Robust Predictable Control

Anonymous Author(s)
Affiliation
Address
email

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, achieving up to $5\times$ higher reward than a standard information bottleneck. We also demonstrate that our method learns policies that are more robust and generalize better to new tasks.[1]

1 Introduction

Many areas of reinforcement learning (RL) research focus on specialized problems, such as learning good 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 gain 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 better transfer to different scenarios. For example, if an agent ignores 99.9% of bits, then corrupting a random bit is very 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 (e.g., the information bottleneck [1,

[1] Project site with videos and code: `http://rpc-anonymous.github.io/`

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

36]) can be directly applied to the RL setting [12, 18, 25]. While these approaches make 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. Our key observation is that *a policy is simple if it is predictable*.

Our method improves upon prior methods that apply an information bottleneck to RL [12, 18, 25] 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 we 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 [10, 16]. **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 a method, **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 achieves much higher compression than prior methods; for example, RPC achieves $\sim 5\times$ higher return than a standard information bottleneck compared at the same bitrate. Experiments demonstrate practical benefits from the compressed policies learned by RPC: they are more robust than those learned by alternative approaches, generalize well to new tasks, and learn representations that can be composed for hierarchical RL.

2 Related Work

The problem of learning simple models has a long history in the machine learning community [16, 21]. Simplicity is often measured by the mutual information between its inputs and outputs [5, 36], a metric that has been used to study study the generalization properties [36] and representations learned by [1] neural networks. 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) [3] to minimize communication between agents in a multi-agent setting [37] and to improve exploration in a goal-reaching setting [12]. The most related RL methods are those that use an information bottleneck in RL to improve generalization [18, 25]. Whereas these prior methods compress observations individually, we will compress *sequences* of observations. This difference, which corresponds to learning a latent-space model, improves compression and increases robustness on downstream tasks.

In the RL setting, compression offers a tool for studying problems such as representation learning, robustness and generalization. Prior RL methods have used mutual information to learn representations that are good for control [11, 20, 22, 31] Our method learns a representation of observations that, like contrastive learning methods [20, 29, 31], avoids the need to reconstruct the observation. While these contrastive methods maximize the mutual information between representations across time, our method will minimize the mutual information between observations and representations. Moreover, we will joint optimize the policy and representation using a single, unified objective. Robust RL studies the problem of learning RL policies that are resilient against perturbations to the environment [17, 19, 28, 35]. While prior robust RL methods typically involve solving a two-player game, we show that compression is a simpler mechanism to achieving some robustness benefits. The problem of learning RL policies that generalize has been studied by many prior papers [8, 9]. We will show that compression also yields RL policies that generalize well.

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 s_t and actions a_t. The agent samples actions from a policy $\pi_\theta(a_t \mid s_t, s_{t-1}, s_{t-2}, \cdots)$. We will construct this policy by learning an encoder $\phi(z_t \mid s_t)$ (which produces a z_t representation of the current state s_t), and a high-level policy $\pi_\theta^z(a_t \mid z_t)$ (which chooses actions using this representation). The environment dynamics are defined an initial state $s_1 \sim p_1(s_1)$ and a transition function $p(s_{t+1} \mid s_t, a_t)$. The standard RL objective is to maximize the expected γ-discounted sum of rewards $r(s_t, a_t)$: $\max_\pi \mathbb{E}_\pi \left[\sum_{t=1}^\infty \gamma^t r(s_t, a_t)\right]$. The discount factor can be interpreted 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 [5, 36]. We will measure the complexity of a function using an *information bottleneck*, which is the mutual information $I(x; y)$ between an input x and an output y [1, 36]. Mutual information is closely tied to the energy required to implement that function on an ideal physical system [27, 32]. The *variational information bottleneck* provides a tractable upper bound on mutual information [1, 3]:

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

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

Following prior work on compression in RL [18, 25], we aim to maximizing rewards while minimizing 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, $s_{1:\infty} \triangleq (s_1, s_2, \cdots)$; the output is a sequence of actions, $a_{1:\infty} \triangleq (a_1, a_2, \cdots)$. The objective is to learn a representation $\phi_\theta(z \mid s)$ and policy $\pi_\theta(a \mid z)$ that maximize reward, subject to the constraint that the policy uses on average $C > 0$ bits of information per episode:

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

While prior work on compression in RL [18, 25] 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 $m(z_{1:\infty})$. Our prior will use previous representations to *predict* the prior for the next representation, which allow us to obtain a tighter bound on mutual information.

