SIMPLIFYING MODEL-BASED RL: LEARNING REPRESENTATIONS, LATENT-SPACE MODELS, AND POLICIES WITH ONE OBJECTIVE

Raj Ghugare[1,2] **Homanga Bharadhwaj**[2] **Benjamin Eysenbach**[2]
Sergey Levine[3] **Ruslan Salakhutdinov**[2]

[1]VNIT Nagpur [2]Carnegie Mellon University [3]UC Berkeley

raj19@students.vnit.ac.in, hbharadh@cs.cmu.edu, beysenba@cs.cmu.edu

ABSTRACT

While reinforcement learning (RL) methods that learn an internal model of the environment have the potential to be more sample efficient than their model-free counterparts, learning to model raw observations from high dimensional sensors can be challenging. Prior work has addressed this challenge by learning low-dimensional representation of observations through auxiliary objectives, such as reconstruction or value prediction. However, the alignment between these auxiliary objectives and the RL objective is often unclear. In this work, we propose a single objective which jointly optimizes a latent-space model and policy to achieve high returns while remaining self-consistent. This objective is a lower bound on expected returns. Unlike prior bounds for model-based RL on policy exploration or model guarantees, our bound is directly on the overall RL objective. We demonstrate that the resulting algorithm matches or improves the sample-efficiency of the best prior model-based and model-free RL methods. While such sample efficient methods typically are computationally demanding, our method attains the performance of SAC in about 50% less wall-clock time.

1 INTRODUCTION

While RL algorithms that learn an internal model of the world can learn more quickly than their model-free counterparts (Hafner et al., 2018; Janner et al., 2019), figuring out exactly *what* these models should predict has remained an open problem: the real world and even realistic simulators are too complex to model accurately. Although model errors may be rare under the training distribution, a learned RL agent will often seek out the states where an otherwise accurate model makes mistakes (Jafferjee et al., 2020). Simply training the model with maximum likelihood will not, in general, produce a model that is good for model-based RL (MBRL). The discrepancy between the policy objective and the model objective is called the objective mismatch problem (Lambert et al., 2020), and remains an active area of research. The objective mismatch problem is especially important in settings with high-dimensional observations, which are challenging to predict with high fidelity.

Prior model-based methods have coped with the difficulty to model high-dimensional observations by learning the dynamics of a compact representation of observations, rather than the dynamics of the raw observations. Depending on their learning objective, these representations might still be hard to predict or might not contain task relevant information. Besides, the accuracy of prediction depends not just on the model's parameters, but also on the states visited by the policy. Hence, another way of reducing prediction errors is to optimize the policy to avoid transitions where the model is inaccurate, while achieving high returns. In the end, we want to train the model, representations, and policy to be self-consistent: the policy should only visit states where the model is accurate, the representation should encode information that is task-relevant and predictable. *Can we design a model-based RL algorithm that automatically learns compact yet sufficient representations for model-based reasoning?*

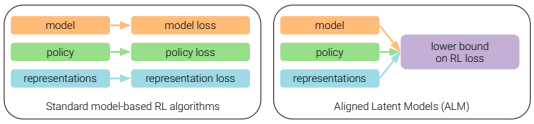


Figure 1: (*left*) Most model-based RL methods learn the representations, latent-space model, and policy using three different objectives. (*Right*) Our method jointly optimizes all three components using a single objective, which is a lower bound on expected returns.

In this paper, we present a simple yet principled answer to this question by devising a single objective that *jointly* optimizes the three components of the model-based algorithm: the representation, the model, and the policy. As shown in Fig. 1, this is in contrast to prior methods, which use three separate objectives. We build upon prior work that views model-based RL as a latent-variable problem: the objective is to maximize the returns (the likelihood), which is an expectation over trajectories (the unobserved latent variable) (Botvinick & Toussaint, 2012; Attias, 2003; Eysenbach et al., 2021a). This is different from prior work that maximizes the likelihood of observed data, independent of the reward function (Hafner et al., 2019; Lee et al., 2020). This perspective suggests that model-based RL algorithms should resemble inference algorithms, sampling trajectories (the latent variable) and then maximizing returns (the likelihood) on those trajectories. However, sampling trajectories is challenging when observations are high-dimensional. The key to our work is to infer both the trajectories (observations, actions) and the representations of the observations. Crucially, we show how to maximize the expected returns under this inferred distribution by sampling only the representations, without the need to sample high-dimensional observations.

The main contribution of this paper is Aligned Latent Models (ALM), an MBRL algorithm that jointly optimizes the observation representations, a model that predicts those representations, and a policy that acts based on those representations. To the best of our knowledge, this objective is the first lower bound for a model-based RL method with a latent-space model. Across a range of continuous control tasks, we demonstrate that ALM achieves higher sample efficiency than prior model-based and model-free RL methods, including on tasks that stymie prior MBRL methods. Because ALM does not require ensembles (Chua et al., 2018; Janner et al., 2019) or decision-time planning (Deisenroth & Rasmussen, 2011; Sikchi et al., 2020; Morgan et al., 2021), our open-source implementation performs updates $10\times$ and $6\times$ faster than MBPO (Janner et al., 2019) and REDQ (Chen et al., 2021) respectively, and achieves near-optimal returns in about 50% less time than SAC.

2 Related Work

Prior model-based RL methods use models in many ways, using it to search for optimal action sequences (Garcia et al., 1989; Springenberg et al., 2020; Hafner et al., 2018; Chua et al., 2018; Hafner et al., 2019; Xie et al., 2020), to generate synthetic data (Sutton, 1991; Luo et al., 2018; Hafner et al., 2019; Janner et al., 2019; Shen et al., 2020), to better estimate the value function (Deisenroth & Rasmussen, 2011; Chua et al., 2018; Buckman et al., 2018; Feinberg et al., 2018), or some combination thereof (Schrittwieser et al., 2020; Hamrick et al., 2020; Hansen et al., 2022). Similar to prior work on stochastic value gradients (Heess et al., 2015; Hafner et al., 2019; Clavera et al., 2020; Amos et al., 2020), our approach uses model rollouts to estimate the value function for a policy gradient.

Because learning a model of high-dimensional observations is challenging, many prior model-based methods first learn a compact representation using a representation learning objective (e.g., image reconstruction (Kaiser et al., 2019; Oh et al., 2015; Buesing et al., 2018; Ha & Schmidhuber, 2018; Hafner et al., 2018; 2019; 2020), value and action prediction (Oh et al., 2017; Schrittwieser et al., 2020; Grimm et al., 2020), planning performance (Tamar et al., 2016; Racanière et al., 2017; Okada et al., 2017), or self-supervised learning (Deng et al., 2021; Nguyen et al., 2021; Okada & Taniguchi, 2020)). These methods then learn the dynamics of these representations (not of the raw observations), and use the model for RL. The success of these methods depends on the representation: the representations should be compact (i.e., easy to predict) while retaining task-relevant information. However, prior work does not optimize for this criterion, but instead optimizes the representation using some auxiliary objective.

The standard RL objective is to to maximize the expected returns, but models are typically learned via a different objective (maximum likelihood) and representations are learned via a third objective (e.g., image reconstruction). To solve this *objective mismatch* (Lambert et al., 2020; Joseph et al., 2013; Grimm et al., 2020), prior work study decision aware loss functions which optimize the model to minimize the difference between true and imagined next step values (Farahmand et al., 2017; Farahmand, 2018; D'Oro et al., 2020; Abachi et al., 2020; Voelcker et al., 2022) or directly optimize the model to produce high-return policies (Eysenbach et al., 2021a; Amos et al., 2018; Nikishin et al., 2021). However, effectively addressing the objective mismatch problem for latent-space models remains an open problem. Our method makes progress on this problem by proposing a single objective to be used for jointly optimizing the model, policy, and representation. Because all components

are optimized with the same objective, updates to the representations make the policy better (on this objective), as do updates to the model. While prior theoretical work in latent space models has proposed bounds on exploratory behavior of the policy (Misra et al., 2020), and on learning compressed latent representations (Efroni et al., 2021), our analysis lifts some of the assumptions (e.g., removing the Block-MDP assumption), and bounds the overall RL objective in a model-based setting.

3 A Unified Objective for Latent-Space Model-Based RL

We first introduce notation, then provide a high-level outline of the objective, and then derive our objective. Sec. 4 will discuss a practical algorithm based on this objective.

3.1 Preliminaries

The agent interacts with a Markov decision process (MDP) defined by states s_t, actions a_t, an initial state distribution $p_0(s)$, a dynamics function $p(s_{t+1} \mid s_t, a_t)$, a positive reward function $r(s_t, a_t) \geq 0$ and a discount factor $\gamma \in [0, 1)$. The RL objective is to learn a policy $\pi(a_t \mid s_t)$ that maximizes the discounted sum of expected rewards within an infinite-horizon episode:

$$\max_{\pi} \mathbb{E}_{s_{t+1} \sim p(\cdot|s_t,a_t), a_t \sim \pi(\cdot|s_t)} \left[(1-\gamma) \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \tag{1}$$

The factor of $(1-\gamma)$ does not change the optimal policy, but simplifies the analysis (Janner et al., 2020; Zahavy et al., 2021). We consider policies that are factored into two parts: an observation encoder $e_\phi(z_t \mid s_t)$ and a representation-conditioned policy $\pi_\phi(a_t \mid z_t)$. Our analysis considers infinite-length trajectories τ, which include the actions a_t, observations s_t, and the corresponding observation representations z_t: $\tau \triangleq (s_0, a_0, z_0, s_1, a_1, z_1, \cdots)$. To simplify notation, we write the discounted sum of rewards as $R(\tau) \triangleq (1-\gamma) \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$. Lastly, we define the Q-function of a policy parameterized by ϕ, as $Q(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_t, z_t \sim p(\cdot|s_t,a_t), \pi_\phi(\cdot|z_t), e_\phi(\cdot|s_t)} [R(\tau)]$.

