Stochastic Latent Actor-Critic: Deep Reinforcement Learning with a Latent Variable Model

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep reinforcement learning (RL) algorithms can use high-capacity deep networks to learn directly from image observations. However, these high-dimensional observation spaces present a number of challenges in practice, since the policy must now solve two problems: representation learning and task learning. In this work, we tackle these two problems separately, by explicitly learning latent representations that can accelerate reinforcement learning from images. We propose the stochastic latent actor-critic (SLAC) algorithm: a sample-efficient and high-performing RL algorithm for learning policies for complex continuous control tasks directly from high-dimensional image inputs. SLAC provides a novel and principled approach for unifying stochastic sequential models and RL into a single method, by learning a compact latent representation and then performing RL in the model's learned latent space. Our experimental evaluation demonstrates that our method outperforms both model-free and model-based alternatives in terms of final performance and sample efficiency, on a range of difficult image-based control tasks. Our code and videos of our results are available at our anonymous website.[1]

1 Introduction

Deep reinforcement learning (RL) algorithms can learn to solve tasks directly from raw, low-level observations such as images. However, such high-dimensional observation spaces present a number of challenges in practice: On one hand, it is difficult to directly learn from these high-dimensional inputs, but on the other hand, it is also difficult to tease out a compact representation of the underlying task-relevant information from which to learn instead. Standard model-free deep RL aims to unify these challenges of representation learning and task learning into a single end-to-end training procedure. However, solving *both* problems together is difficult, since an effective policy requires an effective representation, and an effective representation requires meaningful gradient information to come from the policy or value function, while using only the model-free supervision signal (i.e., the reward function). As a result, learning directly from images with standard end-to-end RL algorithms can in practice be slow, sensitive to hyperparameters, and inefficient.

Instead, we propose to separate representation learning and task learning, by relying on predictive model learning to explicitly acquire a latent representation, and training the RL agent *in that learned latent space*. This alleviates the representation learning challenge because predictive learning benefits from a rich and informative supervision signal even before the agent has made any progress on the task, and thus results in improved sample efficiency of the overall learning process. In this work, our predictive model serves to accelerate task learning by separately addressing representation learning, in contrast to existing model-based RL approaches, which use predictive models either for generating cheap synthetic experience [49, 21, 32] or for planning into the future [10, 12, 44, 8, 53, 25].

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

[1] https://github.com/rl-slac/slac/

Our proposed stochastic sequential model (Figure 1) models the high-dimensional observations as the consequence of a latent process, with a Gaussian prior and latent dynamics. This model represents a partially observed Markov decision process (POMDP), where the stochastic latent state enables the model to represent uncertainty about any of the state variables, given the past observations. Solving such a POMDP exactly would be computationally intractable, since it amounts to solving the decision problem in the space of *beliefs* [4, 33]. Recent works approximate the belief as encodings of latent samples from forward rollouts or particle filtering [7, 30], or as learned belief representations in a belief-state forward model [20]. We instead propose a simple approximation, which we derive from the control as inference framework, that trains a Markovian critic on latent state samples and trains an actor on a history of observations and actions, resulting in our stochastic latent actor-critic (SLAC) algorithm. Although this approximation loses some of the benefits of full POMDP solvers, it is easy and stable to train in practice, achieving state-of-the-art results on a range of challenging problems.

The main contribution of this work is a novel and principled approach that integrates learning stochastic sequential models and RL into a single method, performing RL in the model's learned latent space. By formalizing the problem as a control as inference problem within a POMDP, we show that variational inference leads to the objective of our SLAC algorithm. We empirically show that SLAC benefits from the good asymptotic performance of model-free RL while also leveraging the improved latent space representation for sample efficiency, by demonstrating that SLAC substantially outperforms *both* model-free and model-based RL algorithms on a range of image-based continuous control benchmark tasks.

2 Related Work

Representation learning in RL. End-to-end deep RL can in principle learn representations implictly as part of the RL process [43]. However, prior work has observed that RL has a "representation learning bottleneck": a considerable portion of the learning period must be spent acquiring good representations of the observation space [48]. This motivates the use of a distinct representation learning procedure to acquire these representations before the agent has even learned to solve the task. A number of prior works have explored the use of auxiliary supervision in RL to learn such representations [40, 13, 31, 28, 22, 45, 46, 18, 9]. In contrast to this class of representation learning algorithms, we explicitly learn a latent variable model of the POMDP, in which the latent representation and latent-space dynamics are jointly learned. By modeling covariances between consecutive latent states, we make it feasible for our proposed algorithm to perform Bellman backups directly in the latent space of the learned model.

Partial observability in RL. Our work is also related to prior research on RL under partial observability. Prior work has studied exact and approximate solutions to POMDPs, but they require explicit models of the POMDP and are only practical for simpler domains [33]. Recent work has proposed end-to-end RL methods that use recurrent neural networks to process histories of observations and (sometimes) actions, but without constructing a model of the POMDP [27, 14, 54]. Other works, however, learn latent-space dynamical system models and then use them to solve the POMDP with model-based RL [52, 51, 35, 53, 25, 36]. Although some of these works learn latent variable models that are similar to ours, these methods are often limited by compounding model errors and finite horizon optimization. In contrast to these works, our approach does not use the model for prediction, and performs infinite horizon policy optimization. Our approach benefits from the good asymptotic performance of model-free RL, while at the same time leveraging the improved latent space representation for sample efficiency.

Other works have also trained latent variable models and used their representations as the inputs to model-free RL algorithms. They use representations encoded from latent states sampled from the forward model [7], belief representations obtained from particle filtering [30], or belief representations obtained directly from a learned belief-space forward model [20]. Our approach is closely related to these prior methods, in that we also use model-free RL with a latent state representation that is learned via prediction. However, instead of using belief representations, our method learns a critic directly on latent state samples, which more tractably enables scaling to more complex tasks. Concurrent to our work, Hafner et al. [26] proposed to integrate model-free learning with representations from sequence models, as proposed in this paper, with model-based rollouts, further improving on the performance of prior model-based approaches.

Sequential latent variable models. Several previous works have explored various modeling choices to learn stochastic sequential models [39, 3, 34, 15, 16, 11, 19]. They vary in the factorization of the generative and inference models, their network architectures, and the objectives used in their training procedures. Our approach is compatible with any of these sequential latent variable models, with the only requirement being that they provide a mechanism to sample latent states from the belief of the learned Markovian latent space.

3 Preliminaries

This work addresses the problem of learning policies from high-dimensional observations in POMDPs, by simultaneously learning a latent representation of the underlying MDP state using variational inference, as well as learning a policy in a maximum entropy RL framework. In this section, we describe maximum entropy RL [55, 23, 41] in fully observable MDPs, as well as variational methods for training latent state space models for POMDPs.

3.1 Maximum Entropy RL in Fully Observable MDPs

Consider a Markov decision process (MDP), with states $\mathbf{s}_t \in \mathcal{S}$, actions $\mathbf{a}_t \in \mathcal{A}$, rewards r_t, initial state distribution $p(\mathbf{s}_1)$, and stochastic transition distribution $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$. Standard RL aims to learn the parameters ϕ of some policy $\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$ such that the expected sum of rewards is maximized under the induced trajectory distribution ρ_π. This objective can be modified to incorporate an *entropy* term, such that the policy also aims to maximize the expected entropy $\mathcal{H}(\pi_\phi(\cdot|\mathbf{s}_t))$. This formulation has a close connection to variational inference [55, 23, 41], and we build on this in our work. The resulting maximum entropy objective is $\sum_{t=1}^{T} \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha\mathcal{H}(\pi_\phi(\cdot|\mathbf{s}_t))]$, where r is the reward function, and α is a temperature parameter that trades off between maximizing for the reward and for the policy entropy. Soft actor-critic (SAC) [23] uses this maximum entropy RL framework to derive soft policy iteration, which alternates between policy evaluation and policy improvement within the described maximum entropy framework. SAC then extends this soft policy iteration to handle continuous action spaces by using parameterized function approximators to represent both the Q-function Q_θ (critic) and the policy π_ϕ (actor). The soft Q-function parameters θ are optimized to minimize the soft Bellman residual,

$$J_Q(\theta) = \frac{1}{2}\left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \left(\; r_t + \gamma \mathop{\mathbb{E}}_{\mathbf{a}_{t+1}\sim\pi_\phi}\left[Q_{\bar\theta}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - \alpha \log \pi_\phi(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})\right] \; \right)\right)^2, \quad (1)$$

where γ is the discount factor, and $\bar\theta$ are delayed parameters. The policy parameters ϕ are optimized to update the policy towards the exponential of the soft Q-function, resulting in the policy loss

$$J_\pi(\phi) = \mathop{\mathbb{E}}_{\mathbf{a}_t\sim\pi_\phi}\left[\alpha \log(\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t)\right]. \quad (2)$$

SLAC builds on top of this maximum entropy RL framework, by further integrating explicit representation learning and handling partial observability.

3.2 Sequential Latent Variable Models and Amortized Variational Inference in POMDPs

To learn representations for RL, we use latent variable models trained with amortized variational inference. The learned model must be able to process a large number of pixels that are present in the entangled image $\mathbf{x}$, and it must tease out the relevant information into a compact and disentangled representation $\mathbf{z}$. To learn such a model, we can consider maximizing the probability of each observed datapoint $\mathbf{x}$ from some training set under the entire generative process $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\,\mathrm{d}\mathbf{z}$. This objective is intractable to compute in general due to the marginalization of the latent variables $\mathbf{z}$. In amortized variational inference, we utilize the evidence lower bound for the log-likelihood [38]:

$$\log p(\mathbf{x}) \geq E_{\mathbf{z}\sim q}\left[\log p(\mathbf{x}|\mathbf{z})\right] - \mathrm{D}_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})). \quad (3)$$

We can maximize the probability of the observed datapoints (i.e., the left hand side of Equation (3)) by learning an encoder $q(\mathbf{z}|\mathbf{x})$ and a decoder $p(\mathbf{x}|\mathbf{z})$, and then directly performing gradient ascent on the right hand side of the equation. In this setup, the distributions of interest are the prior $p(\mathbf{z})$, the observation model $p(\mathbf{x}|\mathbf{z})$, and the variational approximate posterior $q(\mathbf{z}|\mathbf{x})$.