3.2 Using Fewer Bits by Predicting the Future

The main idea of our method is that if the agent can accurately predict the future, then the agent will not need to observe as many bits from future observations. Precisely, the agent will learn a latent dynamics model that predicts the next representation using the current representation and action. In addition to predicting the future, the agent can also decrease the number of bits by changing its behavior. 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 exactly to solving the optimization problem in Eq. 1 with a prior that is factored autoregressively: $m_{1:\infty}(z_{1:\infty}) = \prod_t m_\theta(z_{t+1} \mid z_t, a_t)$. Note that the prior has learnable parameters θ. We apply the VIB to obtain an upper bound on the constraint in Eq. 1:

$$\log \left(\frac{p(z_{1:\infty} \mid s_{1:\infty})}{m(z_{1:\infty})}\right) = \sum_{t=1}^\infty \gamma^t \underbrace{(\log \phi_\theta(z_t \mid s_t) - \log m_\theta(z_{t+1} \mid z_t, a_t))}_{\text{information cost}}. \tag{2}$$

This objective is different from prior work that applies the VIB to RL [12, 18, 25] because the prior $m_\theta(z_t)$ 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 r to maximize reward and minimizing information:

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

For the encoder, this looks like a modification of the information bottleneck where the prior is *predicted* from the previous representation and action. This objective can be understood as maximizing the information-regularized reward function:

$$\tilde{r}_\lambda(s_t, a_t) \triangleq r(s_t, a_t) + \lambda \left(\log m_\theta(z_t \mid z_{t-1}, a_{t-1}) - \log \phi_\theta(z_t \mid s_t)\right). \tag{4}$$

where λ reflects the cost per bit. The information cost has two terms. The second term, $\log \phi_\theta(z_t \mid s_t)$, corresponds to the number of bits required to represent the representation z_t. The agent wants to minimize this number of bits. The second term, $\log m_\theta(z_{t+1} \mid z_t, a_t)$, reflects the agent's uncertainty about what the next representation will be. Precisely, it indicates the number of *additional* bits the agent must obtain from the next observation. In effect, the agent does not have to pay for observing bits from the current observation that it could have predicted: that is, the agent gets a "refund" on bits that it predicted from the previous time step, analogous to bits-back coding [10, 16]. Note that the standard VIB approaches to RL [12, 18, 25] do not receive this refund because they compress observations independently, rather than compressing *sequences* of observations. 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 objective (Eq. 3) can be understood as an actor-critic method for maximizing the following information-augmented reward function in Eq. 4. We introduce a Q function $Q_\psi(s_t, a_t) = \mathbb{E}\left[\sum_t \gamma^t \tilde{r}(s_t, a_t)\right]$ for estimating the expected future returns of information-regularized reward function $\tilde{r}$. We optimize this Q function using standard temporal difference learning:

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

where $y_t = \tilde{r}_t + \gamma Q_\psi(s_{t+1}, a_{t+1})$. Note that the Q-function is conditioned directly on the state, not the compressed representation z_t. In effect, we provide the Q-function with "side" information that is not available to the policy, similarly to prior work [2, 24, 33]. A representation z_t sufficient for predicting the Q value needs at least as many bits as a representation sufficient for predicting the action, as the action can be computed from the Q value. Note that only the policy (not the Q-function) is used at deployment, so the information constraint is satisfied at deployment.

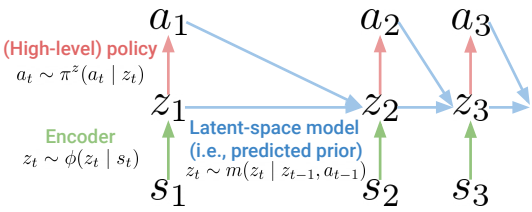


Figure 1: **Robust Predictable Control (RPC)**: Our method learns three components: an encoder $\phi(z_t \mid s_t)$, a latent-space model $m(z_t \mid z_{t-1}, a_{t-1})$, and a high-level policy. All three are trained to be self-consistent: the representation of the next state should be equal to the representation predicted by the model. In contrast, a conventional VIB [18, 25] omits the blue arrows.