3.2 Method Overview

Our method consists of three components, shown in Fig. 2. The *first component* is an encoder $e_\phi(z_t \mid s_t)$, which takes as input a high-dimensional observation s_t and produces a compact representation z_t. This representation should be as compact as possible, while retaining the bits for selecting good actions and for predicting the Q-function. The *second component* is a dynamics model of representations, $m_\phi(z_{t+1} \mid z_t, a_t)$, which takes as input the representation of the current observation and the action and predicts the representation of the next observation. The *third component* is a policy $\pi_\phi(a_t \mid z_t)$, which takes representations as inputs and chooses an action. This policy will be optimized to select actions to maximize rewards, while also keeping the agent in states where the dynamics model is accurate.

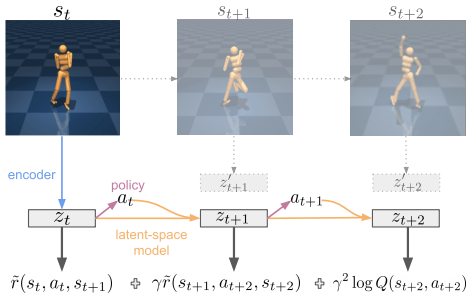


Figure 2: **Aligned Latent Models (ALM)** performs model-based RL by jointly optimizing the policy, the latent-space model, and the representations produced by the encoder using the same objective: maximize predicted rewards while minimizing the errors in the predicted representations. This objective corresponds to RL with an augmented reward function $\tilde{r}$. ALM estimates this objective without predicting high-dimensional observations s_{t+1}.

3.3 Deriving the Objective

To derive our objective, we build on prior work (Toussaint, 2009; Kappen et al., 2012) and view the RL objective as a latent-variable problem, where the return $R(\tau)$ is the likelihood and the trajectory τ is the latent variable. Different from prior work, we include *representations* in this trajectory, in addition to the raw states, a difference which allows our method to learn good representations for

MBRL. We can write the RL objective (Eq. 1) in terms of trajectories as $\mathbb{E}_{p(\tau)}[R(\tau)]$ by defining the distribution over trajectories $p(\tau)$ as

$$p_\phi(\tau) \triangleq p_0(s_0) \prod_{t=0}^{\infty} p(s_{t+1} \mid s_t, a_t) \pi_\phi(a_t \mid z_t) e_\phi(z_t \mid s_t). \tag{2}$$

Estimating and optimizing this objective directly is challenging because drawing samples from $p(\tau)$ requires interacting with the environment, an expensive operation. What we would like to do instead is estimate this same expectation via trajectories sampled from a different distribution, $q(\tau)$. We can estimate a lower bound on the expected return objective using samples from this different objective, by using the standard evidence lower bound (Jordan et al., 1999):

$$\log \mathbb{E}_{p(\tau)}[R(\tau)] \geq \mathbb{E}_{q(\tau)} \left[\log R(\tau) + \log p(\tau) - \log q(\tau) \right]. \tag{3}$$

This lower bound resolves a first problem, allowing us to estimate (a bound on) the expected return by drawing samples from the learned model, rather than from the true environment. However, learning a distribution over trajectories is difficult due to challenges in modeling high-dimensional observations, and potential compounding errors during sampling that can cause the policy to incorrectly predict high returns. We resolve these issues by only sampling compact representations of observations, instead of the observations themselves. By learning to predict the rewards and Q-values as a function of these representations, we are able to estimate this lower bound without sampling high-dimensional observations. Further, we carefully parameterize the learned distribution $q(\tau)$ to support an arbitrary length of model rollouts (K), which allows us to estimate the lower bound accurately:

$$q_\phi^K(\tau) = p_0(s_0) e_\phi(z_0 \mid s_0) \pi_\phi(a_0 \mid z_0) \prod_{t=1}^{K} p(s_t \mid s_{t-1}, a_{t-1}) m_\phi(z_t \mid z_{t-1}, a_{t-1}) \pi_\phi(a_t \mid z_t). \tag{4}$$

While it may seem strange that the future representations sampled from $q_\phi^K(\tau)$ are independent of states, this is an important design choice. It allows us to estimate the lower bound for any policy, using only samples from the latent-space model, without access to high dimensional states from the environment's dynamics function.

Combining the lower bound (Eq. 3) with this choice of parameterization, we obtain the following objective for model-based RL:

$$\mathcal{L}_\phi^K \triangleq \mathbb{E}_{q_\phi^K(\tau)} \left[\left(\sum_{t=0}^{K-1} \gamma^t \tilde{r}(s_t, a_t, s_{t+1}) \right) + \gamma^K \log Q(s_K, a_K) \right], \tag{5}$$

$$\text{where } \tilde{r}(s_t, a_t, s_{t+1}) = \underbrace{(1-\gamma) \log r(s_t, a_t)}_{(a)} + \underbrace{\log e_\phi(z_{t+1} \mid s_{t+1}) - \log m_\phi(z_{t+1} \mid z_t, a_t)}_{(b)}. \tag{6}$$

This objective is an evidence lower bound on the RL objective (see Proof in Appendix A.2).

Theorem 3.1. *The following bound holds for any representation $e_\phi(z_t \mid s_t)$, latent-space model $m_\phi(z_{t+1} \mid z_t, a_t)$, policy $\pi_\phi(a_t \mid z_t)$ and $K \in \mathbb{N}$*

$$\log \mathbb{E}_{p_\phi(\tau)} \left[(1-\gamma) \sum_t \gamma^t r(s_t, a_t) \right] \geq \mathcal{L}_\phi^K.$$

3.4 Connections with Prior Work

In this section, we provide intuition for our objective and relate it to objectives in prior work. We start by looking at the augmented reward. The first term (a: extrinsic term) in this augmented reward function is the log of true rewards, which is analogous to maximizing the true reward function in the real environment, albeit on a different scale. The second term (b: intrinsic term), i.e., the negative KL divergence between the latent-space model and the encoder, is reminiscent of the prior methods (Goyal et al., 2019; Eysenbach et al., 2021b; Bharadhwaj et al., 2021; Rakelly et al., 2021) that regularize the encoder against a prior, to limit the number of bits used from the observations. Taken together, all the components are trained to make the model self-consistent with the policy and representation.

Algorithm 1 The ALM objective can be optimized with any RL algorithm. We present an implementation based on DDPG (Lillicrap et al., 2015).

Initialize the encoder $e_\phi(z_t \mid s_t)$, model $m_\phi(z_{t+1} \mid z_t, a_t)$, policy $\pi_\phi(a_t \mid z_t)$, classifier $C_\theta(z_{t+1}, a_t, z_t)$, reward $r_\theta(z_t, a_t)$, Q-function $Q_\theta(z_t, a_t)$, replay buffer $\mathcal{B}$
for $n = 1, \cdots, N$ do **do**
  Select action $a_n = \pi_\phi(a_n \mid e_\phi(s_n)) + \mathcal{N}$ using the current policy and exploration noise $\mathcal{N}$.
  Execute action a_n and observe reward r_n and next state s_{n+1}.
  Store transition (s_n, a_n, r_n, s_{n+1}) in $\mathcal{B}$; sample length-K sequence $(s_i, a_i, s_{i+1}\}_{i=t}^{t+K-1} \sim \mathcal{B}$
  Compute lower bound using off-policy actions: $\mathcal{L}^K_{e_\phi, m_\phi}((s_i, a_i, s_{i+1}\}_{i=t}^{t+K-1})$ ▷ 7
  Update encoder and model by gradient ascent on off-policy lower bound: $\mathcal{L}^K_{e_\phi, m_\phi}$
  Compute lower bound using on-policy model-based trajectories: $\mathcal{L}^K_{\pi_\phi}((s_{i=t}))$ ▷ 8
  Update policy by gradient ascent on on-policy lower bound: $\mathcal{L}^K_{\pi_\phi}$
  Update classifier, Q-function and reward by gradient descent on: $\mathcal{L}_{C_\theta}, \mathcal{L}_{Q_\theta}, \mathcal{L}_{r_\theta}$ ▷ 9, 10, 11

The last part of our objective is the length of rollouts (K), that the model is used for. Our objective is directly applicable to all prior model-based RL algorithms which use the model only for a fixed number of rollouts rather than the entire horizon, like SVG style updates (Amos et al., 2020; Heess et al., 2015) and trajectory optimization (Tedrake, 2022). Larger values of K correspond to looser bounds (see Appendix A.6). Although this suggests that a model-free estimate is the tightest, a Q-function learned using function approximation with TD-estimates (Thrun & Schwartz, 1993) is biased and difficult to learn. A larger value of K decreases this bias by reducing the dependency on a learned Q-function. While the lower bound in Theorem 3.1 is not tight, we can include a learnable discount factor such that it becomes tight (see Appendix A.5). In our experiments 4, we find that the objective in Theorem 3.1 is still a good estimate of true expected returns.