In order to extend such models to sequential decision making settings, we must incorporate actions and impose temporal structure on the latent state. Consider a partially observable MDP (POMDP), with latent states $\mathbf{z}_t \in \mathcal{Z}$ and its corresponding observations $\mathbf{x}_t \in \mathcal{X}$. We make an explicit distinction between an observation $\mathbf{x}_t$ and the underlying latent state $\mathbf{z}_t$, to emphasize that the latter is unobserved and its distribution is unknown. Analogous to the MDP, the initial and transition distributions are $p(\mathbf{z}_1)$ and $p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$, and the reward is r_t. In addition, the observation model is given by $p(\mathbf{x}_t|\mathbf{z}_t)$.

As in the case for VAEs, a generative model of these observations $\mathbf{x}_t$ can be learned by maximizing the log-likelihood. In the POMDP setting, however, we note that $\mathbf{x}_t$ alone does not provide all necessary information to infer $\mathbf{z}_t$, and prior observations must be taken into account during inference. This brings us to the discussion of sequential latent variable models. The distributions of interest are $p(\mathbf{z}_1)$ and $p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$, the observation model $p(\mathbf{x}_t|\mathbf{z}_t)$, and the approximate variational posteriors $q(\mathbf{z}_1|\mathbf{x}_1)$ and $q(\mathbf{z}_{t+1}|\mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t)$. The log-likehood of the observations can then be bounded,

$$\log p(\mathbf{x}_{1:\tau+1}|\mathbf{a}_{1:\tau}) \geq \mathop{\mathbb{E}}_{\mathbf{z}_{1:\tau+1}\sim q}\left[\sum_{t=0}^{\tau} \log p(\mathbf{x}_{t+1}|\mathbf{z}_{t+1}) - \mathrm{D}_{\mathrm{KL}}(q(\mathbf{z}_{t+1}|\mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t) \parallel p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t))\right]. \tag{4}$$

For notational convenience, we define $q(\mathbf{z}_1|\mathbf{x}_1, \mathbf{z}_0, \mathbf{a}_0) := q(\mathbf{z}_1|\mathbf{x}_1)$ and $p(\mathbf{z}_1|\mathbf{z}_0, \mathbf{a}_0) := p(\mathbf{z}_1)$. Prior work [7, 30, 20, 25, 19, 36, 11, 53] has explored modeling such non-Markovian observation sequences, using methods such as recurrent neural networks with deterministic hidden state, as well as probabilistic state-space models. In this work, we enable the effective training of a fully stochastic sequential latent variable model, and bring it together with a maximum entropy actor-critic RL algorithm to create SLAC: a sample-efficient and high-performing RL algorithm for learning policies for complex continuous control tasks directly from high-dimensional image inputs.

4 Joint Modeling and Control as Inference

For a fully observable MDP, the control problem can be embedded into a graphical model by introducing a binary random variable $\mathcal{O}_t$, which indicates if time step t is optimal. When its distribution is chosen to be $p(\mathcal{O}_t = 1|\mathbf{s}_t, \mathbf{a}_t) = \exp(r(\mathbf{s}_t, \mathbf{a}_t))$, then maximization of $p(\mathcal{O}_{1:T})$ via approximate inference in that model yields the optimal policy for the maximum entropy objective [41].

In this paper, we extend this idea to the POMDP setting, where the probabilistic graphical model includes latent variables, as shown in Figure 1, and the distribution can analogously be given by $p(\mathcal{O}_t = 1|\mathbf{z}_t, \mathbf{a}_t) = \exp(r(\mathbf{z}_t, \mathbf{a}_t))$. Instead of maximizing the likelihood of the optimality variables alone, we jointly model the observations (including the observed rewards of the past time steps) and learn maximum entropy policies by maximizing the marginal likelihood $p(\mathbf{x}_{1:\tau+1}, \mathcal{O}_{\tau+1:T}|\mathbf{a}_{1:\tau})$. This objective represents both the likelihood of the observed data from the past $\tau + 1$ steps, as well as the optimality of the agent's actions for future steps, effectively combining both representation learning and control into a single graphical model. We factorize our variational distribution into a product of *recognition* terms $q(\mathbf{z}_{t+1}|\mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t)$, *dynamics* terms $p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$, and *policy* terms $\pi(\mathbf{a}_t|\mathbf{x}_{1:t}, \mathbf{a}_{1:t-1})$:

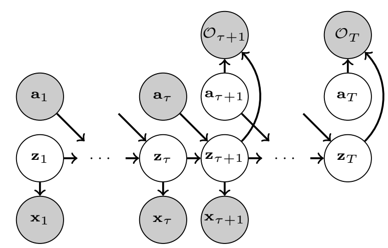


Figure 1: Graphical model of POMDP with optimality variables for $t \geq \tau + 1$.

$$q(\mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T}|\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}) = \prod_{t=0}^{\tau} q(\mathbf{z}_{t+1}|\mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t) \prod_{t=\tau+1}^{T-1} p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t) \prod_{t=\tau+1}^{T} \pi(\mathbf{a}_t|\mathbf{x}_{1:t}, \mathbf{a}_{1:t-1}). \tag{5}$$

The variational distribution uses the dynamics for future time steps to prevent the agent from controlling the transitions and from choosing optimistic actions, analogously to the fully observed MDP setting described by Levine [41]. The posterior over the actions represents the policy π.

We use the posterior from Equation (5) to obtain the evidence lower bound (ELBO) of the likelihood,

$$\begin{aligned}
&\log p(\mathbf{x}_{1:\tau+1}, \mathcal{O}_{\tau+1:T}|\mathbf{a}_{1:\tau}) \\
&\quad \geq \mathop{\mathbb{E}}_{(\mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T})\sim q}\Big[\log p(\mathbf{x}_{1:\tau+1}, \mathcal{O}_{\tau+1:T}, \mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T}|\mathbf{a}_{1:\tau}) - \log q(\mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T}|\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau})\Big] \\
&\quad = \mathop{\mathbb{E}}_{(\mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T})\sim q}\Bigg[\underbrace{\sum_{t=0}^{\tau}\Big(\log p(\mathbf{x}_{t+1}|\mathbf{z}_{t+1}) - \mathrm{D}_{\mathrm{KL}}(q(\mathbf{z}_{t+1}|\mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t) \parallel p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t))\Big)}_{\text{model objective terms}} \\
&\qquad\qquad + \underbrace{\sum_{t=\tau+1}^{T}\Big(r(\mathbf{z}_t, \mathbf{a}_t) + \log p(\mathbf{a}_t) - \log \pi(\mathbf{a}_t|\mathbf{x}_{1:t}, \mathbf{a}_{1:t-1})\Big)}_{\text{policy objective terms}}\Bigg],
\end{aligned} \tag{6}$$

where $r(\mathbf{z}_t, \mathbf{a}_t) = \log p(\mathcal{O}_t = 1|\mathbf{z}_t, \mathbf{a}_t)$ by construction and $p(\mathbf{a}_t)$ is the action prior. The full derivation of the ELBO is given in Appendix A.

5 Stochastic Latent Actor Critic

We now describe our stochastic latent actor critic (SLAC) algorithm, which maximizes the ELBO using function approximators to model the prior and posterior distributions. The ELBO objective in Equation (6) can be split into a model objective and a maximum entropy RL objective. The model objective can be optimized directly, while the maximum entropy RL objective can be optimized via approximate message passing, with messages corresponding to the Q-function. We can rewrite the RL objective to express it in terms of these messages, yielding an actor-critic algorithm analogous to SAC. Additional details of the derivation of the SLAC objectives are given in Appendix A.

Latent variable model. The first part of the ELBO corresponds to training the latent variable model to maximize the likelihood of the observations, analogous to the ELBO in Equation (4) for the sequential latent variable model. The generative model is given by $p_\psi(\mathbf{z}_1)$, $p_\psi(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$, and $p_\psi(\mathbf{x}_t|\mathbf{z}_t)$, and the inference model is given by $q_\psi(\mathbf{z}_1|\mathbf{x}_1)$ and $q_\psi(\mathbf{z}_{t+1}|\mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t)$. These distributions are diagonal Gaussian, where the means and variances are given by outputs of neural networks. Further details of our specific model architecture are given in Appendix B. The distribution parameters ψ are optimized with respect to the ELBO in Equation (6), where the only terms that depend on ψ, and therefore constitute the model objective, are given by

$$J_M(\psi) = \mathop{\mathbb{E}}_{\mathbf{z}_{1:\tau+1} \sim q_\psi}\left[\sum_{t=0}^{\tau} - \log p_\psi(\mathbf{x}_{t+1}|\mathbf{z}_{t+1}) + \mathrm{D}_{\mathrm{KL}}(q_\psi(\mathbf{z}_{t+1}|\mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t) \| p_\psi(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t))\right], \quad (7)$$

where we define $q_\psi(\mathbf{z}_1|\mathbf{x}_1, \mathbf{z}_0, \mathbf{a}_0) \coloneqq q_\psi(\mathbf{z}_1|\mathbf{x}_1)$ and $p_\psi(\mathbf{z}_1|\mathbf{z}_0, \mathbf{a}_0) \coloneqq p_\psi(\mathbf{z}_1)$. We use the reparameterization trick to sample from the filtering distribution $q_\psi(\mathbf{z}_{1:\tau+1}|\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau})$.