The encoder, prior, and policy are jointly optimized by taking gradients on the following objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{z_{t-1}\sim\phi_\theta(z_{t-1}), s_t\sim p(s_t|s_{t-1},a_{t-1}) \\ z_t\sim\phi(s_t), a_t\sim\pi^z(a_t|z_t)}}\left[Q_\psi(s_t, a_t) + \lambda \left(\log m_\theta(z_t \mid z_{t-1}, a_{t-1}) - \log \phi_\theta(z_t \mid s_t)\right)\right]. \tag{5}$$

Since the encoder is stochastic, we compute gradients through it using the reparametrization trick. The fact that all three components are optimized with respect to the same objective makes implementation of RPC surprisingly simple. Note that RPC does not require sampling from $m_\theta(z_{t+1} \mid z_t, a_t)$ and does not require backpropagation through time. In our implementation, we instantiate the encoder $\phi_\theta(s_t)$, the prior $m_\theta(z_{t+1} \mid z_t, a_t)$, and the high-level policy $\pi_\theta(a_t \mid s_t)$ as neural networks with parameters θ. The Q function $Q_\psi(s_t, a_t)$ 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

RPC is closely related to a number of ideas in the RL literature. This section explains these connections, 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 and an analytic expression for the optimal encoder in Appendix A.1.

Model-Based RL. The prior $m_\theta(z_t \mid z_{t-1}, a_{t-1})$ 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 take actions where the model will be more accurate.*

Representation learning. This section explains what the representation z_t represents? Given a prior $m(z_{t+1} \mid z_t, a_t)$ and latent-conditioned policy $\pi^z(a_t \mid z_t)$, we can use the current representation z_t to predict good actions at both the current time step and (by unrolling the prior) at future time steps. Thus, *the representation z_t can also 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 view RPC as *learning a new action space for the MDP*. Precisely, define a new MDP where the action space is $\mathcal{Z}$ and the reward function is $\tilde{r}$ (Eq. 4). The new MDP encodes a strong prior for open-loop policies: simply sampling actions from the prior yields high reward. Our encoder $\phi(z_t \mid s_t)$ samples "actions" z_t from this new MDP. In fact, the updates for the encoder are equivalent to MaxEnt RL with policy $\phi(z_t \mid s_t)$ and reward function $r(s_t, a_t) + \log m(z_t \mid z_{t-1}, a_{t-1})$.

Open-loop control. RPC learns a model $m(z_t \mid z_{t-1}, a_{t-1})$ 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).

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 z_t instead of predicting 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, and will confirm this prediction experimentally in Fig. 4. In practice, the policy learned by RPC will look at every observation, but may only look at a few bits from that observation.

5.2 Theoretical Guarantees

We conclude this section by formally relating model compression to open-loop performance and generalization error. We do not intend this section to be a comprehensive analysis of all benefits deriving from model compression (see, e.g., [4, 5]). All proofs are in Appendix C.

Intuitively, we know that 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.

Lemma 5.1. *Let encoder $\phi(z_t \mid s_t)$, policy $\pi^z(a_t \mid z_t)$, prior $m(z_t \mid z_{t-1}, a_{t-1})$, and reward function $r(s_t, a_t)$. Then applying our model compression objective (Eq. 3) with reward function $(1-\gamma)\log r(s_t, a_t)$ maximizes a lower bound on the expected return of the open-loop policy.*

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

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

Our next result shows that, not only does RPC optimize for open loop performance, but the difference between the performance of the open-loop policy and the reactive policy can be bounded by the policy's bitrate. This result could be useful for quantifying the regret incurred when using the learned representation z_t as an action space of temporally-extended behaviors.

Let π^{open} be the open-loop policy corresponding to the composition of the prior $m(z_{t+1} \mid z_t, a_t)$ and the high-level policy $\pi^z(a_t \mid z_t)$. Let π^{reactive} be the reactive policy corresponding to the composition of the encoder $\phi(z_t \mid s_t)$ and the high-level policy $\pi^z(a_t \mid z_t)$. To simplify notation, we further define the sum of discounted rewards of a trajectory as $R(\tau) \triangleq \sum_t \gamma^t r(s_t, a_t)$, and let $R_{\text{max}} = \max_\tau R(\tau)$ be the maximum return of any trajectory.

Lemma 5.2. *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 $\pi^{reactive}$:*

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

One oft-cited benefit of compression is that the resulting models generalize better from the training set to the testing set. Our final result is to apply 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. Stochastic MDPs implicitly define a distribution of deterministic MDPs [30]. For the following result, we assume that the given stochastic MDP is a mixture of deterministic MDPs that can each be described by a b-bit random string.

Lemma 5.3. *Let stochastic MDP $\mathcal{M}$ and reactive policy $\pi^{reactive}$ be given. Define $R^\pi(M)$ to be the expected reward of policy π on MDP M. Then the probability that the policy's expected return on an observed (deterministic) MDP is much worse 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. [5, Theorem 8]. This result may be of interest in the offline RL setting, where observed trajectories effectively constitute a deterministic MDP. The intuition is that policies that use fewer bits will be less likely to overfit to the offline dataset.