In Appendix A.4, we derive a closed loop form for the optimal latent-dynamics and show that they are biased towards high-return trajectories: they reweight the true probabilities of trajectories with their rewards. We also derive a lower bound for the model-based offline RL setting, obtaining a similar objective with an additional behavior cloning term (Appendix A.7).

4 A Practical Algorithm

We now describe a practical method to jointly train the policy, model, and encoder using the lower bound ($\mathcal{L}^K_\phi$). We call the resulting algorithm Aligned Latent Models (ALM) because joint optimization means that the objectives for the model, policy, and encoder are the same; they are aligned. For training the encoder and model (latent-space learning phase), $q^K_\phi(\tau)$ is unrolled using actions from a replay buffer, whereas for training the policy (planning phase), $q^K_\phi(\tau)$ is unrolled using actions imagined from the latest policy. To estimate the lower bound using just representations, our method also learns to predict the reward and Q-function from the learned representations z_t using real data only (see Appendix C for details). Algorithm 1 provides pseudocode.

Maximizing the lower bound with respect to the encoder and latent-space model. To train the encoder and the latent-space model, we estimate the lower bound $\mathcal{L}^K_\phi$ using K-length sequences of transitions $\{s_i, a_i, s_{i+1}\}_{i=t}^{t+K-1}$ sampled from the replay buffer:

$$\mathcal{L}^K_{e_\phi, m_\phi}(\{s_i, a_i, s_{i+1}\}_{i=t}^{t+K-1}) = \mathbb{E}_{\substack{e_\phi(z_{i=t} \mid s_t) \\ m_\phi(z_{i>t} \mid z_{t:i-1}, a_{i-1})}} \Bigg[\gamma^K Q_\theta(z_K, \pi(z_K)) + \sum_{i=t}^{t+K-1} \gamma^i \left(r_\theta(z_i, a_i) - \mathrm{KL}(m_\phi(z_{i+1} \mid z_i, a_i) \| e_{\phi_{\text{targ}}}(z_{i+1} \mid s_{i+1})) \right) \Bigg]. \quad (7)$$

To optimize this objective, we sample an initial representation from the encoder and roll out the latent-space model using action sequences taken in the real environment (see Fig. 2)[1]n our code we do not use the γ discounting for Equation7. We find that using a target encoder $e_{\phi_{\text{targ}}}(z_t \mid s_t)$ to calculate the KL consistency term leads to stable learning.

[1] I

Maximizing the lower bound with respect to the policy. The latent-space model allows us to evaluate the lower bound for the current policy by generating on-policy trajectories. Starting from a sampled state s_t the latent-space model is recurrently unrolled using actions from the current policy to generate a K-length trajectory of representations and actions $(z_{t:t+K}, a_{t:t+K})$. Calculating the intrinsic term in the augmented reward $(\log e_\phi(z_{t+1} \mid s_{t+1}) - \log m_\phi(z_{t+1} \mid z_t, a_t))$ for on-policy actions is challenging, as we do not have access to the next high dimensional state s_{t+1}. Following prior work (Eysenbach et al., 2020; Eysenbach et al., 2021a), we learn a classifier to differentiate between representations sampled from the encoder $e_\phi(z_{t+1} \mid s_{t+1})$ and the latent-space model $m_\phi(z_{t+1} \mid z_t, a_1)$ to estimate this term (see Appendix C for details).

We train the latent policy by recurrently backpropagating stochastic gradients (Heess et al., 2015; Amos et al., 2020; Hafner et al., 2019) of the lower bound evaluated on this trajectory:

$$\mathcal{L}^K_{\pi_\phi}(s_t) = \mathbb{E}_{q^K_\phi(z_{t:K}, a_{t:K} \mid s_t)} \left[\sum_{i=t}^{t+K-1} \gamma^t \left(r_\theta(z_i, a_i) + c \cdot \log \frac{C_\theta(z_{i+1}, a_i, z_i)}{1 - C_\theta(z_{i+1}, a_i, z_i)} \right) + \gamma^K Q_\theta(z_K, \pi(z_K)) \right]. \tag{8}$$

Estimating the augmented reward function. During policy training, estimating the intrinsic reward, $\log e_\phi(z_{t+1} \mid s_{t+1}) - \log m_\phi(z_{t+1} \mid z_t, a_t)$, is challenging because we do not have access to the next high dimensional state (s_{t+1}). Following prior work (Eysenbach et al., 2020; Eysenbach et al., 2021a), we note that a learned classifier between representations sampled from the encoder $e_\phi(z_{t+1} \mid s_{t+1})$ versus the latent-space model $m_\phi(z_{t+1} \mid z_t, a_1)$ can also be used to estimate the difference between log-likelihoods under them, which is exactly equal to the augmented reward function:

$$\log e_\phi(z_{t+1} \mid s_{t+1}) - \log m_\phi(z_{t+1} \mid z_t, a_t) \approx \log \frac{C_\theta(z_{t+1}, a_t, z_t)}{1 - C_\theta(z_{t+1}, a_t, z_t)}.$$

Here, $C_\theta(z_{t+1}, a_t, z_t) \in [0, 1]$ is a learned classifier's prediction of the probability that z_{t+1} is sampled from the encoder conditioned on the next state after starting at (z_t, a_t) pair. The classifier is trained via the standard cross entropy loss:

$$\mathcal{L}_{C_\theta}(z_t \sim e_\phi(\cdot \mid s_t), z_{t+1}, \hat{z_{t+1}}) = \log(C_\theta(z_{t+1}, z_t, a_t)) + \log(1 - C_\theta(\hat{z_{t+1}}, z_t, a_t)). \tag{9}$$

where $z_{t+1} \sim e_\phi(\cdot \mid s_{t+1})$ is the *real* next representation and $\hat{z_{t+1}} \sim m_\phi(\cdot \mid z_t, a_t)$ is the *imagined* next representation, both from the same start (z_t, a_t) pair.

Differences between theory and experiments. While our theory suggests a coefficient $c = 1$, we use $c = 0.1$ in our experiments because it slightly improves the results. We do provide an ablation which shows that ALM performs well across different values of c Figure 16. We also omit the log of the true rewards in both Eq. 8 and 7. We show that this change is equivalent to the first one for all practical purposes (see Appendix D.1.). Nevertheless, these changes mean that the objective we use in practice is not guaranteed to be a lower bound.

5 EXPERIMENTS

Our experiments focus on whether jointly optimizing the model, representation, and policy yields benefits relative to prior methods that use different objectives for different components. We use SAC-SVG (Amos et al., 2020) as the main baseline, as it structurally resembles our method but uses different objectives and architectures. While design choices like ensembling (MBPO, REDQ) are orthogonal to our paper's contribution, we nonetheless show that ALM achieves similar sample efficiency MBPO and REDQ without requiring ensembling; as a consequence, it achieves good performance in $\sim 6\times$ less wall-clock time. Additional experiments analyze the Q-values, ablate components of our method, and visualize the learned representations and model. All plots and tables show the mean and standard deviation across five random seeds. Where possible, we used hyperparameters from prior works; for example, our network dimensions were taken from Amos et al. (2020). Additional implementation details and hyperparameters are in Appendix D and a summary of successful and failed experiments are in Appendix E.

Table 1: On the model-based benchmark from Wang et al. (2019), ALM outperforms model-based and model-free methods on $4/5$ tasks, often by a wide margin. We report mean and std. dev. across 5 random seeds. T-Humanoid-v2 and T-Ant-v2 refer to the respective truncated environments.

	T-Humanoid-v2	T-Ant-v2	HalfCheetah-v2	Walker2d-v2	Hopper-v2
ALM(3)	**5306** ± **437**	**4887** ± **1027**	**10789** ± **366**	**3006** ± **1183**	2546 ± 1074
SAC-SVG(2) (Amos et al., 2020)	501 ± 37	4473 ± 893	8752 ± 1785	448 ± 1139	**2852** ± **361**
SAC-SVG(3)	472 ± 85	3833 ± 1418	9220 ± 1431	878 ± 1533	2024 ± 1981
SVG(1) (Heess et al., 2015)	811.8 ± 241	185 ± 141	336 ± 387	252 ± 48	435 ± 163
SLBO (Luo et al., 2018)	1377 ± 150	200 ±40	1097 ± 166	207 ± 108	805 ± 142
TD3 (Fujimoto et al., 2018)	147 ± 0.7	870 ± 283	3016 ± 969	-516 ± 812	1817 ± 994
SAC (Haarnoja et al., 2018)	1470 ± 794	548 ± 146	3460 ± 1326	166 ± 1318	788 ± 977