Actor and critic. The second part of the ELBO corresponds to the maximum entropy RL objective. As in the fully observable case from Section 3.1 and as described by Levine [41], this optimization can be solved via message passing of soft Q-values. However, in our method, we must use the latent states $\mathbf{z}$, since the true state is unknown. The messages are approximated by minimizing the soft Bellman residual, which we use to train our soft Q-function parameters θ,

$$J_Q(\theta) = \mathop{\mathbb{E}}_{\mathbf{z}_{1:\tau+1} \sim q_\psi}\left[\tfrac{1}{2}\left(Q_\theta(\mathbf{z}_\tau, \mathbf{a}_\tau) - (r_\tau + \gamma V_{\bar{\theta}}(\mathbf{z}_{\tau+1}))\right)^2\right], \quad (8)$$

$$V_\theta(\mathbf{z}_{\tau+1}) = \mathop{\mathbb{E}}_{\mathbf{a}_{\tau+1} \sim \pi_\phi}\left[Q_\theta(\mathbf{z}_{\tau+1}, \mathbf{a}_{\tau+1}) - \alpha \log \pi_\phi(\mathbf{a}_{\tau+1}|\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau})\right], \quad (9)$$

where V_θ is the soft state value function and $\bar{\theta}$ are delayed target network parameters, obtained as exponential moving averages of θ. Notice that the latents $\mathbf{z}_\tau$ and $\mathbf{z}_{\tau+1}$, which are used in the Bellman backup, are sampled from the same filtering distribution, i.e. $\mathbf{z}_{\tau+1} \sim q_\psi(\mathbf{z}_{\tau+1}|\mathbf{x}_{\tau+1}, \mathbf{z}_\tau, \mathbf{a}_\tau)$. The RL objective, which corresponds to the second part of the ELBO, can then be rewritten in terms of the soft Q-function. The policy parameters ϕ are optimized to maximize this objective, resulting in a policy loss analogous to soft actor-critic [23]:

$$J_\pi(\phi) = \mathop{\mathbb{E}}_{\mathbf{z}_{1:\tau+1} \sim q_\psi}\left[\mathop{\mathbb{E}}_{\mathbf{a}_{\tau+1} \sim \pi_\phi}\left[\alpha \log \pi_\phi(\mathbf{a}_{\tau+1}|\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}) - Q_\theta(\mathbf{z}_{\tau+1}, \mathbf{a}_{\tau+1})\right]\right]. \quad (10)$$

We assume a uniform action prior, so $\log p(\mathbf{a}_t)$ is a constant term that we omit from the policy loss. This loss only uses the last sample $\mathbf{z}_{\tau+1}$ of the sequence for the critic, and we use the reparameterization trick to sample from the policy. Note that the policy is not conditioned on the latent state, as this can lead to over-optimistic behavior since the algorithm would learn Q-values for policies that have perfect access to the latent state. Instead, the learned policy in our algorithm is conditioned directly on the past observations and actions. This has the additional benefit that the learned policy can be executed at run time without requiring inference of the latent state. Finally, we note that for the expectation over latent states in the Bellman residual in Equation (9), rather than sampling latent states for all $\mathbf{z} \sim \mathcal{Z}$, we sample latent states from the filtering distribution $q_\psi(\mathbf{z}_{1:\tau+1}|\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau})$. This design choice allows us to minimize the critic loss for samples that are most relevant for Q_θ, while also allowing the critic loss to use the Q-function in the same way as implied by the policy loss in Equation (10).

Algorithm 1 Stochastic Latent Actor-Critic (SLAC)

Require: Environment E and initial parameters $\psi, \phi, \theta_1, \theta_2$ for the model, actor, and critics.
 $\mathbf{x}_1 \sim E_{\text{reset}}()$
 $\mathcal{D} \leftarrow (\mathbf{x}_1)$
 for each iteration **do**
  for each environment step **do**
   $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{x}_{1:t}, \mathbf{a}_{1:t-1})$
   $r_t, \mathbf{x}_{t+1} \sim E_{\text{step}}(\mathbf{a}_t)$
   $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{a}_t, r_t, \mathbf{x}_{t+1})$
  for each gradient step **do**
   $\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}, r_\tau \sim \mathcal{D}$
   $\mathbf{z}_{1:\tau+1} \sim q_\psi(\mathbf{z}_{1:\tau+1}|\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau})$
   $\psi \leftarrow \psi - \lambda_M \nabla_\psi J_M(\psi)$
   $\theta_i \leftarrow \theta_i - \lambda_Q \nabla_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
   $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi J_\pi(\phi)$
   $\bar{\theta}_i \leftarrow \nu\theta_i + (1-\nu)\bar{\theta}_i$ for $i \in \{1, 2\}$

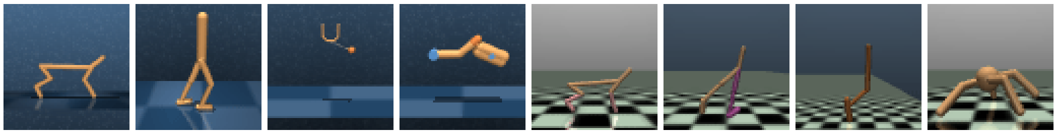


Figure 2: Example image observations for our continuous control benchmark tasks: DeepMind Control's cheetah run, walker walk, ball-in-cup catch, and finger spin, and OpenAI Gym's half cheetah, walker, hopper, and ant (left to right). These images, which are rendered at a resolution of 64×64 pixels, are the observation inputs to our algorithm, i.e. to the latent variable model and to the policy.

SLAC is outlined in Algorithm 1. The actor-critic component follows prior work, with automatic tuning of the temperature α and two Q-functions to mitigate overestimation [17, 23, 24]. SLAC can be viewed as a variant of SAC [23] where the critic is trained on the stochastic latent state of our sequential latent variable model. The backup for the critic is performed on a tuple $(\mathbf{z}_\tau, \mathbf{a}_\tau, r_\tau, \mathbf{z}_{\tau+1})$, sampled from the filtering distribution $q_\psi(\mathbf{z}_{\tau+1}, \mathbf{z}_\tau | \mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau})$. The critic can, in principle, take advantage of the perfect knowledge of the state $\mathbf{z}_t$, which makes learning easier. However, the policy does not have access to $\mathbf{z}_t$, and must make decisions based on a history of observations and actions. SLAC is not a model-based algorithm, in that in does not use the model for prediction, but we see in our experiments that SLAC can achieve similar sample efficiency as a model-based algorithm.

6 Experimental Evaluation

We evaluate SLAC on multiple image-based continuous control tasks from both the DeepMind Control Suite [50] and OpenAI Gym [6], as illustrated in Figure 2. Full details of SLAC's network architecture are described in Appendix B. Aside from the value of action repeats (i.e., control frequency) for the tasks, we kept all of SLAC's hyperparameters constant across all tasks in all domains. Training and evaluation details are given in Appendix C, and image samples from our model for all tasks are shown in Appendix E. Additionally, visualizations of our results and code are available on the project website.[2]

6.1 Comparative Evaluation on Continuous Control Benchmark Tasks

To provide a comparative evaluation against prior methods, we evaluate SLAC on four tasks (cheetah run, walker walk, ball-in-cup catch, finger spin) from the DeepMind Control Suite [50], and four tasks (cheetah, walker, ant, hopper) from OpenAI Gym [6]. Note that the Gym tasks are typically used with low-dimensional state observations, while we evaluate on them with raw image observations. We compare our method to the following state-of-the-art model-based and model-free algorithms:

SAC [23]: This is an off-policy actor-critic algorithm, which represents a comparison to state-of-the-art model-free learning. We include experiments showing the performance of SAC based on true state (as an upper bound on performance) as well as directly from raw images.

D4PG [5]: This is also an off-policy actor-critic algorithm, learning directly from raw images. The results reported in the plots below are the performance after 10^8 training steps, as stated in the benchmarks from Tassa et al. [50].

MPO [2, 1]: This is an off-policy actor-critic algorithm that performs an expectation maximization form of policy iteration, learning directly from raw images.

PlaNet [25]: This is a model-based RL method for learning from images, which uses a partially stochastic sequential latent variable model, but without explicit policy learning. Instead, the model is used for planning with model predictive control (MPC), where each plan is optimized with the cross entropy method (CEM).

DVRL [30]: This is an on-policy model-free RL algorithm that also trains a partially stochastic latent-variable POMDP model. DVRL uses the *full belief* over the latent state as input into both the actor and critic, as opposed to our method, which trains the critic with the latent state and the actor with a history of actions and observations.

Our experiments on the DeepMind Control Suite in Figure 3 show that the sample efficiency of SLAC is comparable or better than *both* model-based and model-free alternatives. This indicates that overcoming the representation learning bottleneck, coupled with efficient off-policy RL, provides

[2] `https://github.com/rl-slac/slac/`

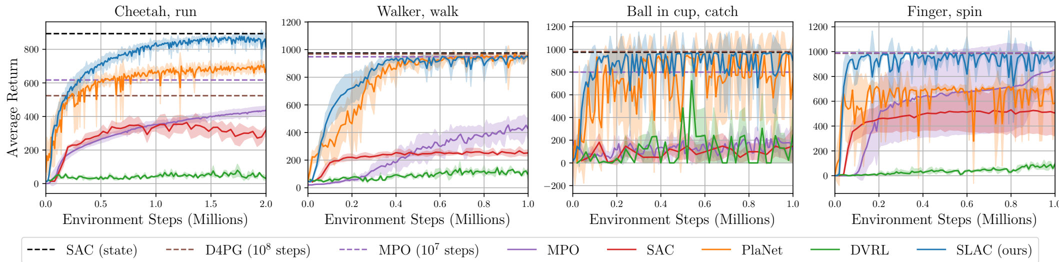


Figure 3: Experiments on the DeepMind Control Suite from images (unless otherwise labeled as "state"). SLAC (ours) converges to similar or better final performance than the other methods, while almost always achieving reward as high as the upper bound SAC baseline that learns from true state. Note that for these experiments, 1000 environments steps corresponds to 1 episode.

