action sequences (see Sec. 5.1). We hypothesize that these representations can accelerate the learning of new tasks in a hierarchical setting. Our goal is not to propose a complete hierarchical RL system, but rather evaluate whether these action representations are suitable for high-level control. During training, we apply RPC to a goal-conditioned object pushing task, shown in Fig. 8 *(top-left)*. The initial position of the object and the goal position are randomized, so we expect that different representations $\mathbf{z_t}$ will correspond to high-level behaviors of moving the end effector and object to different positions. At test-time, the agent is presented with a new task. The agent will attempt to solve the task by commanding one or two behaviors $\mathbf{z_t}$. See Appendix B for details and pseudocode.

We compare the action representations learned by RPC to three baselines. The first baseline is "Action Repeat", which constantly outputs the same action, effectively allowing the agent to command desired poses. To test whether RPC has learned a prior over *useful* action sequences, we use a variant of RPC ("RPC (randomly initialized)") where the policy and model are randomly initialized. Finally, "RL from scratch" applies a state-of-the-art off-policy RL algorithm (SAC [17]) to the task. Taken together, these baselines allow us to study whether RPC not only learns behaviors that do more than move to particular poses, but also learns behaviors for obstacle avoidance and object manipulation. We apply all methods to four tasks and present results in Fig. 8. First, as a sanity check, we apply all methods to the training task, finding that RPC quickly finds a single $\mathbf{z_t}$ that solves the task. On the remaining three tasks, we likewise observe that the representations learned by RPC allow for much faster learning than the baselines. The final task, "pusher wall" requires chaining together multiple representations $\mathbf{z_t}$ to solve. While the "RL from scratch" baseline eventually matches and then surpasses the performance of RPC, RPC accelerates learning in the low-data regime.

7 Conclusion

In this paper, we presented a method for learning robust and predictable policies. Our objective differs from prior work by compressing *sequences* of observations, resulting in a method that jointly trains a policy and a model to be self-consistent. Not only does our approach achieve better compression than prior methods, it also learns policies that are more robust (Fig. 6). We also demonstrate that our method learns representations that are suitable for use in hierarchical RL.

Our work suggests a few areas for future work. Ideas based on RPC may be useful for model-based RL, as training the policy to be self-consistent with the model may help prevent model exploitation. Similarly, because the skills learned by RPC are optimized to be predictable, we expect that they will perform well when used for open-loop planning. As compression is closely related to the energy required to implement a function on an ideal physical system [37], RPC might be useful to building energy-efficient control policies.

Limitations. The main limitation of this work is that policies that use few bits will often receive lower reward on the training tasks. Also, for the purpose of exploration, the most informative states may be those that are hardest to compress. While the first limitation is likely irreconcilable, the second might be lifted by *maximizing* information collected during exploration but *minimizing* information for policy optimization.

Acknowledgments. We thank Dibya Ghosh and the members of the Salakhutdinov lab for feedback throughout the project. We thank Oscar Ramirez for help setting up the image-based experiments and thank Rico Jonschkowski for introductions to potential collaborators. We thank Ryan Julian, Vincent Vanhoucke, and anonymous reviewers for feedback on the paper, and thank Abhishek Gupta for discussions during the early stages of this project.

This material is supported by the Fannie and John Hertz Foundation and the NSF GRFP (DGE1745016).

References

[1] Achille, A. and Soatto, S. (2018). Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980.

[2] Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. (2017). Deep variational information bottleneck. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

[3] Arora, S., Ge, R., Neyshabur, B., and Zhang, Y. (2018). Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning*, pages 254–263. PMLR.

[4] Bassily, R., Moran, S., Nachum, I., Shafer, J., and Yehudayoff, A. (2018). Learners that use little information. In *Algorithmic Learning Theory*, pages 25–55. PMLR.

[5] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

[6] Chen, X. and He, K. (2021). Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758.

[7] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Moschitti, A., Pang, B., and Daelemans, W., editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1724–1734. ACL.

[8] Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. *Advances in neural information processing systems*, 28:2980–2988.

[9] Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. (2019). Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR.

[10] Farebrother, J., Machado, M. C., and Bowling, M. (2018). Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*.

[11] Foerster, J., Farquhar, G., Al-Shedivat, M., Rocktäschel, T., Xing, E., and Whiteson, S. (2018). Dice: The infinitely differentiable monte carlo estimator. In *International Conference on Machine Learning*, pages 1529–1538. PMLR.

[12] Frey, B. J. and Hinton, G. E. (1997). Efficient stochastic source coding and an application to a bayesian network source model. *The Computer Journal*, 40:157–165.

[13] Gelada, C., Kumar, S., Buckman, J., Nachum, O., and Bellemare, M. G. (2019). Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pages 2170–2179. PMLR.

[14] Goyal, A., Islam, R., Strouse, D., Ahmed, Z., Larochelle, H., Botvinick, M., Levine, S., and Bengio, Y. (2019). Transfer and exploration via the information bottleneck. In *International Conference on Learning Representations*.

[15] Guadarrama, S., Korattikara, A., Ramirez, O., Castro, P., Holly, E., Fishman, S., Wang, K., Gonina, E., Wu, N., Kokiopoulou, E., Sbaiz, L., Smith, J., Bartók, G., Berent, J., Harris, C., Vanhoucke, V., and Brevdo, E. (2018). TF-Agents: A library for reinforcement learning in tensorflow. `https://github.com/tensorflow/agents`. [Online; accessed 25-June-2019].

[16] Gui, S., Wang, H. N., Yang, H., Yu, C., Wang, Z., and Liu, J. (2019). Model compression with adversarial robustness: A unified optimization framework. *Advances in Neural Information Processing Systems*, 32:1285–1296.

[17] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR.

[18] Hansen, N. (2006). The cma evolution strategy: a comparing review. *Towards a new evolutionary computation*, pages 75–102.

[19] Hinton, G. E. and Van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13.

[20] Huang, S. H., Papernot, N., Goodfellow, I. J., Duan, Y., and Abbeel, P. (2017). Adversarial attacks on neural network policies. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

[21] Igl, M., Ciosek, K., Li, Y., Tschiatschek, S., Zhang, C., Devlin, S., and Hofmann, K. (2019). Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in Neural Information Processing Systems*, 32:13978–13990.