In summary, our theoretical results suggest that compressed policies learn representations that can be used for planning and that may generalize better. 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

	feedforward architecture	predicted prior	augmented reward
RPC (ours)	✓	✓	✓
VIB [18]	✓	✗	✗
VIB+reward [25]	✓	✗	✗
VIB+RNN	✗	✗	✓

Our first experiment studies whether RPC outperforms alternative approaches to compression, which we summarize in the inline table. We compare against a standard VIB [18] and an extension that adds the information cost to the reward, "VIB+reward" [25]. 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 [18] 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 OpenAI-Gym [6] and two *image-based* tasks from dm-control [34]. Because of computational constraints, we omit the RNN+VIB baseline on the image-based tasks.

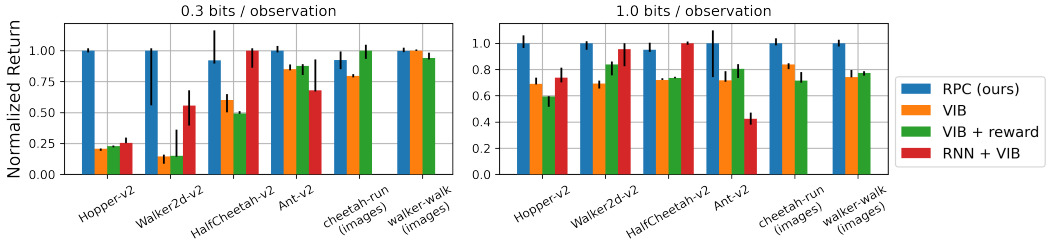
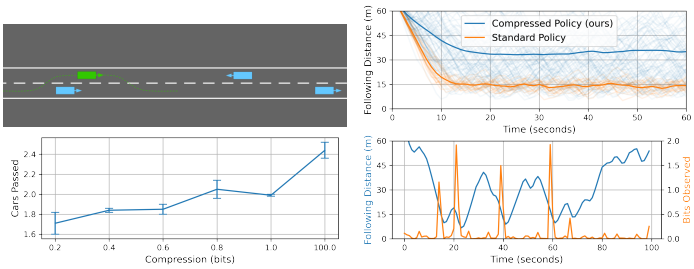


Figure 2: **Learning Compressed Policies.** We measure the return achieved by policies constrained to have a fixed bit rate. Especially at low bit rates, RPC achieves higher return than alternative methods.



(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.

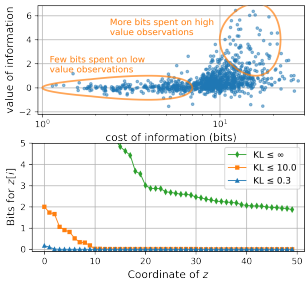


Figure 4: **Representations**: *(Top)* Compressed policies observe more bits from observations that have a large value of information. *(Bottom)* Compressed policies learn sparse representations.

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 [23]. 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. 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.[2]

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 task, we plot

[2]Project site with videos and code: `http://rpc-anonymous.github.io/`

Figure 5: **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.

the value of information versus cost of information for many sampled states (details in Appendix B). As shown in Fig. 4 (top), the RPC uses more bits from observations that are more valuable for maximizing future returns. To visualize the learned representation z_t, we sample the representation from randomly sampled states, and plot the numbers of bits used for each coordinate. of 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.

6.3 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. 5 (top-left). The initial position of the object and the goal position are randomized, so we expect that different representations 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 z_t. See Appendix B for details and pseudocode.

We compare the action representations learned by RPC to three baselines. 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. "Action Repeat" constantly outputs the same action. Finally, "RL from scratch" applies a state-of-the-art off-policy RL algorithm (SAC [14]) to the task. We apply all methods to four tasks and present results in Fig. 5. First, as a sanity check, we apply all method to the training task, finding that RPC quickly finds a single 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 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.

6.4 Robustness

The connection between compression and robustness has been well established in the literature(e.g., [13, 38]). 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 since 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 will actively modify their behavior to adopt strategies that require fewer bits from the observation. In this experiment, we drop each observation in-

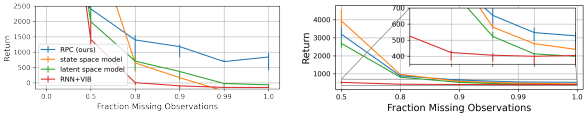


Figure 6: **Robustness to missing observations**: RPC is more robust to missing observations than prior methods, including those that learn dynamics models. We show HalfCheetah-v2 on left and Walker2d-v2 on right.

dependently 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. 6 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, a useful property in real-world environments where sensor measurements may be missing or corrupted.