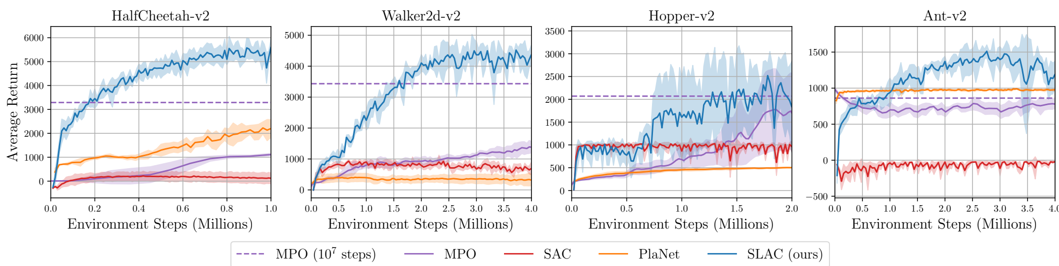


Figure 4: Experiments on the OpenAI Gym benchmark tasks from images. SLAC (ours) converges to higher performance than both PlaNet and SAC on all four of these tasks. The number of environments steps in each episode is variable, depending on the termination.

for fast learning similar to model-based methods, while attaining final performance comparable to fully model-free techniques that learn from state. SLAC also substantially outperforms DVRL. This difference can be explained in part by the use of an efficient off-policy RL algorithm, which can better take advantage of the learned representation.

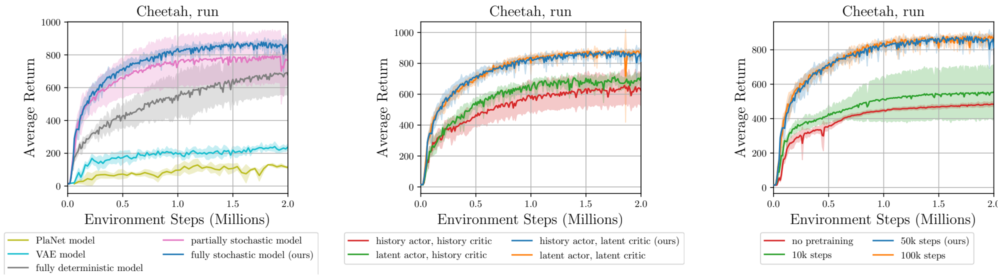
We also evaluate SLAC on continuous control benchmark tasks from OpenAI Gym in Figure 4. We notice that these tasks are more challenging than the DeepMind Control Suite tasks, because the rewards are not as shaped and not bounded between 0 and 1, the dynamics are different, and the episodes terminate on failure (e.g., when the hopper or walker falls over). PlaNet is unable to solve the last three tasks, while for the cheetah task, it learns a suboptimal policy that involves flipping the cheetah over and pushing forward while on its back. To better understand the performance of fixed-horizon MPC on these tasks, we also evaluated with the ground truth dynamics (i.e., the true simulator), and found that even in this case, MPC did not achieve good final performance, suggesting that infinite horizon policy optimization, of the sort performed by SLAC and model-free algorithms, is important to attain good results on these tasks.

Our experiments show that SLAC successfully learns complex continuous control benchmark tasks from raw image inputs. On the DeepMind Control Suite, SLAC exceeds the performance of PlaNet on three of the tasks, and matches its performance on the walker task. However, on the harder image-based OpenAI Gym tasks, SLAC outperforms PlaNet by a large margin. In both domains, SLAC substantially outperforms all prior model-free methods. We note that the prior methods that we tested generally performed poorly on the image-based OpenAI Gym tasks, despite considerable hyperparameter tuning.

6.2 Ablation Experiments

We investigate how SLAC is affected by the choice of latent variable model, the inputs given to the actor and critic, and the model pretraining. Additional results are given in Appendix D.

Latent variable model. We study the tradeoffs between different design choices for the latent variable model in Figure 5a. We compare our *fully stochastic* model to a standard non-sequential *VAE* model [38], which has been used in multiple prior works for representation learning in RL [28, 22, 45], the partially stochastic model used by *PlaNet* [25], as well as two variants of our model: a *fully deterministic* model that removes all stochasticity from the hidden state dynamics, and a *partially*



(a) Latent variable model **(b)** Actor and critic inputs **(c)** Model pretraining

Figure 5: Comparison of different design choices for (a) the latent variable model, (b) the inputs given to the actor and critic, either the history of past observations and actions, or a latent sample, and (c) the number of model pretraining steps. In all cases, we use the RL framework of SLAC.

stochastic model that adds deterministic paths in the transitions, similar to the PlaNet model, but with our latent factorization and architecture. The VAE, fully deterministic, and partially stochastic models use the same architecture as our fully stochastic model, with minimal differences in the transitions. In all cases, we use the RL framework of SLAC and only vary the choice of model for representation learning. Our fully stochastic model outperforms prior models as well as the deterministic and partially stochastic variants of our own model. Contrary to the conclusions in prior work [25, 7], the fully stochastic model slightly outperforms the partially stochastic model, while retaining the appealing interpretation of a stochastic state space model. We hypothesize that these prior works benefit from the deterministic paths (realized as an LSTM or GRU) because they use multi-step samples from the prior. In contrast, our method uses samples from the posterior, which are conditioned on same-step observations, and thus it is less sensitive to the propagation of the latent states through time. We found similar, but less pronounced results for the other tasks, which are shown in Figure 8.

Actor and critic inputs. We next investigate alternative choices for the actor and critic inputs as either the observation-action history or the latent sample. In SLAC, the actor is conditioned on the observation-action history and the critic is conditioned on invididual latent samples. We note that the images in the history are first compressed with the model's convolutional network before they are given to the networks. However, the actor and critic losses do not propagate any gradient signal into the model nor its convolutional layers, i.e. the convolutional layers used for the observation-action history are only trained by the model loss.

Figure 5b shows that the performance is significantly worse when the critic input is the history instead of the latent sample, and indifferent to the choice for the actor input. This is consistent with our derivation—the critic should be given latent samples, but the actor can be conditioned on anything (since the policy is the variational posterior). We note that a latent-conditioned actor could lead to overly-optimistic behaviors in uncertain environments. For generality, we choose to give the raw history directly to the actor. The performance is the lowest when both the actor and the critic are conditioned on the history, even though the convolutional layers are trained with the rich supervision from the model. We show results for other tasks in Figure 9, where we see similar results.

Model pretraining. We next study the effect of pretraining the model before the agent starts learning on the task. In our experiments, the agent first collects a small amount of data by executing uniformly random actions, and then the model is pretrained with that data. The model is pretrained for 50000 iterations, unless otherwise specified. Figure 5c shows that little or no pretraining results in slower learning and worse asymptotic performance, whereas there is no difference when using 100000 instead of 50000 iterations. We found similar, but less pronounced results for the other tasks, which are shown in Figure 10. This shows that the agent benefits from the supervision signal of the model even before the agent has made any progress on the task.

7 Conclusion

We presented SLAC, an efficient RL algorithm for learning from high-dimensional image inputs that combines efficient off-policy model-free RL with representation learning via a sequential stochastic state space model. Through representation learning in conjunction with effective task learning in the learned latent space, our method achieves improved sample efficiency and final task performance as compared to *both* prior model-based and model-free RL methods.

Broader Impact

Despite the existence of automated robotic systems in controlled environments such as factories or labs, standard approaches to controlling systems still require precise and expensive sensor setups to monitor the relevant details of interest in the environment, such as the joint positions of a robot or pose information of all objects in the area. To instead be able to learn directly from the more ubiquitous and rich modality of vision would greatly advance the current state of our learning systems. Not only would this ability to learn directly from images preclude expensive real-world setups, but it would also remove the expensive need for human-engineering efforts in state estimation. While it would indeed be very beneficial for our learning systems to be able to learn directly from raw image observations, this introduces algorithm challenges of dealing with high-dimensional as well as partially observable inputs. In this paper, we study the use of explicitly learning latent representations to assist model-free reinforcement learning directly from raw, high-dimensional images.

Traditional end-to-end RL methods try to solve both representation learning and task learning together, and in practice, this leads to brittle solutions which are sensitive to hyperparameters but are also slow and inefficient. These challenges illustrate the predominant use of simulation in the deep RL community; we hope that with more efficient, stable, easy-to-use, and easy-to-train deep RL algorithms such as the one we propose in this work, we can help the field of deep RL to transition to more widespread use in real-world setups such as robotics.

From a broader perspective, there are numerous use cases and areas of application where autonomous decision making agents can have positive effects in our society, from automating dangerous and undersirable tasks, to accelerating automation and economic efficiency of society. That being said, however, automated decision making systems do introduce safety concerns, further exacerbated by the lack of explainability when they do make mistakes. Although this work does not explicitly address safety concerns, we feel that it can be used in conjunction with levels of safety controllers to minimze negative impacts, while drawing on its powerful deep reinforcement learning roots to enable automated and robust tasks in the real world.

References

[1] A. Abdolmaleki, J. T. Springenberg, J. Degrave, S. Bohez, Y. Tassa, D. Belov, N. Heess, and M. A. Riedmiller. Relative entropy regularized policy iteration. *arXiv preprint arXiv:1812.02256*, 2018.

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

[3] E. Archer, I. M. Park, L. Buesing, J. Cunningham, and L. Paninski. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

[4] K. J. Astrom. Optimal control of markov processes with incomplete state information. *Journal of mathematical analysis and applications*, 1965.

[5] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, A. Muldal, N. Heess, and T. Lillicrap. Distributed distributional deterministic policy gradients. In *International Conference on Learning Representations (ICLR)*, 2018.

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

[7] L. Buesing, T. Weber, S. Racanière, S. M. A. Eslami, D. J. Rezende, D. P. Reichert, F. Viola, F. Besse, K. Gregor, D. Hassabis, and D. Wierstra. Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*, 2018.

[8] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Neural Information Processing Systems (NeurIPS)*, 2018.

[9] R. Dadashi, A. A. Taïga, N. L. Roux, D. Schuurmans, and M. G. Bellemare. The value function polytope in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2019.

[10] M. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning (ICML)*, 2011.

[11] A. Doerr, C. Daniel, M. Schiegg, D. Nguyen-Tuong, S. Schaal, M. Toussaint, and S. Trimpe. Probabilistic recurrent state-space models. In *International Conference on Machine Learning (ICML)*, 2018.