[22] Kamalaruban, P., Huang, Y., Hsieh, Y., Rolland, P., Shi, C., and Cevher, V. (2020). Robust reinforcement learning via adversarial training with langevin dynamics. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[23] Ke, N. R., Singh, A., Touati, A., Goyal, A., Bengio, Y., Parikh, D., and Batra, D. (2019). Learning dynamics model in reinforcement learning by incorporating the long term future. *arXiv preprint arXiv:1903.01599*.

[24] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR (Poster)*.

[25] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

[26] Laskin, M., Srinivas, A., and Abbeel, P. (2020). Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR.

[27] LeCun, Y., Denker, J. S., and Solla, S. A. (1990). Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605.

[28] Lee, A., Nagabandi, A., Abbeel, P., and Levine, S. (2020). Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33.

[29] Leurent, E. (2018). An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env`.

[30] Littlestone, N. and Warmuth, M. (1986). Relating data compression and learnability.

[31] Lu, X., Lee, K., Abbeel, P., and Tiomkin, S. (2020). Dynamics generalization via information bottleneck in deep reinforcement learning. *arXiv preprint arXiv:2008.00614*.

[32] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[33] Morimoto, J. and Doya, K. (2005). Robust reinforcement learning. *Neural computation*, 17(2):335–359.

[34] Nachum, O., Gu, S., Lee, H., and Levine, S. (2019). Near-optimal representation learning for hierarchical reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[35] Ng, A. Y. and Jordan, M. I. (2000). PEGASUS: A policy search method for large mdps and pomdps. In Boutilier, C. and Goldszmidt, M., editors, *UAI '00: Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence, Stanford University, Stanford, California, USA, June 30 - July 3, 2000*, pages 406–415. Morgan Kaufmann.

[36] Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

[37] Ortega, P. A. and Braun, D. A. (2013). 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.

[38] Schmidhuber, J. (1992). Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242.

[39] Shamir, O., Sabato, S., and Tishby, N. (2010). Learning and generalization with the information bottleneck. *Theoretical Computer Science*, 411(29-30):2696–2711.

[40] Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. (2018). Deepmind control suite. *arXiv preprint arXiv:1801.00690*.

[41] Tessler, C., Efroni, Y., and Mannor, S. (2019). Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pages 6215–6224. PMLR.

[42] Tishby, N. and Polani, D. (2011). Information theory of decisions and actions. In *Perception-action cycle*, pages 601–636. Springer.

[43] Tishby, N. and Zaslavsky, N. (2015). Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE.

[44] Veness, J., Bellemare, M. G., Hutter, M., Chua, A., and Desjardins, G. (2015). Compress and control. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[45] Wang, R., He, X., Yu, R., Qiu, W., An, B., and Rabinovich, Z. (2020). Learning efficient multi-agent communication: An information bottleneck approach. In *International Conference on Machine Learning*, pages 9908–9918. PMLR.

[46] Yarats, D., Kostrikov, I., and Fergus, R. (2021). Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*.

[47] Ye, S., Xu, K., Liu, S., Cheng, H., Lambrechts, J.-H., Zhang, H., Zhou, A., Ma, K., Wang, Y., and Lin, X. (2019). Adversarial robustness vs. model compression, or both? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 111–120.

[48] Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. (2020). Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR.

A More Analysis

A.1 Objective for the Encoder, Model, and Policy

This section describes how the objective for the encoder, model, and policy (Eq. 6) is derived from our overall objective (Eq. 1). Our aim is to maximizing the sum of (information-augmented) rewards (Eq. 3), starting at state $\mathbf{s_t}$:

$$
\begin{aligned}
\mathcal{L}(\theta; \mathbf{s_t}) &= E\left[\sum_{t'=t}^{\infty} \gamma^{t'-t} \tilde{r}(\mathbf{s_{t'}}, \mathbf{a_{t'}}) \mid \mathbf{s_t}\right] \\
&= E\left[\sum_{t'=t}^{\infty} \gamma^{t'-t} (r(\mathbf{s_{t'}}, \mathbf{a_{t'}}) + \lambda(\log m_\theta(\mathbf{z_{t'+1}} \mid \mathbf{z_{t'}}, \mathbf{a_{t'}}) - \log \phi_\theta(\mathbf{z_{t'}} \mid \mathbf{s_{t'}}))) \mid \mathbf{s_t}\right] \\
&= E\left[r(\mathbf{s_t}, \mathbf{a_t}) + \lambda(\log m_\theta(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t}) - \log \phi_\theta(\mathbf{z_t} \mid \mathbf{s_t})) \right. \\
&\qquad \left. + \sum_{t'=t+1}^{\infty} \gamma^{t'-t} (r(\mathbf{s_{t'}}, \mathbf{a_{t'}}) + \lambda(\log m_\theta(\mathbf{z_{t'+1}} \mid \mathbf{z_{t'}}, \mathbf{a_{t'}}) - \log \phi_\theta(\mathbf{z_{t'}} \mid \mathbf{s_{t'}}))) \mid \mathbf{s_t}\right] \\
&= E\left[r(\mathbf{s_t}, \mathbf{a_t}) + \lambda(\log m_\theta(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t}) - \log \phi_\theta(\mathbf{z_t} \mid \mathbf{s_t})) + \gamma Q_\psi(\mathbf{s_{t+1}}, \mathbf{a_{t+1}})\right].
\end{aligned}
$$

Note that the reward at the current time step, $r(\mathbf{s_t}, \mathbf{a_t})$, is not influenced by the parameters θ, so we can drop this term:

$$
\mathcal{L}(\theta; \mathbf{s_t}) \overset{\text{const.}}{=} E\left[\lambda(\log m_\theta(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t}) - \log \phi_\theta(\mathbf{z_t} \mid \mathbf{s_t})) + \gamma Q_\psi(\mathbf{s_{t+1}}, \mathbf{a_{t+1}})\right].
$$

The remaining difference between this objective and Eq. 5 is that the Q value term is scaled by γ. However, this difference has no effect on the optimization problem because the parameter λ is automatically tuned. If we scale the second term, Q, by some value (say, γ), then tuning λ to satisfy the bitrate constraint will result in a different value for λ (one which is γ times smaller). For optimizing this objective, we sample states $\mathbf{s_t}$ from the replay buffer.