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 [28, 35], here we investigate whether compression is simple yet effect means for achieving a modicum of robustness; we do not claim that compression is the best method for robust RL.

We first study adversarial perturbations to the **dynamics**. Given a policy $\pi(a \mid s)$ and the current state s, 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 [26]; see Appendix B for details. Fig. 7 (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 the **observation**. Unlike the previous experiment, we let the adversary perturb *every* step in an episode; see Appendix B for full details. Fig. 7 (right) shows that policies that use fewer bits achieve higher returns in this adversarial setting. Policies that use too many bits (3 or more) are flipped over by this adversary after about 500 steps, whereas policies that use fewer bits remain upright.

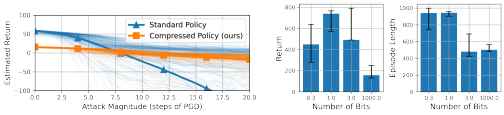


Figure 7: **Adversarial robustness**: Compressed policies are more robust against adversarial attacks to the *(Left)* dynamics and *(Right)* observations.

Robust RL. Our final set of experiments look at higher-level perturbations to the dynamics, as are typically studied in the robust RL community [28, 35]. 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. 8 shows that compressed policies generalize to larger masses and smaller frictions more effectively than an uncompressed policy. Comparing the policies learned 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. 14 for a larger version of this plot with error bars.

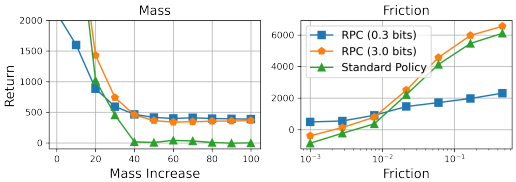


Figure 8: **Robust RL**: Compressed policies are more robust to increases in mass and decreases in friction.

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. 7). We also demonstrate that our method learns representations that are suitable for use in hierarchical RL.

Limitations. The main limitation of this work is that policies that use few bits will often receive lower reward on the training tasks. Second, 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.

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] Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., et al. (2019). Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*.

[3] Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. (2016). Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*.

[4] 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.

[5] 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.

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

[7] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[8] 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.

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

[10] 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(2_and_3):157–165.

[11] 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.

[12] Goyal, A., Islam, R., Strouse, D., Ahmed, Z., Botvinick, M., Larochelle, H., Bengio, Y., and Levine, S. (2019). Infobot: Transfer and exploration via the information bottleneck. *arXiv preprint arXiv:1901.10902*.

[13] Gui, S., Wang, H., Yu, C., Yang, H., Wang, Z., and Liu, J. (2019). Model compression with adversarial robustness: A unified optimization framework. *arXiv preprint arXiv:1902.03538*.

[14] 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.

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

[16] 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.

[17] Huang, S., Papernot, N., Goodfellow, I., Duan, Y., and Abbeel, P. (2017). Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*.

[18] 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. *arXiv preprint arXiv:1910.12911*.

[19] Kamalaruban, P., Huang, Y.-T., Hsieh, Y.-P., Rolland, P., Shi, C., and Cevher, V. (2020). Robust reinforcement learning via adversarial training with langevin dynamics. *arXiv preprint arXiv:2002.06063*.

[20] 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.

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

[22] Lee, A. X., Nagabandi, A., Abbeel, P., and Levine, S. (2019). Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*.

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

[24] Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373.

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

[26] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

[27] Maxwell, J. C. and Pesic, P. (2001). *Theory of heat*. Courier Corporation.

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

[29] Nachum, O., Gu, S., Lee, H., and Levine, S. (2018). Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*.

[30] Ng, A. Y. and Jordan, M. I. (2013). Pegasus: A policy search method for large mdps and pomdps. *arXiv preprint arXiv:1301.3878*.

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

[32] 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.

[33] Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W., and Abbeel, P. (2017). Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*.

[34] 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*.

[35] 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.

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

[37] 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.

[38] 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.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec. 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Limitations in Sec. 7.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]

(b) Did you include complete proofs of all theoretical results? [Yes]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] All experiments were run for approximately one day. The image-based experiments used half of a NVIDIA p100 or v100 GPU. The state-based experiments did not use a GPU.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]