[12] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *International Conference on Robotics and Automation (ICRA)*, 2017.

[13] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning. In *International Conference on Robotics and Automation (ICRA)*, 2016.

[14] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Neural Information Processing Systems (NIPS)*, 2016.

[15] M. Fraccaro, S. K. Sonderby, U. Paquet, and O. Winther. Sequential neural models with stochastic layers. In *Neural Information Processing Systems (NIPS)*, 2016.

[16] M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Neural Information Processing Systems (NIPS)*, 2017.

[17] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, 2018.

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

[19] K. Gregor, G. Papamakarios, F. Besse, L. Buesing, and T. Weber. Temporal difference variational auto-encoder. In *International Conference on Learning Representations (ICLR)*, 2019.

[20] K. Gregor, D. J. Rezende, F. Besse, Y. Wu, H. Merzic, and A. v. d. Oord. Shaping belief states with generative environment models for rl. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[21] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning (ICML)*, 2016.

[22] D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

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

[24] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[25] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning (ICML)*, 2019.

[26] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020.

[27] M. Hausknecht and P. Stone. Deep recurrent Q-learning for partially observable MDPs. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, 2015.

[28] I. Higgins, A. Pal, A. Rusu, L. Matthey, C. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner. DARLA: Improving zero-shot transfer in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

[29] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[30] M. Igl, L. Zintgraf, T. A. Le, F. Wood, and S. Whiteson. Deep variational reinforcement learning for POMDPs. In *International Conference on Machine Learning (ICML)*, 2018.

[31] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations (ICLR)*, 2017.

[32] M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[33] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[34] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *International Conference on Learning Representations (ICLR)*, 2016.

[35] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *International Conference on Learning Representations (ICLR)*, 2017.

[36] T. Kim, S. Ahn, and Y. Bengio. Variational temporal abstraction. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[37] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[38] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[39] R. G. Krishnan, U. Shalit, and D. Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

[40] S. Lange and M. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *International Joint Conference on Neural Networks (IJCNN)*, 2010.

[41] S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[42] L. Maaloe, M. Fraccaro, V. Liévin, and O. Winther. Biva: A very deep hierarchy of latent variables for generative modeling. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[43] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing Atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*, 2013.

[44] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *International Conference on Robotics and Automation (ICRA)*, 2018.

[45] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine. Visual reinforcement learning with imagined goals. In *Neural Information Processing Systems (NeurIPS)*, 2018.

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

[47] A. Razavi, A. v. d. Oord, and O. Vinyals. Generating diverse high-fidelity images with vq-vae-2. 2019.

[48] E. Shelhamer, P. Mahmoudieh, M. Argus, and T. Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307*, 2016.

[49] R. S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, 1991.

[50] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. Lillicrap, and M. Riedmiller. DeepMind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[51] N. Wahlström, T. B. Schön, and M. P. Deisenroth. From pixels to torques: Policy learning with deep dynamical models. *arXiv preprint arXiv:1502.02251*, 2015.

[52] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Neural Information Processing Systems (NIPS)*, 2015.

[53] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. J. Johnson, and S. Levine. SOLAR: Deep structured latent representations for model-based reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2019.

[54] P. Zhu, X. Li, P. Poupart, and G. Miao. On improving deep reinforcement learning for POMDPs. *arXiv preprint arXiv:1804.06309*, 2018.

[55] B. D. Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010.

A Derivation of the Evidence Lower Bound and SLAC Objectives

In this appendix, we discuss how the SLAC objectives can be derived from applying a variational inference scheme to the control as inference framework for reinforcement learning [41]. In this framework, the problem of finding the optimal policy is cast as an inference problem, conditioned on the evidence that the agent is behaving optimally. While Levine [41] derives this in the fully observed case, we present a derivation in the POMDP setting. For reference, we reproduce the probabilistic graphical model in Figure 6.

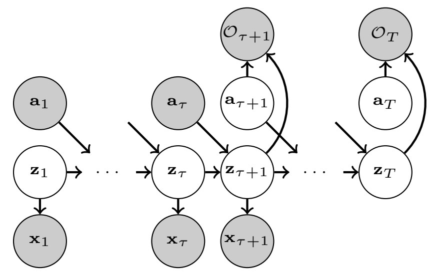


Figure 6: Graphical model of POMDP with optimality variables for $t \geq \tau + 1$.

We aim to maximize the marginal likelihood $p(\mathbf{x}_{1:\tau+1}, \mathcal{O}_{\tau+1:T} | \mathbf{a}_{1:\tau})$, where τ is the number of steps that the agent has already taken. This likelihood reflects that the agent cannot modify the past τ actions and they might have not been optimal, but it can choose the future actions up to the end of the episode, such that the chosen future actions are optimal. Notice that unlike the standard control as inference framework, in this work we not only maximize the likelihood of the optimality variables but also the likelihood of the observations, which provides additional supervision for the latent representation. This does not come up in the MDP setting since the state representation is fixed and learning a dynamics model of the state would not change the model-free equations derived from the maximum entropy RL objective.

For reference, we restate the factorization of our variational distribution:

$$q(\mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T} | \mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}) = \prod_{t=0}^{\tau} q(\mathbf{z}_{t+1} | \mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t) \prod_{t=\tau+1}^{T-1} p(\mathbf{z}_{t+1} | \mathbf{z}_t, \mathbf{a}_t) \prod_{t=\tau+1}^{T} \pi(\mathbf{a}_t | \mathbf{x}_{1:t}, \mathbf{a}_{1:t-1}). \tag{11}$$

As discussed by Levine [41], the agent does not have control over the stochastic dynamics, so we use the dynamics $p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$ for $t \geq \tau + 1$ in the variational distribution in order to prevent the agent from choosing optimistic actions.

The joint likelihood is

$$p(\mathbf{x}_{1:\tau+1}, \mathcal{O}_{\tau+1:T}, \mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T} | \mathbf{a}_{1:\tau}) = \prod_{t=1}^{\tau+1} p(\mathbf{x}_t | \mathbf{z}_t) \prod_{t=0}^{T-1} p(\mathbf{z}_{t+1} | \mathbf{z}_t, \mathbf{a}_t) \prod_{t=\tau+1}^{T} p(\mathcal{O}_t | \mathbf{z}_t, \mathbf{a}_t) \prod_{t=\tau+1}^{T} p(\mathbf{a}_t). \tag{12}$$

We use the posterior from Equation (11), the likelihood from Equation (12), and Jensen's inequality to obtain the ELBO of the marginal likelihood,

$$\begin{aligned}
&\log p(\mathbf{x}_{1:\tau+1}, \mathcal{O}_{\tau+1:T} | \mathbf{a}_{1:\tau}) \\
&\quad = \log \int_{\mathbf{z}_{1:T}} \int_{\mathbf{a}_{\tau+1:T}} p(\mathbf{x}_{1:\tau+1}, \mathcal{O}_{\tau+1:T}, \mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T} | \mathbf{a}_{1:\tau}) \, \mathrm{d}\mathbf{z}_{1:T} \, \mathrm{d}\mathbf{a}_{\tau+1:T} && (13) \\
&\quad \geq \mathop{\mathbb{E}}_{(\mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T}) \sim q} \Big[\log p(\mathbf{x}_{1:\tau+1}, \mathcal{O}_{\tau+1:T}, \mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T} | \mathbf{a}_{1:\tau}) - \log q(\mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T} | \mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}) \Big] && (14) \\
&\quad = \mathop{\mathbb{E}}_{(\mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T}) \sim q} \Bigg[\underbrace{\sum_{t=0}^{\tau} \Big(\log p(\mathbf{x}_{t+1} | \mathbf{z}_{t+1}) - \mathrm{D}_{\mathrm{KL}}(q(\mathbf{z}_{t+1} | \mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t) \parallel p(\mathbf{z}_{t+1} | \mathbf{z}_t, \mathbf{a}_t)) \Big)}_{\text{model objective terms}} \\
&\qquad\qquad + \underbrace{\sum_{t=\tau+1}^{T} \Big(r(\mathbf{z}_t, \mathbf{a}_t) + \log p(\mathbf{a}_t) - \log \pi(\mathbf{a}_t | \mathbf{x}_{1:t}, \mathbf{a}_{1:t-1}) \Big)}_{\text{policy objective terms}} \Bigg], && (15)
\end{aligned}$$

We are interested in the likelihood of optimal trajectories, so we use $\mathcal{O}_t = 1$ for $t \geq \tau + 1$, and its distribution is given by $p(\mathcal{O}_t = 1 | \mathbf{z}_t, \mathbf{a}_t) = \exp(r(\mathbf{z}_t, \mathbf{a}_t))$ in the control as inference framework.

Notice that the dynamics terms $\log p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$ for $t \geq \tau + 1$ from the posterior and the prior cancel each other out in the ELBO.

The first part of the ELBO corresponds to the model objective. When using the parametric function approximators, the negative of it corresponds directly to the model loss in Equation (7).