A.2 Prior on the Initial Representation

For simplicity, we have omitted the prior $m(\mathbf{z_1})$ on the representation of the first observation. This prior cannot be predicted from prior observations. Instead, we fix $m(\mathbf{z_1})$ to be a zero-mean, unit-variance Gaussian. The information-augmented reward at the first time step is therefore

$$
\tilde{r}_\lambda(\mathbf{s_0}, \mathbf{a_0}, \mathbf{s_1}) \triangleq \cancel{r(\mathbf{s_0}, \mathbf{a_0})} + \lambda\left(\log m_\theta(\mathbf{z_1}) - \log \phi_\theta(\mathbf{z_1} \mid \mathbf{s_1})\right).
$$

A.3 MaxEnt RL and the Optimal Encoder

Maximum entropy (MaxEnt) RL is a special case of our compression objective. If the high-level policy is the identity function ($\pi^z(\mathbf{a_t} \mid \mathbf{z_t}) = \delta(\mathbf{a_t} = \mathbf{z_t})$) and the prior is the uniform distribution over $\mathcal{Z}$ (i.e., $m(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t}) = \text{Unif}(\mathbf{z_{t+1}})$), then we recover standard MaxEnt RL. Another way of describing this connection is that MaxEnt RL is equivalent to imposing an information bottleneck on the final (i.e., action) output of the policy. This connection suggests that some of the empirically-observed benefits of MaxEnt RL might be derived from the fact that it implicitly is performing model compression. For example, the objective from MaxEnt RL can be interpreted as optimizing for behavior under which random actions (sampled from the prior) do not significantly decrease the expected reward. Nonetheless, directly optimizing for the model compression objective yields a prior that depends on time. As shown in our experiments, such a prior yields a policy that not only is more compressed, but is also more robust to sensor failure (i.e., the open-loop setting).

In MaxEnt RL, the optimal policy can be expressed in terms of the *soft* value function: $\pi(\mathbf{a_t} \mid \mathbf{s_t}) \propto e^{\tilde{Q}(\mathbf{s_t}, \mathbf{a_t})}$, where $\tilde{Q}(\mathbf{s_t}, \mathbf{a_t})$ is the expected, entropy-regularized return. We can express the optimal encoder learned by RPC in similar terms. First, we reparametrize the Q function in terms of $\mathbf{z_t}$ instead of $\mathbf{a_t}$: $Q^\pi(\mathbf{s_t}, \mathbf{z_t}) = \mathbb{E}_{\pi^z(\mathbf{a_t} \mid \mathbf{z_t})}[Q(\mathbf{s_t}, \mathbf{a_t})]$. The optimization problem for the encoder is

$$
\begin{aligned}
&\max_{\phi(\mathbf{z_t} \mid \mathbf{s_t})} \mathbb{E}_{\phi(\mathbf{z_t} \mid \mathbf{s_t})}\left[\sum_t \gamma^t \tilde{r}(\mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+1}})\right] \\
&\qquad = \mathbb{E}_{\phi(\mathbf{z_t} \mid \mathbf{s_t})}\left[r(\mathbf{s_t}, \mathbf{a_t}) + \log m(\mathbf{z_t} \mid z_{t-1}, a_{t-1}) - \log \phi(\mathbf{z_t} \mid \mathbf{s_t}) + \gamma Q(\mathbf{s_{t+1}}, \mathbf{a_{t+1}})\right].
\end{aligned}
$$

Algorithm 1 Robust Predictable Control. The updates below are written using a learning rate of η. In practice we perform gradient steps using the Adam [24] optimizer.

Initialize prior $m_\theta(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})$, encoder $\phi_\theta(\mathbf{z_{t+1}} \mid \mathbf{s_t})$, Q function $Q_\psi(\mathbf{s_t}, \mathbf{a_t})$, and high-level policy $\pi^z_\theta(\mathbf{a_t} \mid \mathbf{z_t})$.
Initialize replay buffer $\mathcal{D} \leftarrow \emptyset$
Initialize dual variable $\log \lambda \leftarrow \log(1e-6)$
while not converged **do**
    Sample a batch of transitions: $\{(\mathbf{s_t}, \mathbf{a_t}, \mathbf{r_t}, \mathbf{s_{t+1}}) \sim \mathcal{D}\}$
    Compute information cost: $\mathbf{c_t} \leftarrow \mathbb{E}[\log \phi_\theta(\mathbf{z_t} \mid \mathbf{s_t}) - \log m_\theta(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})]$ ▷ Eq. 4
    Compute information-regularized reward $\tilde{\mathbf{r}} \leftarrow \mathbf{r_t} - \lambda \mathbf{c_t}$
    Update Q function: $\psi \leftarrow \psi - \eta \nabla_\psi \mathcal{L}(\psi)$ ▷ Eq. 5
    Update prior, encoder, and high-level policy: $\theta \leftarrow \theta + \eta \nabla_{theta} \mathcal{L}(\mathbf{s_t})$ ▷ Eq. 6
    Update dual variable: $\log \lambda \leftarrow \log \lambda - \eta(C - \mathbb{E}[c_t])$
return $\pi^z_\theta(\mathbf{a_t} \mid \mathbf{z_t}), \phi_\theta(\mathbf{z_t} \mid \mathbf{s_t}), m_\theta(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})$

Using calculus of variations, we determine that the optimal encoder is given by

$$\phi(\mathbf{z_t} \mid \mathbf{s_t}) = \frac{m(\mathbf{z_t} \mid \mathbf{z_{t-1}}, \mathbf{a_{t-1}}) e^{Q^\pi(\mathbf{s_t}, \mathbf{z_t})}}{\int m(\mathbf{z_t}' \mid \mathbf{z_{t-1}}, \mathbf{a_{t-1}}) e^{Q^\pi(\mathbf{s_t}, \mathbf{z_t}')} d\mathbf{z_t}'}.$$

Thus, the optimal encoder is trained to *tilt* the predictions from the prior by the Q function.

A.4 Value of information.