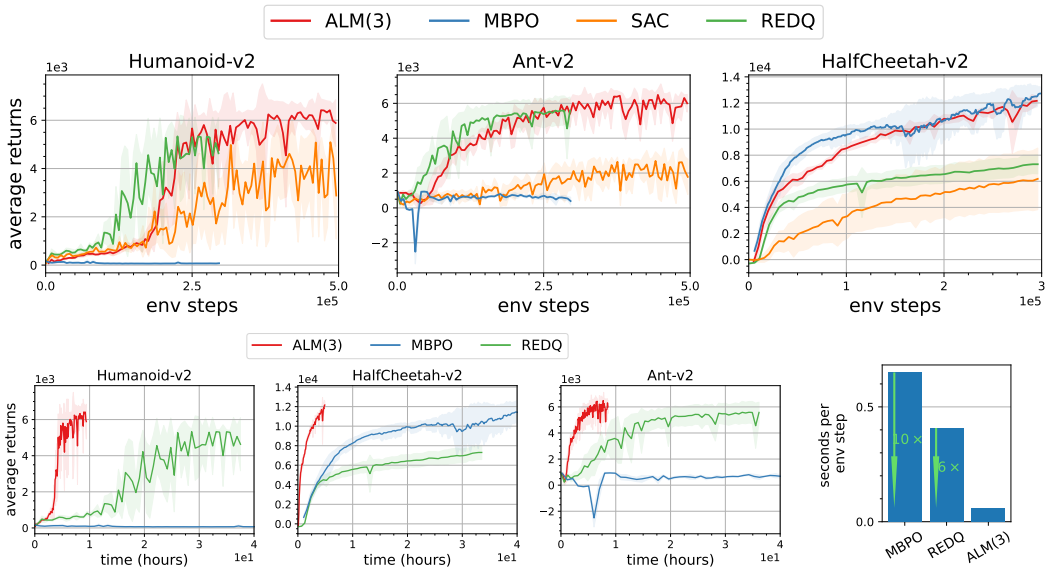


Figure 3: **Good performance without ensembles.** Our method (ALM) can (*Top*) match the sample complexity of ensembling-based methods (MBPO, REDQ) while (*Bottom*) requiring less runtime. Compared to MBPO, ALM takes $\sim 10\times$ less time per environment step. See Appendix Fig. 9 for results on other environments.

Baselines. We provide a quick conceptual comparison to baselines in Table 5. In Sec. 5.1, we will compare with the most similar prior methods, SAC-SVG (Amos et al., 2020) and SVG (Heess et al., 2015). Like ALM, these methods use a learned model to perform SVG-style actor updates. SAC-SVG also maintains a hidden representation, using a GRU, to make recurrent predictions in the observation space. While SAC-SVG learns the model and representations using a reconstruction objective, ALM trains these components with the same objective as the policy. The dynamics model from SAC-SVG bears a resemblance to Dreamer-v2 (Hafner et al., 2020) and other prior work that targets image-based tasks (our experiments target state-based tasks). SAC-SVG reports better results than many prior model-based methods (POPLIN-P (Wang & Ba, 2019), SLBO (Luo et al., 2018), ME-TRPO (Kurutach et al., 2018)).

In Sec. 5.2, we focus on prior methods that use ensembles. MBPO (Janner et al., 2019) is a model-based method that uses an ensemble of dynamics models for both actor updates and critic updates. REDQ (Chen et al., 2021) is a model-free method which achieves sample efficiency on-par with model-based methods through the use of ensembles of Q-functions. We also compare TD3 (Fujimoto et al., 2018) and SAC (Haarnoja et al., 2018); while typically not sample efficient, these methods can achieve good performance asymptotically.

5.1 Is the ALM objective useful?

We start by comparing ALM with the baselines on the locomotion benchmark proposed by Wang et al. (2019). In this benchmark, methods are evaluated based on the policy return after training for

2e5 environment steps. The TruncatedAnt-v2 and TruncatedHumanoid-v2 tasks included in this benchmark are easier than the standard Ant-v2 and Humanoid-v2 task, which prior model-based methods struggle to solve (Janner et al., 2019; Chua et al., 2018; Amos et al., 2020; Shen et al., 2020; Rajeswaran et al., 2020; Feinberg et al., 2018; Buckman et al., 2018). The results, shown in Table 1, show that ALM achieves better performance than the prior methods on 4/5 tasks. The results for SAC-SVG are taken from Amos et al. (2020) and the rest are from Wang et al. (2019). Because SAC-SVG is structurally similar to ALM, the better results from ALM highlight the importance of training the representations and the model using the same objective as the policy.

5.2 Can ALM achieve good performance without ensembles?

Prior methods such as MBPO and REDQ use ensembles to achieve SOTA sample efficiency at the cost of long training times. We hypothesize that the self-consistency property of ALM will make the latent-dynamics simpler, allowing it to achieve good sample efficiency without the use of ensembles. Our next experiment studies whether ALM can achieve the benefits of ensembles without the computational costs. As shown in Figure 3, ALM matches the sample complexity of REDQ and MBPO, but requires $\sim 6\times$ less wall-clock time to train. Note that MBPO fails to solve the highest-dimensional tasks, Humanoid-v2 and Ant-v2 ($\mathbb{R}^{376}$ and $\mathbb{R}^{111}$). We optimized the ensemble training in the official REDQ code to be parallelized, leading to an increase in training speeds by $3\times$, which is still $2\times$ slower than our method (which does not employ parallelization optimization).

5.3 Why does ALM work?

To better understand why ALM achieves high sample complexity without ensembles, we analyzed the Q-values, ran ablation experiments, and visualized the learned representations.

Analyzing the Q-values. One way of interpreting ALM is that it uses a model and an augmented reward function to obtain better estimates of the Q-values, which are used to train the policy. In contrast, REDQ uses a minimum of a random subset of the Q-function ensemble and SAC-AVG (baseline used in REDQ paper (Chen et al., 2021)) uses an average value of the Q-function ensemble to obtain a low variance estimate of these Q-values. While ensembles can be an effective way to improve the estimates of neural networks (Garipov et al., 2018; Abdar et al., 2021), we hypothesize that our latent-space model might be a more effective approach in the RL setting because it incorporates the dynamics, while also coming at a much lower computational cost.

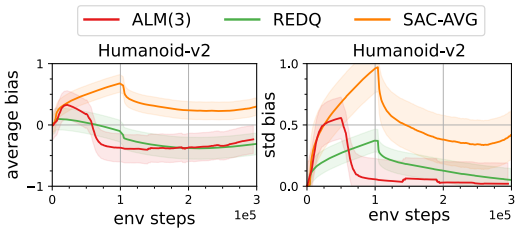
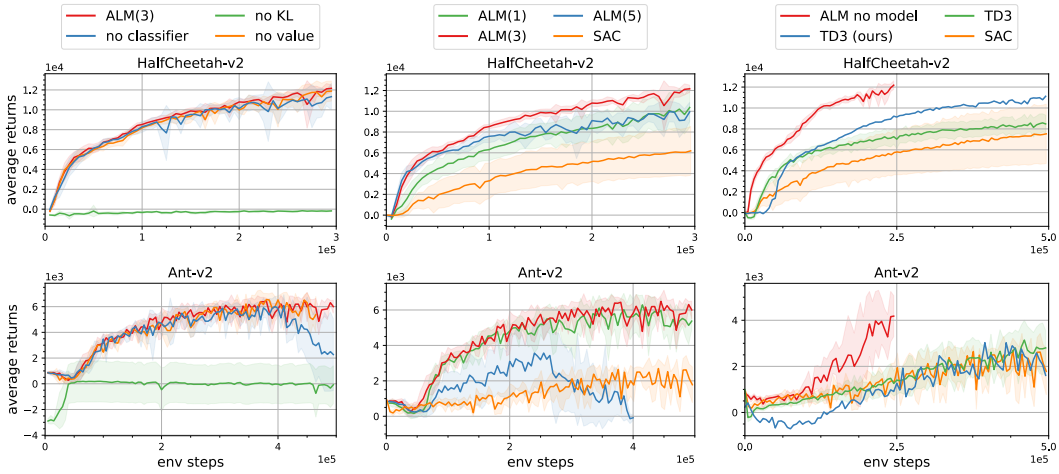


Figure 4: **Analyzing Q-values.** See text for details.

To test this hypothesis, we measure the bias of the Q-values, as well as the standard deviation of that bias, following the protocol of Chen et al. (2021); Fujimoto et al. (2018). See Appendix D.2 for details. The positive bias will tell us whether the Q-values overestimates the true returns, while the standard deviation of this bias is more relevant for the purpose of selecting actions. We see from Fig. 4 that the S.D. of the bias is lower for ALM than for REDQ and SAC-AVG, suggesting that the actions maximizing the lower bound are similar to actions that maximize true returns.

Ablation experiments. In our first ablation experiment, we compare ALM to ablations that separately remove the KL term and the value term from the encoder objective (Eq. 7), and remove the classifier term from the policy objective (Eq. 8). As shown in Fig. 5a, the KL term, which is a purely self supervised objective Grill et al. (2020), is crucial for achieving good performance. The classifier term stabilizes learning (especially on Ant and Walker), while the value term has little effect. We hypothesize that the value term may not be necessary because its effect, driving exploration, may already be incorporated by Q-value overestimation (a common problem for RL algorithms Sutton & Barto (2018); Fujimoto et al. (2018)).A second ablation experiment (Fig. 5b) shows the performance of ALM for different number of unrolling steps (K) We perform a third ablation experiment of ALM(3), which uses the TD3 actor loss for training the policy. This ablation investigates whether the representations learned by ALM(3) are beneficial for model-free RL. Prior work (Gupta et al., 2017; Eysenbach et al., 2021b; Zhang et al., 2020) has shown that representations learning can



(a) Terms in the objective. (b) Sequence length. (c) Model free RL.