The second part of the ELBO corresponds to the maximum entropy RL objective. We assume a uniform action prior, so the $\log p(\mathbf{a}_t)$ term is a constant term that can be omitted when optimizing this objective. We use message passing to optimize this objective, with messages defined as

$$Q(\mathbf{z}_t, \mathbf{a}_t) = r(\mathbf{z}_t, \mathbf{a}_t) + \mathop{\mathbb{E}}_{\mathbf{z}_{t+1} \sim q(\cdot|\mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t)} \Big[V(\mathbf{z}_{t+1}) \Big] \tag{16}$$

$$V(\mathbf{z}_t) = \log \int_{\mathbf{a}_t} \exp(Q(\mathbf{z}_t, \mathbf{a}_t)) \, \mathrm{d}\mathbf{a}_t. \tag{17}$$

Then, the maximum entropy RL objective can be expressed in terms of the messages as

$$\begin{aligned}
&\mathop{\mathbb{E}}_{(\mathbf{z}_{\tau+1:T}, \mathbf{a}_{\tau+1:T}) \sim q} \left[\sum_{t=\tau+1}^{T} \Big(r(\mathbf{z}_t, \mathbf{a}_t) - \log \pi(\mathbf{a}_t|\mathbf{x}_{1:t}, \mathbf{a}_{1:t-1}) \Big) \right] \\
&\quad = \mathop{\mathbb{E}}_{\mathbf{z}_{\tau+1} \sim q(\cdot|\mathbf{x}_{\tau+1}, \mathbf{z}_\tau, \mathbf{a}_\tau)} \left[\mathop{\mathbb{E}}_{\mathbf{a}_{\tau+1} \sim \pi(\cdot|\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau})} \Big[Q(\mathbf{z}_{\tau+1}, \mathbf{a}_{\tau+1}) - \log \pi(\mathbf{a}_{\tau+1}|\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}) \Big] \right] && (18) \\
&\quad = \mathop{\mathbb{E}}_{\mathbf{a}_{\tau+1} \sim \pi(\cdot|\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau})} \left[\mathop{\mathbb{E}}_{\mathbf{z}_{\tau+1} \sim q(\cdot|\mathbf{x}_{\tau+1}, \mathbf{z}_\tau, \mathbf{a}_\tau)} \Big[Q(\mathbf{z}_{\tau+1}, \mathbf{a}_{\tau+1}) \Big] - \log \pi(\mathbf{a}_{\tau+1}|\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}) \right] && (19) \\
&\quad = -\mathrm{D}_{\mathrm{KL}} \left(\pi(\mathbf{a}_{\tau+1}|\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}) \,\middle\|\, \frac{\exp\left(\mathbb{E}_{\mathbf{z}_{\tau+1} \sim q}\left[Q(\mathbf{z}_{\tau+1}, \mathbf{a}_{\tau+1})\right]\right)}{\exp\left(\mathbb{E}_{\mathbf{z}_{\tau+1} \sim q}\left[V(\mathbf{z}_{\tau+1})\right]\right)} \right) + \mathop{\mathbb{E}}_{\mathbf{z}_{\tau+1} \sim q} \Big[V(\mathbf{z}_{\tau+1}) \Big], && (20)
\end{aligned}$$

where the first equality is obtained from dynamic programming (see Levine [41] for details), the second equality is obtain by swapping the order of the expectations, the third from the definition of KL divergence, and $\mathbb{E}_{\mathbf{z}_t \sim q}\left[V(\mathbf{z}_t)\right]$ is the normalization factor for $\mathbb{E}_{\mathbf{z}_t \sim q}\left[Q(\mathbf{z}_t, \mathbf{a}_t)\right]$ with respect to $\mathbf{a}_t$. Since the KL divergence term is minimized when its two arguments represent the same distribution, the optimal policy is given by

$$\pi(\mathbf{a}_t|\mathbf{x}_{1:t}, \mathbf{a}_{1:t-1}) = \exp\left(\mathop{\mathbb{E}}_{\mathbf{z}_t \sim q} \Big[Q(\mathbf{z}_t, \mathbf{a}_t) - V(\mathbf{z}_t) \Big] \right). \tag{21}$$

Noting that the KL divergence term is zero for the optimal action, the equality from Equation (18) and Equation (20) can be used in Equation (16) to obtain

$$\begin{aligned}
Q(\mathbf{z}_t, \mathbf{a}_t) = r(\mathbf{z}_t, \mathbf{a}_t) + \mathop{\mathbb{E}}_{\mathbf{z}_{t+1} \sim q(\cdot|\mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t)} \Bigg[\mathop{\mathbb{E}}_{\mathbf{a}_{t+1} \sim \pi(\cdot|\mathbf{x}_{1:t+1}, \mathbf{a}_{1:t})} \Big[Q(\mathbf{z}_{t+1}, \mathbf{a}_{t+1}) \\
- \log \pi(\mathbf{a}_{t+1}|\mathbf{x}_{1:t+1}, \mathbf{a}_{1:t}) \Big] \Bigg].
\end{aligned} \tag{22}$$

This equation corresponds to the Bellman backup with a soft maximization for the value function.

As mentioned in Section 5, our algorithm conditions the parametric policy in the history of observations and actions, which allows us to directly execute the policy without having to perform inference on the latent state at run time. When using the parametric function approximators, the negative of the maximum entropy RL objective, written as in Equation (18), corresponds to the policy loss in Equation (10). Lastly, the Bellman backup of Equation (22) corresponds to the Bellman residual in Equation (9) when approximated by a regression objective.

We showed that the SLAC objectives can be derived from applying variational inference in the control as inference framework in the POMDP setting. This leads to the joint likelihood of the past observations and future optimality variables, which we aim to optimize by maximizing the ELBO of the log-likelihood. We decompose the ELBO into the model objective and the maximum entropy RL objective. We express the latter in terms of messages of Q-functions, which in turn are learned by minimizing the Bellman residual. These objectives lead to the model, policy, and critic losses.

B Latent Variable Factorizationa and Network Architectures

In this section, we describe the architecture of our sequential latent variable model. Motivated by the recent success of autoregressive latent variables in VAEs [47, 42], we factorize the latent variable $\mathbf{z}_t$ into two stochastic variables, $\mathbf{z}_t^1$ and $\mathbf{z}_t^2$, as shown in Figure 7. This factorization results in latent distributions that are more expressive, and it allows for some parts of the prior and posterior distributions to be shared. We found this design to provide a good balance between ease of training and expressivity, producing good reconstructions and generations and, crucially, providing good representations for reinforcement learning. Note that the diagram in Figure 7 represents the *Bayes net* corresponding to our full model. However, since all of the latent variables are stochastic, this visualization also presents the design of the computation graph. Inference over the latent variables is performed using amortized variational inference, with all training done via reparameterization. Hence, the computation graph can be deduced from the diagram by treating all solid arrows as part of the generative model and all dashed arrows as part of approximate posterior.

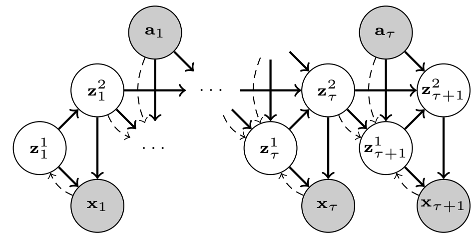


Figure 7: Diagram of our full model. Solid arrows show the generative model, dashed arrows show the inference model. Rewards are not shown for clarity.

The generative model consists of the following probability distributions:

$$\begin{aligned}
\mathbf{z}_1^1 &\sim p(\mathbf{z}_1^1) \\
\mathbf{z}_1^2 &\sim p_\psi(\mathbf{z}_1^2|\mathbf{z}_1^1) \\
\mathbf{z}_{t+1}^1 &\sim p_\psi(\mathbf{z}_{t+1}^1|\mathbf{z}_t^2, \mathbf{a}_t) \\
\mathbf{z}_{t+1}^2 &\sim p_\psi(\mathbf{z}_{t+1}^2|\mathbf{z}_{t+1}^1, \mathbf{z}_t^2, \mathbf{a}_t) \\
\mathbf{x}_t &\sim p_\psi(\mathbf{x}_t|\mathbf{z}_t^1, \mathbf{z}_t^2) \\
r_t &\sim p_\psi(r_t|\mathbf{z}_t^1, \mathbf{z}_t^2, \mathbf{a}_t, \mathbf{z}_{t+1}^1, \mathbf{z}_{t+1}^2).
\end{aligned}$$

The initial distribution $p(\mathbf{z}_1^1)$ is a multivariate standard normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. All of the other distributions are conditional and parameterized by neural networks with parameters ψ. The networks for $p_\psi(\mathbf{z}_1^2|\mathbf{z}_1^1)$, $p_\psi(\mathbf{z}_{t+1}^1|\mathbf{z}_t^2, \mathbf{a}_t)$, $p_\psi(\mathbf{z}_{t+1}^2|\mathbf{z}_{t+1}^1, \mathbf{z}_t^2, \mathbf{a}_t)$, and $p_\psi(r_t|\mathbf{z}_t, \mathbf{a}_t, \mathbf{z}_{t+1})$ consist of two fully connected layers, each with 256 hidden units, and a Gaussian output layer. The Gaussian layer is defined such that it outputs a multivariate normal distribution with diagonal variance, where the mean is the output of a linear layer and the diagonal standard deviation is the output of a fully connected layer with softplus non-linearity. The observation model $p_\psi(\mathbf{x}_t|\mathbf{z}_t)$ consists of 5 transposed convolutional layers (256 4×4, 128 3×3, 64 3×3, 32 3×3, and 3 5×5 filters, respectively, stride 2 each, except for the first layer). The output variance for each image pixel is fixed to 0.1.

The variational distribution q, also referred to as the inference model or the posterior, is represented by the following factorization:

$$\begin{aligned}
\mathbf{z}_1^1 &\sim q_\psi(\mathbf{z}_1^1|\mathbf{x}_1) \\
\mathbf{z}_1^2 &\sim p_\psi(\mathbf{z}_1^2|\mathbf{z}_1^1) \\
\mathbf{z}_{t+1}^1 &\sim q_\psi(\mathbf{z}_{t+1}^1|\mathbf{x}_{t+1}, \mathbf{z}_t^2, \mathbf{a}_t) \\
\mathbf{z}_{t+1}^2 &\sim p_\psi(\mathbf{z}_{t+1}^2|\mathbf{z}_{t+1}^1, \mathbf{z}_t^2, \mathbf{a}_t).
\end{aligned}$$

The networks representing the distributions $q_\psi(\mathbf{z}_1^1|\mathbf{x}_1)$ and $q_\psi(\mathbf{z}_{t+1}^1|\mathbf{x}_{t+1}, \mathbf{z}_t^2, \mathbf{a}_t)$ both consist of 5 convolutional layers (32 5×5, 64 3×3, 128 3×3, 256 3×3, and 256 4×4 filters, respectively, stride 2 each, except for the last layer), 2 fully connected layers (256 units each), and a Gaussian output layer. The parameters of the convolution layers are shared among both distributions.