An optimal agent must balance these information costs against the *value of information* gained from these observations. Precisely, the value of information is how much more reward an optimal agent could receive if it observes the representation $\mathbf{z_t}$ instead of predicting $\mathbf{z_t}$ from the previous representation and action. We expect that the optimal policy will only look at representations where the value of information is greater than the cost of information. We confirm this prediction experimentally in Fig. 4. In practice, the policy learned by RPC looks at every observation, but may only look at a few bits from that observation.

B Experimental Details

B.1 Implementation Details

We implemented RPC on top of the SAC [17] implementation in TF-Agents [15]. Unless otherwise noted, we used the default hyperparameters from that implementation and trained all agents for 3e6 steps. We provide pseudocode in Alg. 1.

State-based RPC. We parameterized the model $m_\theta(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})$ by predicting the *difference* between the $\mathbf{z_t}$ and $\mathbf{z_{t+1}}$. That is, we trained a 2-layer neural network (both layers have 256 units with ReLU activations) to model $p(\mathbf{z_{t+1}} - \mathbf{z_t} \mid \mathbf{z_t}, \mathbf{a_t})$. We parameterized the encoder $\phi_\theta(\mathbf{z_t} \mid \mathbf{s_t})$ as a 2-layer neural network (both layers have 256 units with ReLU activations). Both the model and encoder output the mean and (diagonal) standard deviation of a multivariate Normal distribution. We squashed the mean to be within $[-30, 30]$ and squashed the standard deviation to be within $[0.1, 10.0]$. We used the following function for squashing:

```
def squash_to_range(t, low=-np.inf, high=np.inf):
  if low == -np.inf:
    t_low = t
  else:
    t_low = -low * tf.nn.tanh(t / (-low))
  if high == np.inf:
    t_high = t
  else:
    t_high = high * tf.nn.tanh(t / high)
  return tf.where(t < 0, t_low, t_high)
```

Unless otherwise noted, we set the dimension of $\mathbf{z_t}$ to 50, though found this parameter has little effect.

We parameterized the high-level policy $\pi_\theta^z(\mathbf{a_t} \mid \mathbf{s_t})$ as a 2-layer neural network (both layers have 256 units with ReLU activations). Following the TF-Agents implementation of SAC, this network first predicts a Normal distribution, which is then squashed by `tf.tanh` to only output actions within the action space.

Image-based RPC. Our image-based version of RPC was based on DrQ [46]. Unless otherwise specified, we used the same hyperparameters and architecture as that paper. For example, we used the same action repeat (4 actions) and frame stack (3 frames) as that paper. The only architectural difference was the policy network. We parameterized the encoder $\phi_\theta(\mathbf{z_t} \mid \mathbf{s_t})$ as follows:

```
tf.keras.Sequential([
  tf.keras.layers.Lambda(lambda t: tf.cast(t, tf.float32) / 255.0),  # Normalize image
  tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), strides=(2, 2), activation='relu'),
  tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), strides=(1, 1), activation='relu'),
  tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), strides=(1, 1), activation='relu'),
  tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), strides=(1, 1), activation='relu'),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(tfp.layers.IndependentNormal.params_size(latent_dim), activation=_activation),
  tfp.layers.IndependentNormal(latent_dim),
])
```

We used the same model and high-level policy as in the state-based experiments.

Dual parameter λ. We updated the dual parameter λ via dual gradient ascent to satisfy the bitrate constraint. We parametrized this variable as $\log \lambda$ to ensure that it remains positive. We initialized $\lambda = 10^{-6} \approx 0$, and performed updates using `tf.keras.optimizers.Adam(learning_rate=3e-4)`. We confirmed via visual inspection that the bitrate constraint was satisfied.

RNN baseline. We based the RNN baseline experiments on SAC-RNN implementation in TF-Agents [15].[4] We trained the agent on sequences of length 20. For fair comparison, we used the same number of *transitions* per batch as other agents that trained on individual transitions. The default batch size in TF-Agent's implementation of SAC is 256, so we trained the RNN on batches with 256 // 20 = 13 sequences.

Model-based baselines. The state-space model and latent-space models used the same model architecture as RPC. This model was trained using standard maximum likelihood with the same optimizer that RPC used for optimizing $m(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})$.

B.2 Learning Compressed Policies (Fig. 2)

We ran this experiment with 5 random seeds, with error bars depicting the [25%, 75%] quantiles across random seeds. To compute the return of each policy, we took the average of the last 50 evaluations, corresponding to the last 5e5 training steps (out of a total of 3e6 training steps). For each task, we normalized the return by the median return of the best performing baseline. Fig. 10 plots returns without normalization.

B.3 Behavior of Compressed Policies (Fig. 3a (bottom))

We used the `two-way` task from Leurent [29]. We initially observed that the agent often crashed into other cars, so we modified the reward function by adding a penalty of -50 when the agent crashes. The original environment has discrete actions, so we added a wrapper around the environment that allows RPC to command continuous actions, which would then be discretized. We trained RPC for 1e5 steps on this environment.

B.4 Behavior of Compressed Policies (Fig. 3b)

For this experiment, we used a modified version of the `two-way` task from Leurent [29]. We modified the environment to disable passing, so the car was forced to remain in the same lane. To encourage

[4] https://github.com/tensorflow/agents/blob/master/tf_agents/agents/sac/examples/v2/train_eval_rnn.py

aggressive driving, we modified the reward function to be $(x/2000)^{10}$, where x is the distance traveled. We trained RPC for 2e5 steps. The "Compressed Policy" refers to RPC using a bitrate constraint of 0.1.

B.5 Value of Information (Fig. 4 (top))

The value of observing $\mathbf{z_t}$ is given by

$$\text{ValueOfInfo}(\mathbf{z_{t+1}}) = \mathbb{E}\left[\sum_t \gamma^t r(\mathbf{s_t}, \mathbf{a_t}) \mid \mathbf{z_{t+1}}\right] - \mathbb{E}\left[\sum_t \gamma^t r(\mathbf{s_t}, \mathbf{a_t}) \mid \mathbf{z_t}\right],$$

while the cost of observing $\mathbf{z_t}$ is given by

$$\text{CostOfInfo}(\mathbf{z_{t+1}}) = \mathbb{E}[\log \phi(\mathbf{z_{t+1}} \mid \mathbf{s_{t+1}}) - \log m(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})].$$

Intuitively, the optimal policy will only choose to look at representations $\mathbf{z_t}$ when $\text{ValueOfInfo}(\mathbf{z_{t+1}}) > \text{CostOfInfo}(\mathbf{z_{t+1}})$. We compare these two quantities in the experiments (Fig. 4 (top)). Note that both the cost of information and value of information depend on the encoder, $\phi(\mathbf{s_t})$.