Figure 5: **Ablation experiments** (*left*) Comparison of ALM (3) with no value term for the encoder, no KL term for the encoder and no classifier based rewards for the policy. Results reflect the importance of temporal consistency terms, especially for training the encoder. *(Center)* Comparison of ALM(K) for different values of K and baselines SAC. Using architectures that support larger values of K could promise further improvements in performance. (*Right*) Representation learning objective of ALM(3) leads to higher sample efficiency for model-free RL. To ensure the validity of these results, we implemented TD3 (TD3 (ours)), which uses the same architecture, exploration and learning parameters as our method. Ablation results for other environments can be found in Fig. 10a, 10b, 10c.

facilitate properties like exploration, generalization and transfer. In fig. 5c, we find that the end to end representation learning of ALM(3) achieves high returns faster than standard model-free RL.

6 Conclusion

This paper introduced ALM, an objective for model-based RL that jointly optimizes representations, latent-space models, and policies all using the same objective. This objective mends the objective mismatch problem and results in a method where the representations, model, and policy all "cooperate" to maximize a lower bound on the expected returns. Our experiments demonstrate the benefits of such joint optimization: it achieves better performance than baselines that use separate objectives, and it achieves the benefits of ensembles without their computational costs.

At a high level, our end-to-end method is reminiscent of the success *deep* supervised learning. Supervised learning methods are dependent on their input features, and end-to-end approaches have allowed researchers to avoid manual feature design. Similarly, the success of model-based methods depends on what model is used and what state representations it predicts. Our approach shows how those components can likewise be learned in an end-to-end fashion.

Limitations and future work. The main limitation of our practical method is complexity: while simpler than prior model-based methods, it has more moving parts than model-free algorithms. One potential future work that can directly stem from ALM is an on-policy version of ALM, where Equation 8 can be calculated and simultaneously optimized with the encoder, model and policy, without using a classifier. Nonetheless, we believe that our proposed objective and method are not only practically useful, but may provide a template for designing even better model-based methods with learned representations.

Acknowledgments. The authors thank Shubham Tulsiani for helpful discussions throughout the project and feedback on the paper draft. We thank Melissa Ding, Jamie D Gregory, Midhun Sreekumar, Srikanth Vidapanakal, Ola electric and CMU SCS for helping to set up the compute necessary for running all the experiments. We thank Xinyue Chen and Brandon Amos for answering questions about the baselines used in the paper, and Nicklas Hansen for helpful discussions on model-based RL.

REFERENCES

Romina Abachi, Mohammad Ghavamzadeh, and Amir-massoud Farahmand. Policy-aware model learning for policy gradient methods, 2020. URL https://arxiv.org/abs/2003.00030.

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2021.05.008. URL https://www.sciencedirect.com/science/article/pii/S1566253521001081.

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation, 2018. URL https://arxiv.org/abs/1806.06920.

Brandon Amos, Ivan Dario Jimenez Rodriguez, Jacob Sacks, Byron Boots, and J. Zico Kolter. Differentiable mpc for end-to-end planning and control, 2018. URL https://arxiv.org/abs/1810.13400.

Brandon Amos, Samuel Stanton, Denis Yarats, and Andrew Gordon Wilson. On the model-based stochastic value gradient for continuous reinforcement learning, 2020. URL https://arxiv.org/abs/2008.12775.

Hagai Attias. Planning by probabilistic inference. In Christopher M. Bishop and Brendan J. Frey (eds.), *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, volume R4 of *Proceedings of Machine Learning Research*, pp. 9–16. PMLR, 03–06 Jan 2003. URL https://proceedings.mlr.press/r4/attias03a.html. Reissued by PMLR on 01 April 2021.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Homanga Bharadhwaj, Mohammad Babaeizadeh, Dumitru Erhan, and Sergey Levine. Information prioritization through empowerment in visual model-based rl. In *International Conference on Learning Representations*, 2021.

Matthew Botvinick and Marc Toussaint. Planning as inference. *Trends in Cognitive Sciences*, 16(10):485–488, 2012. ISSN 1364-6613. doi: https://doi.org/10.1016/j.tics.2012.08.006. URL https://www.sciencedirect.com/science/article/pii/S1364661312001957.

Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/f02208a057804ee16ac72ff4d3cec53b-Paper.pdf.

Lars Buesing, Theophane Weber, Sebastien Racaniere, S. M. Ali Eslami, Danilo Rezende, David P. Reichert, Fabio Viola, Frederic Besse, Karol Gregor, Demis Hassabis, and Daan Wierstra. Learning and querying fast generative models for reinforcement learning, 2018. URL https://arxiv.org/abs/1802.03006.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model, 2021. URL https://arxiv.org/abs/2101.05982.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models, 2018. URL https://arxiv.org/abs/1805.12114.

Ignasi Clavera, Violet Fu, and Pieter Abbeel. Model-augmented actor-critic: Backpropagating through paths, 2020. URL https://arxiv.org/abs/2005.08068.

Marc Peter Deisenroth and Carl Edward Rasmussen. Pilco: A model-based and dataefficient approach to policy search. In *In Proceedings of the Twenty-Eighth International Conference on Machine Learning (ICML*, 2011.

Fei Deng, Ingook Jang, and Sungjin Ahn. Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations, 2021. URL https://arxiv.org/abs/2110.14565.

Pierluca D'Oro, Alberto Maria Metelli, Andrea Tirinzoni, Matteo Papini, and Marcello Restelli. Gradient-aware model-based policy search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3801–3808, apr 2020. doi: 10.1609/aaai.v34i04.5791. URL https://doi.org/10.1609%2Faaai.v34i04.5791.

Yonathan Efroni, Dipendra Misra, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Provable rl with exogenous distractors via multistep inverse dynamics. *arXiv preprint arXiv:2110.08847*, 2021.

Benjamin Eysenbach, Swapnil Asawa, Shreyas Chaudhari, Sergey Levine, and Ruslan Salakhutdinov. Off-Dynamics Reinforcement Learning: Training for Transfer with Domain Classifiers. *arXiv e-prints*, art. arXiv:2006.13916, June 2020.

Benjamin Eysenbach, Alexander Khazatsky, Sergey Levine, and Ruslan Salakhutdinov. Mismatched no more: Joint model-policy optimization for model-based rl, 2021a. URL https://arxiv.org/abs/2110.02758.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Robust predictable control, 2021b. URL https://arxiv.org/abs/2109.03214.

Amir-massoud Farahmand. Iterative value-aware model learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/7a2347d96752880e3d58d72e9813cc14-Paper.pdf.

Amir-Massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-Aware Loss Function for Model-based Reinforcement Learning. In Aarti Singh and Jerry Zhu (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1486–1494. PMLR, 20–22 Apr 2017. URL https://proceedings.mlr.press/v54/farahmand17a.html.

Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning, 2018. URL https://arxiv.org/abs/1803.00101.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/fujimoto18a.html.

Carlos E. Garcia, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice - a survey. *Autom.*, 25(3):335–348, 1989. URL http://dblp.uni-trier.de/db/journals/automatica/automatica25.html#GarciaPM89.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns, 2018. URL https://arxiv.org/abs/1802.10026.

Anirudh Goyal, Shagun Sodhani, Jonathan Binas, Xue Bin Peng, Sergey Levine, and Yoshua Bengio. Reinforcement learning with competitive ensembles of information-constrained primitives, 2019. URL https://arxiv.org/abs/1906.10667.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020. URL https://arxiv.org/abs/2006.07733.

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning, 2020. URL https://arxiv.org/abs/2011.03506.

Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning, 2017. URL https://arxiv.org/abs/1703.02949.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels, 2018. URL https://arxiv.org/abs/1811.04551.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination, 2019. URL https://arxiv.org/abs/1912.01603.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models, 2020. URL https://arxiv.org/abs/2010.02193.

Jessica B. Hamrick, Abram L. Friesen, Feryal Behbahani, Arthur Guez, Fabio Viola, Sims Witherspoon, Thomas Anthony, Lars Buesing, Petar Veličković, and Théophane Weber. On the role of planning in model-based deep reinforcement learning, 2020. URL https://arxiv.org/abs/2011.04021.

Nicklas A Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8387–8406. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/hansen22a.html.

Nicolas Heess, Greg Wayne, David Silver, Timothy Lillicrap, Yuval Tassa, and Tom Erez. Learning continuous control policies by stochastic value gradients, 2015. URL https://arxiv.org/abs/1510.09142.

Taher Jafferjee, Ehsan Imani, Erin Talvitie, Martha White, and Micheal Bowling. Hallucinating value: A pitfall of dyna-style planning with imperfect environment models, 2020. URL https://arxiv.org/abs/2006.04363.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization, 2019. URL https://arxiv.org/abs/1906.08253.

Michael Janner, Igor Mordatch, and Sergey Levine. Gamma-models: Generative temporal difference learning for infinite-horizon prediction. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1724–1735. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/12ffb0968f2f56e51a59a6beb37b2859-Paper.pdf.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Joshua Joseph, Alborz Geramifard, John W. Roberts, Jonathan P. How, and Nicholas Roy. Reinforcement learning with misspecified model classes. In *2013 IEEE International Conference on Robotics and Automation*, pp. 939–946, 2013. doi: 10.1109/ICRA.2013.6630686.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model-based reinforcement learning for atari, 2019. URL https://arxiv.org/abs/1903.00374.

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

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization, 2018. URL https://arxiv.org/abs/1802.10592.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. *ArXiv*, abs/2002.04523, 2020.

Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33: 741–752, 2020.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020. URL https://arxiv.org/abs/2005.01643.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2015. URL https://arxiv.org/abs/1509.02971.

Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees, 2018. URL https://arxiv.org/abs/1807.03858.