Note that the variational distribution over $\mathbf{z}_1^2$ and $\mathbf{z}_{t+1}^2$ is intentionally chosen to exactly match the generative model p, such that this term does not appear in the KL-divergence within the ELBO, and a separate variational distribution is only learned over $\mathbf{z}_1^1$ and $\mathbf{z}_{t+1}^1$. In particular, the KL-divergence over $\mathbf{z}_{t+1}$ simplifies to the KL-divergence over $\mathbf{z}_{t+1}^1$:

$$\begin{aligned}
&\mathrm{D}_{\mathrm{KL}}(q(\mathbf{z}_{t+1}|\mathbf{x}_{t+1},\mathbf{z}_t,\mathbf{a}_t)\parallel p(\mathbf{z}_{t+1}|\mathbf{z}_t,\mathbf{a}_t)) && (23)\\
&\quad= \mathop{\mathbb{E}}_{\mathbf{z}_{t+1}\sim q(\cdot|\mathbf{x}_{t+1},\mathbf{z}_t,\mathbf{a}_t)}\Big[\log q(\mathbf{z}_{t+1}|\mathbf{x}_{t+1},\mathbf{z}_t,\mathbf{a}_t)-\log p(\mathbf{z}_{t+1}|\mathbf{z}_t,\mathbf{a}_t)\Big] && (24)\\
&\quad= \mathop{\mathbb{E}}_{\mathbf{z}_{t+1}^1\sim q(\cdot|\mathbf{x}_{t+1},\mathbf{z}_t^2,\mathbf{a}_t)}\Bigg[\mathop{\mathbb{E}}_{\mathbf{z}_{t+1}^2\sim p(\cdot|\mathbf{z}_{t+1}^1,\mathbf{z}_t^2,\mathbf{a}_t)}\Big[\log q(\mathbf{z}_{t+1}^1|\mathbf{x}_{t+1},\mathbf{z}_t^2,\mathbf{a}_t) && (25)\\
&\qquad+\log p(\mathbf{z}_{t+1}^2|\mathbf{z}_{t+1}^1,\mathbf{z}_t^2,\mathbf{a}_t)-\log p(\mathbf{z}_{t+1}^1|\mathbf{z}_t^2,\mathbf{a}_t)-\log p(\mathbf{z}_{t+1}^2|\mathbf{z}_{t+1}^1,\mathbf{z}_t^2,\mathbf{a}_t)\Big]\Bigg]\\
&\quad= \mathop{\mathbb{E}}_{\mathbf{z}_{t+1}^1\sim q(\cdot|\mathbf{x}_{t+1},\mathbf{z}_t^2,\mathbf{a}_t)}\Big[\log q(\mathbf{z}_{t+1}^1|\mathbf{x}_{t+1},\mathbf{z}_t^2,\mathbf{a}_t)-\log p(\mathbf{z}_{t+1}^1|\mathbf{z}_t^2,\mathbf{a}_t)\Big] && (26)\\
&\quad= \mathrm{D}_{\mathrm{KL}}\big(\log q(\mathbf{z}_{t+1}^1|\mathbf{x}_{t+1},\mathbf{z}_t^2,\mathbf{a}_t)\parallel \log p(\mathbf{z}_{t+1}^1|\mathbf{z}_t^2,\mathbf{a}_t)\big)\,. && (27)
\end{aligned}$$

This intentional design decision simplifies the training process.

The latent variables have 32 and 256 dimensions, respectively, i.e. $\mathbf{z}_t^1\in\mathbb{R}^{32}$ and $\mathbf{z}_t^2\in\mathbb{R}^{256}$. For the image observations, $\mathbf{x}_t\in[0,1]^{64\times64\times3}$. All the layers, except for the output layers, use leaky ReLU non-linearities. Note that there are no deterministic recurrent connections in the network—all networks are feedforward, and the temporal dependencies all flow through the stochastic units $\mathbf{z}_t^1$ and $\mathbf{z}_t^2$.

For the reinforcement learning process, we use a critic network Q_θ consisting of 2 fully connected layers (256 units each) and a linear output layer. We use an actor network π_ϕ, which is recurrent for the DeepMind Control tasks and feedforward for the OpenAI Gym tasks. The feedforward actor network consists of 5 convolutional layers, 2 fully connected layers (256 units each), a Gaussian layer, and a tanh bijector, which constrains the actions to be in the bounded action space of $[-1,1]$. The recurrent actor network consists of the same architecture, except that the 2 intermediate fully connected layers are replaced by 2 fully connected layers (400 and 300 units, respectively), an LSTM [29] (40 units), and another fully connected layer (100 units).

C Training and Evaluation Details

As mentioned in Section 6.2, the model is first pretrained for 50000 iterations using a small amount of random data. This data corresponds to 10 episodes for the DeepMind Control Suite and 10000 agent steps for OpenAI Gym. Note that this data is taken into account in our plots. The control portion of our algorithm uses the same hyperparameters as SAC [23], except for a smaller replay buffer size of 100000 environment steps (instead of a million) due to the high memory usage of image observations. All of the parameters are trained with the Adam optimizer [37], and we perform one gradient step per environment step. The Q-function and policy parameters are trained with a learning rate of 0.0003 and a batch size of 256. The model parameters are trained with a learning rate of 0.0001 and a batch size of 32. We use fixed-length sequences of length 8, rather than all the past observations and actions within the episode.

We use action repeats for all the methods, except for D4PG for which we use the reported results from prior work [50]. The number of environment steps reported in our plots correspond to the unmodified steps of the benchmarks. Note that the methods that use action repeats only use a fraction of the environment steps reported in our plots. For example, 1 million environment steps of the cheetah task correspond to 250000 samples when using an action repeat of 4. The action repeats used in our experiments are given in Table 1.

Unlike in prior work [23, 24], we use the same stochastic policy as both the behavioral and evaluation policy since we found the deterministic greedy policy to be comparable or worse than the stochastic policy.

Our plots show results over 5 trials (i.e. seeds), and each trial computes average returns from 10 evaluation episodes.

Benchmark	Task	Action repeat	Original control time step (s)	Effective control time step (s)
DeepMind Control Suite	cheetah run	4	0.01	0.04
	walker walk	2	0.025	0.05
	ball-in-cup catch	4	0.02	0.08
	finger spin	1	0.02	0.02
OpenAI Gym	HalfCheetah-v2	1	0.05	0.05
	Walker2d-v2	4	0.008	0.032
	Hopper-v2	2	0.008	0.016
	Ant-v2	4	0.05	0.2

Table 1: Action repeats and the corresponding agent's control time step used in our experiments.

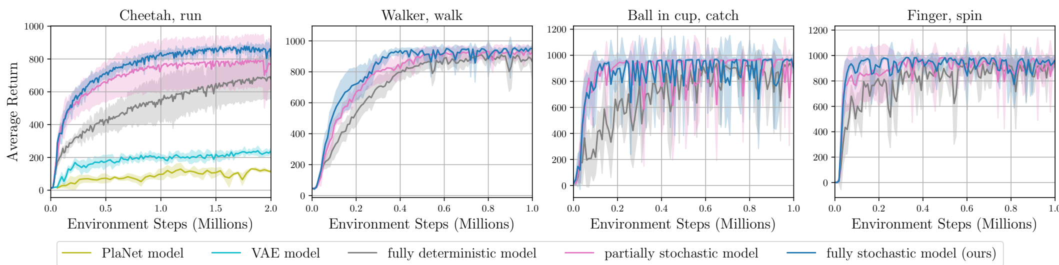


Figure 8: Comparison of different design choices for the latent variable model. In all cases, we use the RL framework of SLAC and only vary the choice of model for representation learning

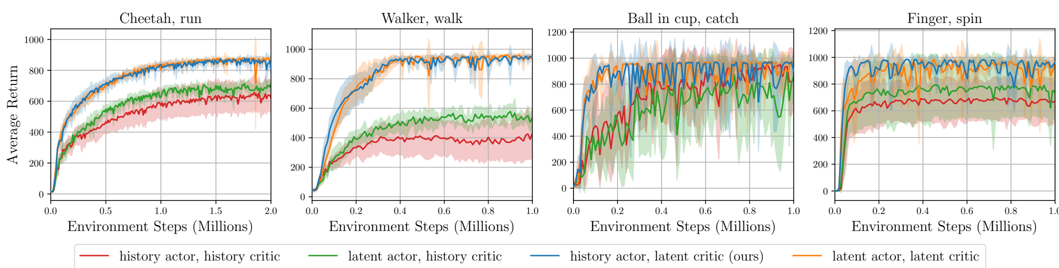


Figure 9: Comparison of alternative choices for the actor and critic inputs as either the observation-action history or the latent sample.

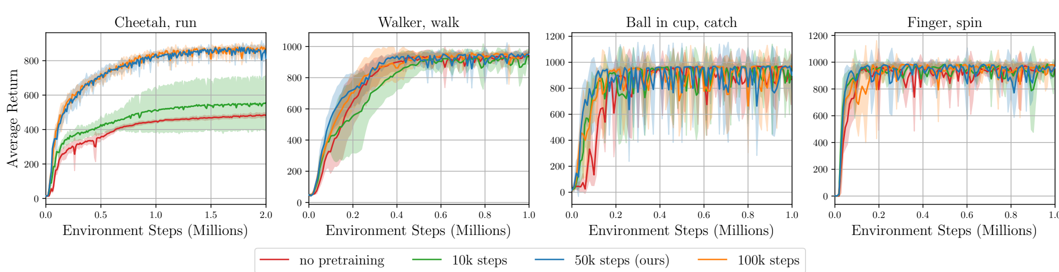


Figure 10: Comparison of the effect of pretraining the model before the agent starts learning on the task.

D Additional Ablation Experiments

We show results for the ablation experiments from Section 6.2 for additional environments. Figure 8 compares different design choices for the latent variable model. Figure 9 compares alternative choices for the actor and critic inputs as either the observation-action history or the latent sample. Figure 10 compares the effect of pretraining the model before the agent starts learning on the task.