For this experiment, we applied RPC to the `HalfCheetah-v2` environment using a bitrate constraint of 10.0.

B.6 Sparse Representations (Fig. 4 (bottom))

For this experiment, we applied RPC to the `HalfCheetah-v2` environment. Since both the encoder $\phi(\mathbf{z_t} \mid \mathbf{s_t})$ and model $m(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})$ predict *diagonal* normal distributions, we can compute the KL divergence between each coordinate. For the plot in Fig. 4 (bottom), we sorted these KLs before plotting.

B.7 Hierarchical RL (Fig. 8)

Pre-training. We learned the representation of actions by applying RPC to the `push-v2` task in Metaworld [48]. To make the subsequent optimization easier, we set the dimension of $\mathbf{z_t}$ to 3.

Hierarchical RL. We optimize over the list of $\mathbf{z_t}$s using CMA-ES [18]. We provide pseudocode in Fig. 9. The aim of this approach is to test whether compression results in behaviors that can transfer to new tasks. We use CMA-ES as a black-box optimizer because of its simplicity, and emphasize that more sophisticated optimization routines (e.g., actor-critic RL with actions corresponding to $\mathbf{z_t}$) would likely perform better.

For the `pusher` task (which was the same as the training task), we used one behavior $\mathbf{z_t}$ for a horizon of 100 steps. For the `pusher_wall` task, we used two behaviors $\mathbf{z_t}$, each with horizon 100 steps. For the `button` task, we used one behavior $\mathbf{z_t}$ for a horizon of 100 steps. For the `drawer_open` task, we used two behaviors $\mathbf{z_t}$, each with a horizon of 75 steps.

```
def objective_fn(z_list, horizon):
  env.reset()
  total_r = 0.0
  for z in z_list:
    for _ in range(horizon):
      a = policy(z)
      _, r, done, _ = env.step(a)
      total_r += r
      if done:
        break
      z = prior(z, a)
  loss = -1 * total_r
cma.fmin(objective_fn)
```

Figure 9: Pseudocode for hierarchical RL

Action Repeat. In some tasks, actions correspond to a desired pose. Thus, action repeat would correspond to moving to a desired pose, which seemed like a reasonable baseline in these manipulation tasks. Our experiments highlight that the behaviors learned by RPC do more than move to a particular pose. In addition, the behaviors learned by RPC do obstacle avoidance and objective manipulation.

B.8 Robustness to missing observations (Fig. 5)

For this experiment, we trained RPC using a bitrate of 0.1 of `HalfCheetah-v2` and 3.0 for `Walker2d-v2`. We evaluated all methods by dropping observations independently. Note that none of the methods were trained using observation dropout, so this experiment explicitly tests robustness to new disturbances introduced at test time.

B.9 Adversarial Robustness to Dynamics (Fig. 6 (left))

The adversary aims to apply a small perturbation to that state to make the policy perform as poorly as possible. The adversary's objective is

$$\min_{\mathbf{s}_{\text{adv}} \in \mathcal{B}(\mathbf{s})} V^{\pi}(\mathbf{s}_{\text{adv}}) = Q^{\pi}(\mathbf{s}_{\text{adv}}, \mathbf{a} \sim \pi(\mathbf{a} \mid \mathbf{s}_{\text{adv}})).$$

We instantiate the adversary using projected gradient descent [32] with step size 0.1. For fair comparison, we evaluate each policy on states collected by rolling out that policy. This experiment used the `Ant-v2` environment, applying RPC using a bitrate of 0.3. We repeated this attack on 20 states sampled from each policy's state distribution. The dark line shows the average across these 20 attacks. As expected, the agent is more vulnerable to attacks in some states than in other states.

B.10 Adversarial Robustness to Observations (Fig. 6 (right))

We optimize the adversary to corrupt the state using the following objective:

$$\min_{\mathbf{s}_{\text{adv}} \in \mathcal{B}(\mathbf{s})} Q^{\pi}(\mathbf{s}, \mathbf{a} \sim \pi(\mathbf{a} \mid \mathbf{s}_{\text{adv}})).$$

The adversary is optimized using PGD, using three steps of size 0.1 for each state. Note that we only change the observation, not the true state of the environment. This difference from the previous experiment is important, as it allows us to unroll the policy and adversary for an entire trajectory. We used the `Ant-v2` environment for this experiment.

B.11 Robust RL (Fig. 7)

We followed prior work in setting up the robust RL experiments [41]. To modify the mass, we scaled the `env.model.body_mass` attribute of the environment by a fixed constant. To modify friction, we scaled the `env.model.geom_friction` attribute of the environment by a fixed constant. We used perturbations much larger than prior work [41] because we found that standard RL was already robust to smaller ranges of perturbations.

C Proofs

C.1 Model Compression is a Lower Bound on Open-Loop Control (Lemma 5.2)

In this section we formally define the assumptions for Lemma 5.2 and then provide the proof.

First, we define an open loop policy with hidden state $\mathbf{z_t}$. This policy updates its hidden state as $m(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})$ and produces actions by sampling $p(\mathbf{a_t} \mid \mathbf{z_t})$. Note that the open-loop sequence of actions generated by the policy is evaluated under the *true* system dynamics, $p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})$. The open loop policy does not observe the transitions from the true system dynamics.

The main idea of the proof will be to apply an evidence lower bound on the expected reward objective. This idea alone almost completes the proof. The main technical challenge is accounting for discount factors: a naïve application of an ELBO will result in discounted rewards but an undiscounted information term.