Dipendra Misra, Mikael Henaff, Akshay Krishnamurthy, and John Langford. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In *International conference on machine learning*, pp. 6961–6971. PMLR, 2020.

Andrew S. Morgan, Daljeet Nandha, Georgia Chalvatzaki, Carlo D'Eramo, Aaron M. Dollar, and Jan Peters. Model predictive actor-critic: Accelerating robot skill acquisition with deep reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2021. doi: 10.1109/icra48506.2021.9561298. URL https://doi.org/10.1109%2Ficra48506.2021.9561298.

Tung D Nguyen, Rui Shu, Tuan Pham, Hung Bui, and Stefano Ermon. Temporal predictive coding for model-based planning in latent space. In *International Conference on Machine Learning*, pp. 8130–8139. PMLR, 2021.

Evgenii Nikishin, Romina Abachi, Rishabh Agarwal, and Pierre-Luc Bacon. Control-oriented model-based reinforcement learning with implicit differentiation, 2021. URL https://arxiv.org/abs/2106.03273.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pp. 2863–2871, Cambridge, MA, USA, 2015. MIT Press.

Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network, 2017. URL https://arxiv.org/abs/1707.03497.

Masashi Okada and Tadahiro Taniguchi. Dreaming: Model-based reinforcement learning by latent imagination without reconstruction, 2020. URL https://arxiv.org/abs/2007.14535.

Masashi Okada, Luca Rigazio, and Takenobu Aoshima. Path integral networks: End-to-end differentiable optimal control, 2017. URL https://arxiv.org/abs/1706.09597.

Jan Peters, Katharina Muelling, and Yasemin Altun. Relative entropy policy search. In *AAAI*, 2010.

Rafael Figueiredo Prudencio, Marcos R. O. A. Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems, 2022. URL https://arxiv.org/abs/2203.01387.

Sébastien Racanière, Théophane Weber, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Manfred Otto Heess, Yujia Li, Razvan Pascanu, Peter W. Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. Imagination-augmented agents for deep reinforcement learning. *ArXiv*, abs/1707.06203, 2017.

Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. A game theoretic framework for model based reinforcement learning, 2020. URL https://arxiv.org/abs/2004.07804.

Kate Rakelly, Abhishek Gupta, Carlos Florensa, and Sergey Levine. Which mutual-information representation learning objectives are sufficient for control? *Advances in Neural Information Processing Systems*, 34: 26345–26357, 2021.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, dec 2020. doi: 10.1038/s41586-020-03051-4. URL https://doi.org/10.1038%2Fs41586-020-03051-4.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2015. URL https://arxiv.org/abs/1506.02438.

Jian Shen, Han Zhao, Weinan Zhang, and Yong Yu. Model-based policy optimization with unsupervised model adaptation, 2020. URL https://arxiv.org/abs/2010.09546.

Harshit Sikchi, Wenxuan Zhou, and David Held. Learning off-policy with online planning, 2020. URL https://arxiv.org/abs/2008.10066.

Jost Tobias Springenberg, Nicolas Heess, Daniel Mankowitz, Josh Merel, Arunkumar Byravan, Abbas Abdolmaleki, Jackie Kay, Jonas Degrave, Julian Schrittwieser, Yuval Tassa, Jonas Buchli, Dan Belov, and Martin Riedmiller. Local search for policy iteration in continuous control, 2020. URL https://arxiv.org/abs/2010.05545.

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

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Aviv Tamar, YI WU, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/c21002f464c5fc5bee3b98ced83963b8-Paper.pdf.

Russ Tedrake. *Underactuated Robotics*. Course Notes for MIT 6.832, 2022. URL http://underactuated.mit.edu.

Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, volume 6, pp. 1–9, 1993.

Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 1049–1056, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553508. URL https://doi.org/10.1145/1553374.1553508.

Claas Voelcker, Victor Liao, Animesh Garg, and Amir-massoud Farahmand. Value gradient weighted model-based reinforcement learning, 2022. URL https://arxiv.org/abs/2204.01464.

Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks, 2019. URL https://arxiv.org/abs/1906.08649.

Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning, 2019. URL https://arxiv.org/abs/1907.02057.

Florian Wenzel, Kevin Roth, Bastiaan S. Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really?, 2020. URL https://arxiv.org/abs/2002.02405.

Kevin Xie, Homanga Bharadhwaj, Danijar Hafner, Animesh Garg, and Florian Shkurti. Latent skill planning for exploration and transfer. In *International Conference on Learning Representations*, 2020.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning, 2021. URL https://arxiv.org/abs/2107.09645.

Tom Zahavy, Brendan O'Donoghue, Guillaume Desjardins, and Satinder Singh. Reward is enough for convex mdps, 2021. URL https://arxiv.org/abs/2106.00661.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction, 2020. URL https://arxiv.org/abs/2006.10742.

In Appendix A, we include all the proofs. In Appendix B and C, we include additional learning details and implementation details of our algorithm. In Appendix D we have mentioned implementation details. In Appendix E, we have a summary of experiments that were tried and did not help. Lastly In Appendix F, we compare various components used by ALM and the baselines.

A PROOFS

A.1 HELPER LEMMAS

Lemma A.1. *Let $P_K(H)$ be a truncated geometric distribution.*

$$P_K(H) = \begin{cases} (1-\gamma)\gamma^H & H \in [0, K-1] \\ \gamma^K & H = K \\ 0 & H > K \end{cases}$$

Given discount factor $\gamma \in (0,1)$ and a random variable x_t, we have the following identity

$$\begin{aligned}
&\mathbb{E}_{P_K(H)}\left[\sum_{t=0}^{H} x_t\right] = \sum_{H=0}^{K} P_K(H) \sum_{t=0}^{H} x_t \\
&= (1-\gamma)\sum_{H=0}^{K-1}\gamma^t \sum_{t=0}^{H} x_t + \gamma^K \sum_{t=0}^{K} x_t \\
&= x_0((1-\gamma)(1+\gamma+\cdots+\gamma^{K-1})+\gamma^K) + x_1((1-\gamma)(\gamma+\gamma^2+\cdots+\gamma^{K-1})+\gamma^K) + \cdots + x_K(\gamma^K) \\
&= x_0((1-\gamma)\frac{1-\gamma^K}{1-\gamma}+\gamma^K) + x_1((1-\gamma)(\frac{\gamma(1-\gamma^{K-1})}{1-\gamma})+\gamma^K) + \cdots + x_K(\gamma^K) \\
&= \sum_{t=0}^{K}\gamma^t x_t.
\end{aligned}$$

Lemma A.2. *Let $P_K(H)$ be a truncated geometric distribution and $p_\phi(\tau \mid H)$ be the distribution over $H+1$ length trajectories:*

$$p_\phi(\tau \mid H) = p_0(s_0)e_\phi(z_0 \mid s_0)\pi_\phi(a_0 \mid z_0)\prod_{t=1}^{H} p(s_t \mid s_{t-1}, a_{t-1})\pi_\phi(a_t \mid z_t)e_\phi(z_t \mid s_t)$$

Then the RL objective can be re-written in the following way.

$$\begin{aligned}
&\mathbb{E}_{p_\phi(\tau)}\left[(1-\gamma)\sum_t \gamma^t r(s_t, a_t)\right] \\
&\mathbb{E}_{p_\phi^K(\tau)}\left[(1-\gamma)\sum_{t=0}^{K-1}\gamma^{t-1} r(s_t, a_t) + \gamma^K Q(s_K, a_K)\right] \\
&= \mathbb{E}_{P_K(H)}\left[\mathbb{E}_{p_\phi(\tau|H=H)}\left[\mathbb{1}\{H \le K-1\} r(s_H, a_H) + \mathbb{1}\{H=K\} Q(s_H, a_H)\right]\right].
\end{aligned}$$

The Q function is defined as $Q(s_H, a_H) = \mathbb{E}_{p_\phi(\tau)}\left[(1-\gamma)\sum_{t=0}^{\infty}\gamma^t r(s_{t+H}, a_{t+H})\right]$. This lemma helps us to interpret the discounting in RL as sampling from a truncated geometric distribution over future time steps.

Lemma A.3. *Let $p(x)$ be a distribution over $\mathbb{R}^n$. The following optimization problem can be solved using the methods of Lagrange multipliers to obtain an optimal solution analytically.*

$$\begin{aligned}
&\max_{p(x)} \mathbb{E}_{p(x)}\left[f(x) - \log p(x)\right] \\
&\textit{such that} \quad \int p(x)dx = 1
\end{aligned}$$

Starting out by writing the Lagrangian for this problem, then differentiating it and then equating it to 0:

$$\mathcal{L}(x, \lambda) \stackrel{a}{=} \mathbb{E}_{p(x)} \left[f(x) - \log p(x) \right] - \lambda (\int p(x) dx - 1)$$
$$\nabla_x \mathcal{L} = f(x) - \log p(x) - 1 - \lambda$$
$$0 = f(x) - \log p(x) - 1 - \lambda$$
$$p*(x) \stackrel{b}{=} e^{f(x) - 1 - \lambda}$$

We find the value of λ by substituting the value of $p(x)$ from (b) in the equality constraint.

$$\int e^{f(x) - 1 - \lambda} dx = 1$$
$$e^{1+\lambda} \stackrel{c}{=} \int e^{f(x)} dx$$

Substituting (c) to remove the dual variable from (b), we obtain

$$p(x) \stackrel{d}{=} \frac{e^{f(x)}}{\int e^{f(x)} dx}.$$

A.2 A LOWER BOUND FOR K-STEP LATENT-SPACE

In this section we present the proof of Theorem 3.1. We restate the theorem for clarity.