E Predictions from the Latent Variable Model

We show example image samples from our learned sequential latent variable model in Figure 11 and Figure 12. Samples from the posterior show the images $\mathbf{x}_t$ as constructed by the decoder $p_\psi(\mathbf{x}_t|\mathbf{z}_t)$, using a sequence of latents $\mathbf{z}_t$ that are encoded and sampled from the posteriors, $q_\psi(\mathbf{z}_1|\mathbf{x}_1)$ and $q_\psi(\mathbf{z}_{t+1}|\mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t)$. Samples from the prior, on the other hand, use a sequence of latents where $\mathbf{z}_1$ is sampled from $p_\psi(\mathbf{z}_1)$ and all remaining latents $\mathbf{z}_t$ are from the propagation of the previous latent state through the latent dynamics $p_\psi(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$. Note that these prior samples do not use any image frames as inputs, and thus they do not correspond to any ground truth sequence. We also show samples from the conditional prior, which is conditioned on the first image from the true sequence: for this, the sampling procedure is the same as the prior, except that $\mathbf{z}_1$ is encoded and sampled from the posterior $q_\psi(\mathbf{z}_1|\mathbf{x}_1)$, rather than being sampled from $p_\psi(\mathbf{z}_1)$. We notice that the generated images samples can be sharper and more realistic by using a smaller variance for $p_\psi(\mathbf{x}_t|\mathbf{z}_t)$ when training the model, but at the expense of a representation that leads to lower returns. Finally, note that we do not actually use the samples from the prior for training.

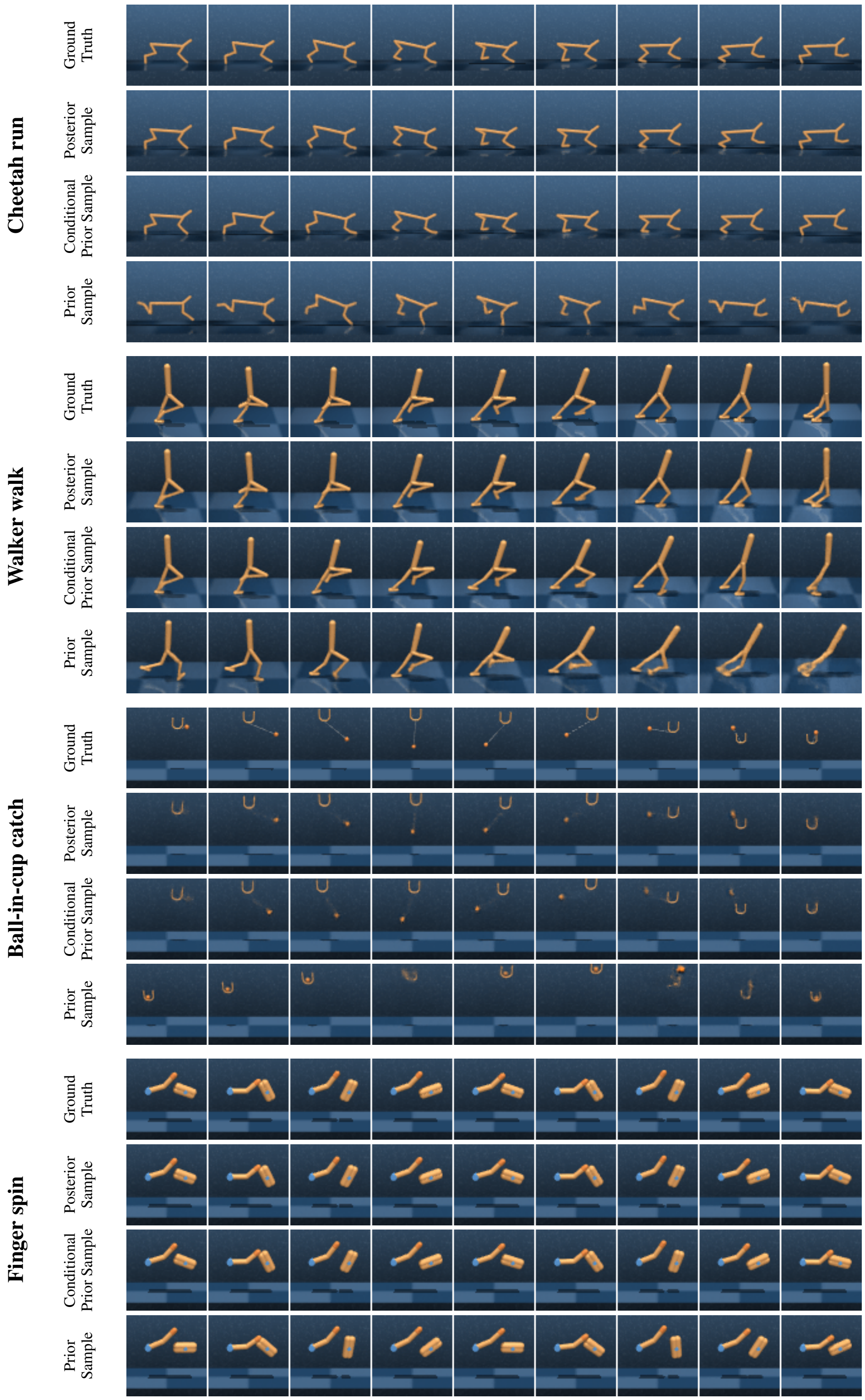


Figure 11: Example image sequences for the four DeepMind Control Suite tasks (first rows), along with corresponding posterior samples (reconstruction) from our model (second rows), and generated predictions from the generative model (last two rows). The second to last row is conditioned on the first frame (i.e., the posterior model is used for the first time step while the prior model is used for all subsequent steps), whereas the last row is not conditioned on any ground truth images. Note that all of these sampled sequences are conditioned on the same action sequence, and that our model produces highly realistic samples, even when predicting via the generative model.

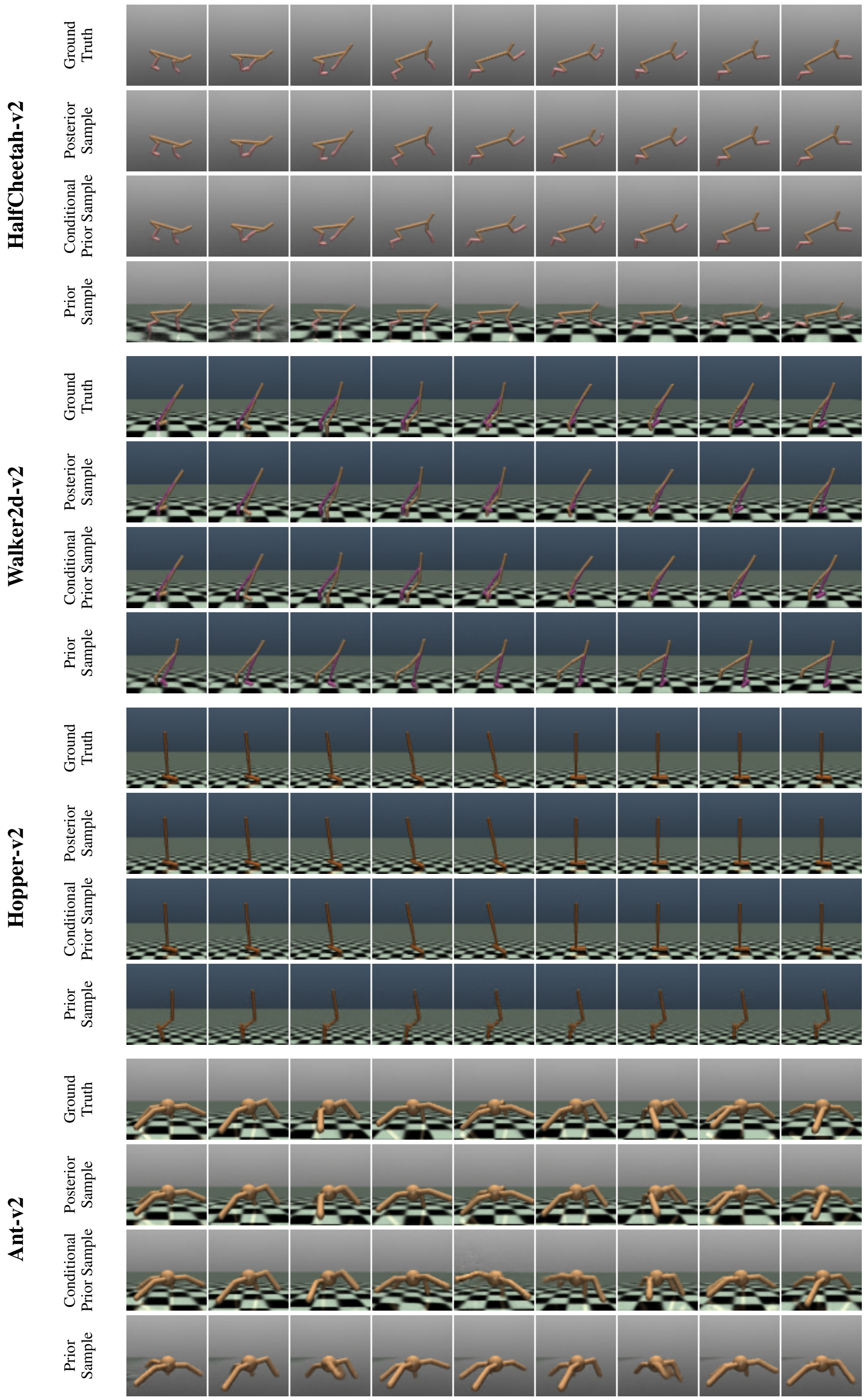


Figure 12: Example image sequences for the four OpenAI Gym tasks (first rows), along with corresponding posterior samples (reconstruction) from our model (second rows), and generated predictions from the generative model (last two rows). The second to last row is conditioned on the first frame (i.e., the posterior model is used for the first time step while the prior model is used for all subsequent steps), whereas the last row is not conditioned on any ground truth images. Note that all of these sampled sequences are conditioned on the same action sequence, and that our model produces highly realistic samples, even when predicting via the generative model.