Proof. The **first step** is to recognize that the expected discounted return objective can be written as the expected *terminal* reward of a mixture of finite-length episodes. Define $p_H(\tau)$ as a distribution over length-H episodes. We can then write the expected discounted return objective as follows:

$$\mathbb{E}_{p(\tau)}\left[\sum_{t=1}^{\infty} \gamma^t r(\mathbf{s_t}, \mathbf{a_t})\right] = \sum_{H=1}^{\infty} \gamma^H \mathbb{E}_{p_H(\tau)}\left[r(\mathbf{s_H}, \mathbf{a_H})\right].$$

We can now obtain a lower bound on the *log* of the expected return objective:

$$\begin{aligned}
&\log \mathbb{E}_{p(\mathbf{s_{1:\infty}}, \mathbf{a_{1:\infty}}, \mathbf{z_{1:\infty}})} \left[\sum_{t=1}^{\infty} \gamma^t r(\mathbf{s_t}, \mathbf{a_t}) \right] \\
&= \log \left(\sum_{H=1}^{\infty} \gamma^H \mathbb{E}_{p(\mathbf{s_{1:H}}, \mathbf{a_{1:H}}, \mathbf{z_{1:H}})} \left[r(\mathbf{s_H}, \mathbf{a_H}) \right] \right) \\
&= \log \left(\frac{1-\gamma}{\gamma} \sum_{H=1}^{\infty} \gamma^H \mathbb{E}_{p(\mathbf{s_{1:H}}, \mathbf{a_{1:H}}, \mathbf{z_{1:H}})} \left[r(\mathbf{s_H}, \mathbf{a_H}) \right] \right) + \log \frac{\gamma}{1-\gamma} \\
&\geq \frac{(1-\gamma)}{\gamma} \sum_{H=1}^{\infty} \gamma^H \log \left(\mathbb{E}_{\mathbf{p}(\mathbf{s_{1:H}}, \mathbf{a_{1:H}}, \mathbf{z_{1:H}})} \left[r(\mathbf{s_H}, \mathbf{a_H}) \right] \right) + \log \frac{\gamma}{1-\gamma}. \qquad (7)
\end{aligned}$$

The inequality is Jensen's inequality. The constant $\frac{1-\gamma}{\gamma}$ was introduced because Jensen's inequality must be applied to a proper probability distribution.

The **second step** is to define a variational distribution $q(\mathbf{s_{1:\infty}}, \mathbf{a_{1:\infty}}, \mathbf{z_{1:\infty}})$ as

$$q(\mathbf{s_{1:\infty}}, \mathbf{a_{1:\infty}}, \mathbf{z_{1:\infty}}) = \prod_t p(\mathbf{a_t} \mid \mathbf{z_t}) q(\mathbf{z_t} \mid \mathbf{s_t}) p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}).$$

We can then obtain a lower bound on Eq. 7

$$\begin{aligned}
&\log \mathbb{E}_{p(\mathbf{s_{1:\infty}}, \mathbf{a_{1:\infty}}, \mathbf{z_{1:\infty}})} \left[\sum_{t=1}^{\infty} \gamma^t r(\mathbf{s_t}, \mathbf{a_t}) \right] \\
&\geq \frac{1-\gamma}{\gamma} \sum_{H=1}^{\infty} \gamma^H \mathbb{E}_{q(\mathbf{s_{1:H}}, \mathbf{a_{1:H}}, \mathbf{z_{1:H}})} \left[\log r(\mathbf{s_H}, \mathbf{a_H}) + \log p(\mathbf{s_{1:H}}, \mathbf{a_{1:H}}, \mathbf{z_{1:H}}) - \log q(\mathbf{s_{1:H}}, \mathbf{a_{1:H}}, \mathbf{z_{1:H}}) \right] + \log \frac{\gamma}{1-\gamma} \\
&= \frac{1-\gamma}{\gamma} \sum_{H=1}^{\infty} \gamma^H \mathbb{E}_{q(\mathbf{s_{1:H}}, \mathbf{a_{1:H}}, \mathbf{z_{1:H}})} \left[\log r(\mathbf{s_H}, \mathbf{a_H}) + \sum_{t=1}^{H} \log m(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t}) - \log \phi(\mathbf{z_t} \mid \mathbf{s_t}) \right] + \log \frac{\gamma}{1-\gamma}. \qquad (8)
\end{aligned}$$

The final line follows from simplifying the definitions of $p(\mathbf{s_{1:\infty}}, \mathbf{a_{1:\infty}}, \mathbf{z_{1:\infty}})$ and $q(\mathbf{s_{1:\infty}}, \mathbf{a_{1:\infty}}, \mathbf{z_{1:\infty}})$. Note that we have used the assumption that $r(\mathbf{s_t}, \mathbf{a_t}) > 0$ to ensure that $\log r(\mathbf{s_t}, \mathbf{a_t})$ is well defined.

Our **third step** is to do the opposite of the first step: recognize that expectations over a mixture of finite-horizon episodes are equivalent to expectations over a discounted infinite-length episode. With this, we can rewrite Eq. 8 as follows:

$$\begin{aligned}
&\log \mathbb{E}_{p(\mathbf{s_{1:\infty}}, \mathbf{a_{1:\infty}}, \mathbf{z_{1:\infty}})} \left[\sum_{t=1}^{\infty} \gamma^t r(\mathbf{s_t}, \mathbf{a_t}) \right] \\
&\geq \frac{1-\gamma}{\gamma} \mathbb{E}_{q(\mathbf{s_{1:\infty}}, \mathbf{a_{1:\infty}}, \mathbf{z_{1:\infty}})} \left[\sum_{H=1}^{\infty} \gamma^H \left(\log r(\mathbf{s_H}, \mathbf{a_H}) + \sum_{t=1}^{H} \log m(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t}) - \log \phi(\mathbf{z_t} \mid \mathbf{s_t}) \right) \right] + \log \frac{\gamma}{1-\gamma}. \qquad (9)
\end{aligned}$$