Theorem 3.1. *The following bound holds for any representation $e_\phi(z_t \mid s_t)$, latent-space model $m_\phi(z_{t+1} \mid z_t, a_t)$, policy $\pi_\phi(a_t \mid z_t)$ and $K \in \mathbb{N}$*

$$\log \mathbb{E}_{p_\phi(\tau)} \left[(1 - \gamma) \sum_t \gamma^t r(s_t, a_t) \right] \geq \mathcal{L}_\phi^K.$$

Note that scaling the rewards by a constant factor $(1-\gamma)$ does not change the RL problem and finding a policy to maximize the log of the expected returns is the same as finding a policy to maximize expected returns, because $\log$ is monotonic.

We want to estimate the RL objective with trajectories sampled from a different distribution $q_\phi(\tau)$ which leads to an algorithm that avoids sampling high-dimensional observations.

$$q_\phi(\tau) = p_0(s_0) e_\phi(z_0 \mid s_0) \prod_{t=0}^{\infty} p(s_{t+1} \mid s_t, a_t) m_\phi(z_{t+1} \mid z_t, a_t) \pi_\phi(a_t \mid z_t)$$

When used as trajectory generative models, $p_\phi(\tau)$ predicts representations z_t using current state s_t. Whereas $q_\phi(\tau)$ predicts z_t by unrolling the learned latent-model recurrently on z_{t-1} (and a_{t-1}). Similar to variational inference, $p_\phi(\tau)$ can be interpreted as the posterior and $q_\phi(\tau)$ as the recurrent prior. Since longer recurrent predictions of a learned model significantly diverge from the true ones, we parameterize q to support an arbitrary length of model rollouts (K) during planning:

$$q_\phi^K(\tau) = p_0(s_0) e_\phi(z_0 \mid s_0) \pi_\phi(a_0 \mid z_0) \prod_{t=1}^{K} p(s_t \mid s_{t-1}, a_{t-1}) m_\phi(z_t \mid z_{t-1}, a_{t-1}) \pi_\phi(a_t \mid z_t)$$

Proof. We derive a lower bound on the RL objective for K-step latent rollouts.

$$
\begin{aligned}
&\log \mathbb{E}_{p_\phi(\tau)}\left[(1-\gamma)\sum_t \gamma^t r(s_t,a_t)\right] \\
&\overset{\text{a}}{=} \log \mathbb{E}_{p_\phi^K(\tau)}\left[(1-\gamma)\sum_{t=0}^{K-1}\gamma^t r(s_t,a_t)+\gamma^K Q(s_K,a_K)\right] \\
&\overset{\text{b}}{=} \log \mathbb{E}_{P_K(H)}\left[\mathbb{E}_{p_\phi(\tau|H=H)}\left[\underbrace{\mathbb{1}\{H\le K-1\}r(s_H,a_H)+\mathbb{1}\{H=K\}Q(s_H,a_H)}_{\Psi}\right]\right] \\
&\overset{\text{c}}{=} \log \iint P_K(H)p_\phi(\tau \mid H=H)\,(\Psi)\,d\tau dH \\
&\overset{\text{d}}{\ge} \int P_K(H)\log\int q_\phi(\tau \mid H=H)\frac{p_\phi(\tau \mid H=H)}{q_\phi(\tau \mid H=H)}\,(\Psi)\,d\tau dH \\
&\overset{\text{e}}{\ge} \iint P_K(H)q_\phi(\tau \mid H=H)\log\left(\frac{p_\phi(\tau \mid H=H)}{q_\phi(\tau \mid H=H)}\,(\Psi)\right)d\tau dH \\
&\overset{\text{f}}{=} \iint P_K(H)q_\phi(\tau \mid H=\infty)\left(\left(\sum_{t=0}^{H}\log e_\phi(z_{t+1}\mid s_{t+1})-\log m_\phi(z_{t+1}\mid z_t,a_t)\right)+\log(\Psi)\right)d\tau dH \\
&\overset{\text{g}}{=} \iint q_\phi(\tau)P_K(H)\left(\left(\sum_{t=0}^{H}\log e_\phi(z_{t+1}\mid s_{t+1})-\log m_\phi(z_{t+1}\mid z_t,a_t)\right)+\log(\Psi)\right)d\tau dH \\
&\overset{\text{h}}{=} \int q_\phi(\tau)\sum_{t=0}^{K-1}\gamma^t\left(\log e_\phi(z_{t+1}\mid s_{t+1})-\log m_\phi(z_{t+1}\mid z_t,a_t)\right)+(1-\gamma)\gamma^t\log r(s_t,a_t)+\gamma^K\log Q(s_K,a_K)d\tau \\
&\overset{\text{i}}{=} \mathbb{E}_{q_\phi^K(\tau)}\left[\sum_{t=0}^{K-1}\gamma^t\left(\log e_\phi(z_{t+1}\mid s_{t+1})-\log m_\phi(z_{t+1}\mid z_t,a_t)\right)+(1-\gamma)\gamma^t\log r(s_t,a_t)+\gamma^K\log Q(s_K,a_K)\right]
\end{aligned}
$$

□

In (a), we start with the K-step version of the RL objective. For (b), we use Lemma A.2. For (d), we use Jensen's inequality and multiply by $\frac{q_\phi(\tau|H=H)}{q_\phi(\tau|H=H)}$. For (e), we apply Jensen's inequality. For (f), since all the terms inside the summation only depends on the first H steps of the trajectory, we change the integration from over H length to over infinite length trajectories. For (g), we change the order of integration. For (h), we use Lemma A.1 and the fact that $\mathbb{E}_{P_K(H)}\left[\log(\mathbb{1}\{H\le K-1\}r(s_H,a_H)+\mathbb{1}\{H=K\}Q(s_H,a_H))\right] = \sum_{t=0}^{K-1}(1-\gamma)\gamma^t\log r(s_t,a_t)+\gamma^K\log Q(z_K,a_K)$.

A.3 A LOWER BOUND FOR LATENT-SPACE LAMBDA WEIGHTED SVG(K)

Using on-policy the data, the policy returns can be estimated using the k-step RL estimator (RL^k) which uses the sum of rewards for the first k steps and truncates it with the Q-function.

$$
\text{RL}^k = (1-\gamma)\sum_{t=0}^{k-1}\gamma^t r(s_t,a_t)+\gamma^k Q(s_k,a_k)
$$

Like previous works (Schulman et al., 2015; Hafner et al., 2019), we write the RL objective as λ-weighted average of k-step returns for different horizons (different values of k), to substantially reduce the variance of the estimated returns at the cost of some bias.

$$
\mathbb{E}_{p_\phi(\tau)}\left[(1-\gamma)\sum_t\gamma^t r(s_t,a_t)\right] = \mathbb{E}_{p_\phi(\tau)}\left[(1-\lambda)\sum_{k=1}^{\infty}\lambda^{n-1}\text{RL}^k\right]
$$

Our K-step lower bound $\mathcal{L}_\phi^K$ involves rolling out the model on-policy for K time-steps. A larger value of K reduces the dependency on a learned Q-function and hence reduces the estimation bias at the cost of higher variance.

Lemma A.4. *We show that a λ-weighted average of our lower bounds for different horizons (different values of K) is a lower bound on the RL objective.*

$$
\begin{aligned}
&\log \mathbb{E}_{p_\phi(\tau)}\left[(1-\gamma)\sum_t \gamma^t r(s_t,a_t)\right] \\
&\overset{\text{a}}{=} \log \mathbb{E}_{p_\phi(\tau)}\left[(1-\lambda)\sum_{k=1}^{\infty}\lambda^{k-1}RL^k\right] \\
&\overset{\text{b}}{=} \log(1-\lambda)\sum_{k=1}^{\infty}\lambda^{k-1}\mathbb{E}_{p_\phi(\tau)}\left[RL^k\right] \\
&\overset{\text{c}}{=} \log \mathbb{E}_{P_{Geom}(k)}\left[\mathbb{E}_{p_\phi(\tau)}\left[RL^{k+1}\right]\right] \\
&\overset{\text{d}}{\geq} \mathbb{E}_{P_{Geom}(k)}\log\left[\mathbb{E}_{p_\phi(\tau)}\left[RL^{k+1}\right]\right] \\
&\overset{\text{e}}{=} (1-\lambda)\sum_{k=1}^{\infty}\lambda^{k-1}\log\left[\mathbb{E}_{p_\phi(\tau)}\left[RL^{k}\right]\right] \\
&\overset{\text{f}}{\geq} (1-\lambda)\sum_{k=1}^{\infty}\lambda^{k-1}\mathcal{L}^k(\phi)
\end{aligned}
$$

In (a) we use λ-weighted average estimation of the RL objective. In (b) we use linearity of expectation. In (c), we note that for every k, the coefficient of RL^k is actually the probability of $k-1$ under the geometric distribution. Hence, we rewrite it as an expectation over the geometric distribution. In (d), we use Jensen's inequality. In (e) we write out the probability values of the geometric distribution. In (f), we use Theorem 3.1 for every value of k.