We can simplify the double summation using the following identity, where $\mathbf{x_t}$ is shorthand for $\log m(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t}) - \log \phi(\mathbf{z_t} \mid \mathbf{s_t})$:

$$\begin{aligned}
\sum_{H=1}^{\infty} \gamma^H \sum_{t=1}^{H} \mathbf{x_t} &= \gamma(\mathbf{x_1}) + \gamma^2(\mathbf{x_1} + \mathbf{x_2}) + \gamma^3(\mathbf{x_1} + \mathbf{x_2} + \mathbf{x_3}) + \cdots \\
&= \mathbf{x_1}(\gamma + \gamma^2 + \gamma^3 + \cdots) + \mathbf{x_2}(\gamma^2 + \gamma^3 + \cdots) + \cdots \\
&= \mathbf{x_1} \frac{\gamma}{1-\gamma} + \mathbf{x_2} \frac{\gamma^2}{1-\gamma} + \mathbf{x_3} \frac{\gamma^3}{1-\gamma} + \cdots \\
&= \frac{1}{1-\gamma} \sum_{t=1}^{\infty} \gamma^t \mathbf{x_t}.
\end{aligned}$$

Applying this identity to Eq. 9, we get

$$\log \mathbb{E}_{p(\mathbf{s_{1:\infty}}, \mathbf{a_{1:\infty}}, \mathbf{z_{1:\infty}})} \left[\sum_{t=1}^{\infty} \gamma^t r(\mathbf{s_t}, \mathbf{a_t}) \right]$$
$$\geq \frac{1-\gamma}{\gamma} \mathbb{E}_{q(\mathbf{s_{1:\infty}}, \mathbf{a_{1:\infty}}, \mathbf{z_{1:\infty}})} \left[\sum_{t=1}^{\infty} \gamma^t \left(\log r(\mathbf{s_t}, \mathbf{a_t}) + \frac{1}{1-\gamma} \log m(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t}) - \frac{1}{1-\gamma} \log \phi(\mathbf{z_t} \mid \mathbf{s_t}) \right) \right] + \log \frac{\gamma}{1-\gamma}$$
$$\geq \frac{1}{\gamma} \mathbb{E}_{q(\mathbf{s_{1:\infty}}, \mathbf{a_{1:\infty}}, \mathbf{z_{1:\infty}})} \left[\sum_{t=1}^{\infty} \gamma^t \left((1-\gamma) \log r(\mathbf{s_t}, \mathbf{a_t}) + \log m(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t}) - \log \phi(\mathbf{z_t} \mid \mathbf{s_t}) \right) \right] + \log \frac{\gamma}{1-\gamma}.$$

Exponentiating both sides, we obtain the desired result.

□

This derivation is useful because it relates model compression to open-loop control. Precisely, it says that a compressed model is one that will perform well under open loop rollouts.

C.2 Proof of Lemma 5.3

Proof. To start, we define the distribution over trajectories $\tau = (\mathbf{s_{1:\infty}}, \mathbf{a_{1:\infty}}, \mathbf{z_{1:\infty}})$ produced by π^{open} and π^{reactive}:

$$p^{\text{open}}(\tau) = p_1(\mathbf{s_1}) \prod_t p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}) \pi^z(\mathbf{a_t} \mid \mathbf{z_t}) m(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})$$
$$p^{\text{reactive}}(\tau) = p_1(\mathbf{s_1}) \prod_t p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}) \pi^z(\mathbf{a_t} \mid \mathbf{z_t}) \phi(\mathbf{z_t} \mid \mathbf{s_t}).$$

Note that $R(\tau) \triangleq \sum_t \gamma^t r(\mathbf{s_t}, \mathbf{a_t})$ is a deterministic function of a random variable, τ. We can now write the difference in returns as $\mathbb{E}_{\pi^{\text{reactive}}}[R(\tau)] - \mathbb{E}_{\pi^{\text{open}}}[R(\tau)]$. The main idea of our proof will be to show that this difference is not too large.

$$\begin{aligned}
\mathbb{E}_{\pi^{\text{reactive}}}[R(\tau)] - \mathbb{E}_{\pi^{\text{open}}}[R(\tau)] &= \int R(\tau)(p^{\text{reactive}}(\tau) - p^{\text{open}}(\tau)) d\tau \\
&\overset{(a)}{\leq} R_{\text{max}} \int |(p^{\text{reactive}}(\tau) - p^{\text{open}}(\tau))| d\tau \\
&\overset{(b)}{\leq} R_{\text{max}} \sqrt{\frac{1}{2} KL(p^{\text{reactive}}(\tau) \| p^{\text{open}}(\tau))}. \qquad (10)
\end{aligned}$$

We used Hölder's inequality in *(a)* and Pinsker's inequality in *(b)*. Our second step is to relate the KL divergence to the compression objective.

$$\begin{aligned}
&KL(p^{\text{reactive}}(\tau) \| p^{\text{open}}(\tau)) \\
&= \mathbb{E}_{p^{\text{reactive}}} \Bigg[\cancel{\log p_1(\mathbf{s_1})} + \sum_t \cancel{p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})} + \cancel{\log \pi^z(\mathbf{a_t} \mid \mathbf{z_t})} + \log \phi(\mathbf{z_t} \mid \mathbf{s_t}) \\
&\qquad - \cancel{\log p_1(\mathbf{s_1})} - \sum_t \left(\cancel{p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})} + \cancel{\log \pi^z(\mathbf{a_t} \mid \mathbf{z_t})} + \log m(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t}) \right) \Bigg] \\
&= \mathbb{E}_{p^{\text{reactive}}} \left[\sum_t \log \phi(\mathbf{z_t} \mid \mathbf{s_t}) - \log m(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t}) \right] \leq C.
\end{aligned}$$

where C is the information constraint (Eq. 3). We then substitute this simplified expression for the KL into Eq. 10:

$$\mathbb{E}_{\pi^{\text{reactive}}}[R(\tau)] - \mathbb{E}_{\pi^{\text{open}}}[R(\tau)] \leq R_{\text{max}} \sqrt{\frac{1}{2} C}.$$

Rearranging terms, we obtain the desired result:

$$\mathbb{E}_{\pi^{\text{open}}}[R(\tau)] \geq \mathbb{E}_{\pi^{\text{reactive}}}[R(\tau)] - R_{\text{max}} \sqrt{\frac{1}{2} C}.$$

□

For this result to be non-vacuous, we need that $\sqrt{\frac{1}{2}C} < 1$, so the average per-transition c must satisfy $c = (1-\gamma)C \leq 2 \cdot (1-\gamma)$:

$$\mathbb{E}_\pi[\log \phi(\mathbf{z_t} \mid \mathbf{s_t}) - \log m(\mathbf{z_{t+1}} \mid \mathbf{z_t}, \mathbf{a_t})] \leq 2 \cdot (1-\gamma).$$

For example, when $\gamma = 0.99$, we would need that the average per-transition KL be less than 0.02.

D Additional Experiments

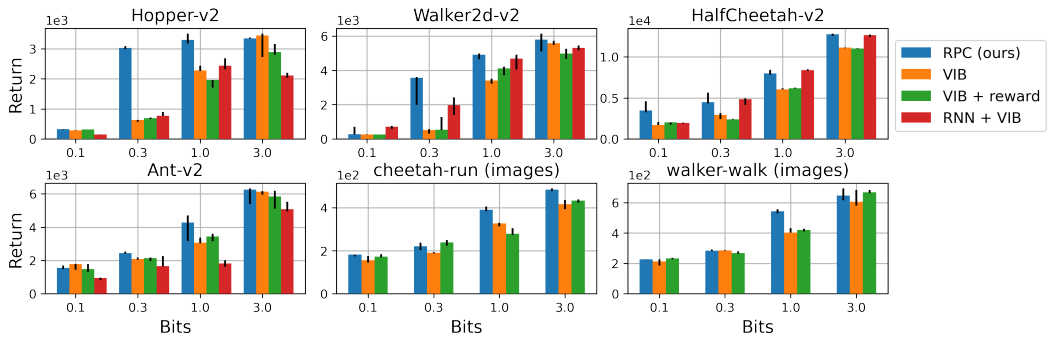


Figure 10: **Learning Compressed Policies.** This plot shows the same experiment as Fig. 2 on a wider range of bitrates, without using normalized returns.

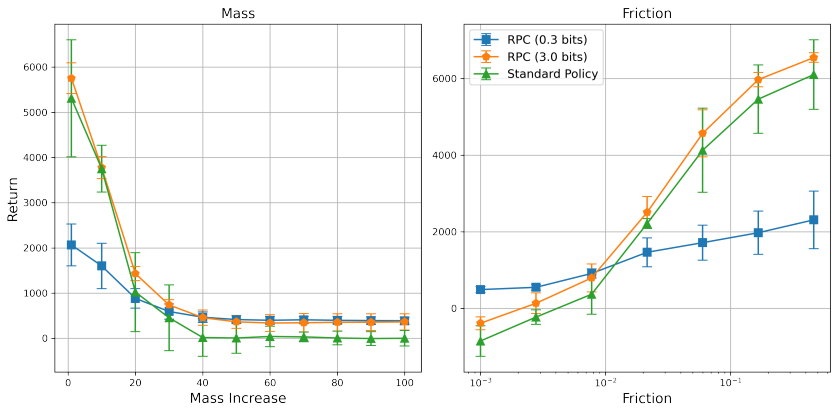


Figure 11: **Robust RL**: This plot shows the same experiment as Fig. 7 with error bars corresponding to the standard deviation across five *training* random seeds.

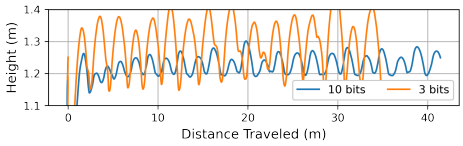


Figure 12: Behavior of compressed walking policies.

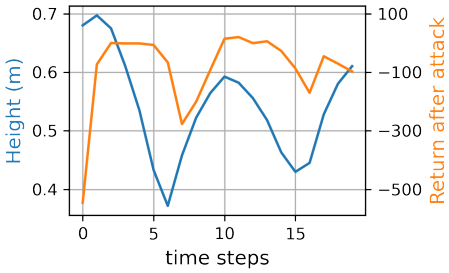


Figure 13: **Adversarial Robustness**: The agent is most vulnerable just before its feet leave the ground

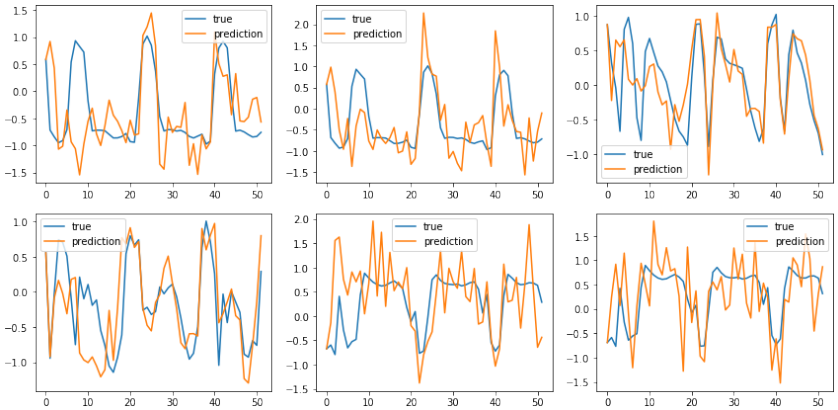


Figure 14: **Open-loop predictions of representations**

Visualizing more compressed policies. In Fig. 12, we demonstrate that compressed policies also learn different walking gaits on a locomotion task. The agent requires many more bits each time it touches the ground, so the compressed policy takes bigger steps to minimize the number of footsteps. See the project website for videos comparing the behaviors learned for varying levels of compression.

Behavior of attacked policies. We visualize the behavior of the adversarial attack on dynamics described in Sec. 6.3. Visualizing the uncompressed policy in Fig. 13, we observe that the policy is most vulnerable to attack the moment the robot launches off the ground for each step. This makes sense, as the actions right before takeoff dictate the path taken when the robot is in the air before the next step.

Open-loop predictions of representations. In Fig. 14, we plot the 6 coordinates of the representation $\mathbf{z_t}$ learned by RPC for the `HalfCheetah-v2` task. Recall that RPC learns a sparse representation, so we choose the 6 coordinates with the largest variance. We observe that open-loop (i.e., autoregressive) predictions from the latent-space model, m_θ, closely match the true representations.