A.4 Deriving the optimal latent dynamics and the optimal discount distribution

We define $\gamma_{\phi,K}(H)$ to be a learned discount distribution over the first $K+1$ timesteps $\{0,1,\ldots,K\}$. We use this learned discount factor when using data from imaginary rollouts. We start by lower bounding the RL objective and derive the optimal discount distribution $\gamma^*_{\phi,K}(H)$ and the optimal latent dynamics distribution $q^*(\tau \mid H)$ that maximizes this lower bound.

$$
\begin{aligned}
&\overset{\text{a}}{=} \log \mathbb{E}_{P_K(H)}\left[\mathbb{E}_{p_\phi(\tau|H=H)}\left[\underbrace{\mathbb{1}\{H\leq K-1\}r(s_H,a_H)+\mathbb{1}\{H=K\}Q(s_H,a_H)}_{\Psi}\right]\right] \\
&\overset{\text{b}}{=} \log \iint\left(\frac{\gamma_{\phi,K}(H)q_\phi(\tau\mid H)}{\gamma_{\phi,K}(H)q_\phi(\tau\mid H)}P_K(H)p_\phi(\tau\mid H)\psi\right)d\tau dH \\
&\overset{\text{c}}{\geq} \int \gamma_{\phi,K}(H)\int q_\phi(\tau\mid H)\log\frac{P_K(H)p_\phi(\tau\mid H)\psi}{\gamma_{\phi,K}(H)q_\phi(\tau\mid H)}d\tau dH
\end{aligned}
$$

Given a horizon $H\in\{1,\cdots,K\}$, the optimal dynamics $q^*(\tau\mid H)$ can be calculated analytically:

$$
q^*(\tau\mid H)\overset{\text{d}}{=}\frac{p_\phi(\tau\mid H)\psi}{\int p_\phi(\tau'\mid H)\psi d\tau'}
$$

We substitute this value of q^* in equation (c):

$$\stackrel{\text{e}}{=} \int \gamma_{\phi,K}(H) \int \frac{p_\phi(\tau \mid H)\psi}{\int p_\phi(\tau' \mid H)\psi d\tau'} \log \frac{P_K(H) p_\phi(\tau \mid H)\psi \int p_\phi(\tau' \mid H)\psi d\tau'}{\gamma_{\phi,K}(H) p_\phi(\tau \mid H)\psi} d\tau dH$$

$$\stackrel{\text{f}}{=} \int \gamma_{\phi,K}(H) \log \left(\frac{P_K(H)}{\gamma_{\phi,K}(H)} \int p_\phi(\tau' \mid H)\psi d\tau' \right) dH$$

We now calculate the optimal discount distribution analytically $\gamma^*_{\phi,K}(H)$:

$$\gamma^*_{\phi,K}(H) \stackrel{g}{=} \frac{P_K(H) \int p_\phi(\tau' \mid H)\psi d\tau'}{\sum_{H=0}^{K} P_K(H) \int p_\phi(\tau' \mid H)\psi d\tau'}$$

In (a), we rewrite the RL objective using Lemma A.2. In (b), we multiply by $\frac{\gamma_{\phi,K}(H) q_\phi(\tau|H)}{\gamma_{\phi,K}(H) q_\phi(\tau|H)}$. In (c), we use Jensen's inequality. In (d) and (g) we use the method of Lagrange multipliers to derive optimal distributions. We use Lemma A.3 for this result. Below we write the optimal latent-space in terms of rewards and Q-function.

Writing out the optimal latent-space distribution: The optimal latent-dynamics are non-Markovian and do not match MDP dynamics, but are optimistic towards high-return trajectories.

$$q^*(\tau \mid H) = \begin{cases} \frac{p_\phi(\tau|H) r(s_H, a_H)}{\mathbb{E}_{p_\phi}\left[r(s'_H, a'_H)\right]} & H \in [1, K-1] \\ \frac{p_\phi(\tau|H) Q(s_H, a_H)}{\mathbb{E}_{p_\phi}\left[Q(s'_H, a'_H)\right]} & H = K \end{cases}$$

Writing out the optimal discount distribution:

$$\gamma^*_{\phi,K}(H) = \begin{cases} \frac{(1-\gamma)\gamma^H \mathbb{E}_{p_\phi}[r(s_H, a_H)]}{\mathbb{E}_{p_\phi}\left[Q(s'_0, a'_0)\right]} & H \in [0, K-1] \\ \frac{\gamma^K \mathbb{E}_{p_\phi}[Q(s_H, a_H)]}{\mathbb{E}_{p_\phi}\left[Q(s'_0, a'_0)\right]} & H = K \\ 0 & H > K \end{cases}$$

A.5 Tightening the K-step lower bound using the optimal discount distribution and the optimal latent dynamics

We start from equation (f) from Appendix A.4 the previous proof which is a lower bound on the RL objective. To derive (f), we have already substituted the optimal latent-dynamics. We now substitute the value of $\gamma^*_{\phi,K}(H)$ and verify that the lower bound (f) leads to the original objective, suggesting that the bound is tight.

$$\stackrel{\text{f}}{=} \int \gamma_{\phi,K}(H) \log \left(\frac{P_K(H)}{\gamma_{\phi,K}(H)} \int p_\phi(\tau' \mid H)\psi d\tau' \right) dH$$

$$\stackrel{h}{=} \int \frac{P_K(H) \int p_\phi(\tau' \mid H)\psi d\tau'}{\sum_{H=0}^{K} P_K(H) \int p_\phi(\tau' \mid H)\psi d\tau'} \log \left(\frac{\cancel{P_K(H)} \sum_{H=0}^{K} P_K(H) \int p_\phi(\tau' \mid H)\psi d\tau'}{\cancel{P_K(H)} \int p_\phi(\tau' \mid H)\psi d\tau'} \cancel{\int p_\phi(\tau' \mid H)\psi d\tau')} \right) dH$$

$$\stackrel{i}{=} \frac{\log(\sum_{H=0}^{K} P_K(H) \int p_\phi(\tau' \mid H)\psi d\tau')}{\cancel{\sum_{H=0}^{K} P_K(H) \int p_\phi(\tau' \mid H)\psi d\tau'}} \cancel{\int P_K(H) \int p_\phi(\tau' \mid H)\psi d\tau' dH}$$

$$\stackrel{k}{=} \log \mathbb{E}_{p_\phi(\tau)} \left[(1-\gamma) \sum_t \gamma^t r(s_t, a_t) \right]$$

In (h) we substitute the value of $\gamma^*_{\phi,K}(H)$ and cancel out common terms. Since the integration in (i) is over H, we bring out all the terms that do not depend on H. For (k), we use the result from Lemma A.2.

Hence we have proved that the bound becomes tight when using the optimal discount and trajectory distributions.

A.6 TIGHTNESS OF LOWER BOUND WITH LENGTH OF ROLLOUTS K

Proof. Proving $\mathcal{L}^K \geq \mathcal{L}^{K+1}$ for all $K \in \mathbb{N}$ will do.

$$
\begin{aligned}
&= \mathcal{L}^K \\
&= \mathbb{E}_{q_\phi^K(\tau)}\left[\sum_{t=0}^{K-1} \gamma^t \underbrace{\left(\log e_\phi(z_{t+1} \mid s_{t+1}) - \log m_\phi(z_{t+1} \mid z_t, a_t) + (1-\gamma)\log r(s_t, a_t)\right)}_{\tilde{r}(s_t, a_t, s_{t+1})} + \gamma^K \log Q(s_K, a_K)\right] \\
&= \mathbb{E}_{q_\phi^K(\tau)}\left[\sum_{t=0}^{K-1} \gamma^t \tilde{r}(s_t, a_t, s_{t+1}) + \gamma^K \log E_{p_\phi(\tau' \mid s_0 = s_K, a_0 = a_K)}\left[(1-\gamma) R(\tau')\right]\right] \\
&\geq \mathbb{E}_{q_\phi^K(\tau)}\left[\sum_{t=0}^{K-1} \gamma^t \tilde{r}(s_t, a_t, s_{t+1}) + \gamma^K E_{q_\phi(\tau' \mid s_0 = s_K, a_0 = a_K)}\left[\tilde{r}(s_K, a_K, s_{K+1}) + \gamma^{K+1} \log Q(s_{K+1}, a_{K+1})\right]\right] \\
&= \mathbb{E}_{q_\phi^{K+1}(\tau)}\left[\sum_{t=0}^{K} \gamma^t \tilde{r}(s_t, a_t, s_{t+1}) + \gamma^{K+1} \log Q(s_{K+1}, a_{K+1})\right] = \mathcal{L}^{K+1}
\end{aligned}
$$

□

A.7 A LOWER BOUND FOR LATENT-SPACE SVG(K) IN OFFLINE RL

In the offline RL setting (Levine et al., 2020; Prudencio et al., 2022), we have access to a static dataset of trajectories from the environment. These trajectories are collected from one or more unknown policies. We derive a lower bound similar to Theorem 3.1. The main difference is that in Theorem 3.1, once a policy was updated ($\phi_t \to \phi_{t+1}$), we were able to collect new data using it, while in offline RL we have to re-use the same static data. Similar to Theorem 3.1, we define the distribution over trajectories $p(\tau)$:

$$
p_{\phi,\mathrm{b}}(\tau) \triangleq p_0(s_0) \prod_{t=0}^{\infty} p(s_{t+1} \mid s_t, a_t) \pi_{\mathrm{b}}(a_t \mid s_t) e_\phi(z_t \mid s_t),
$$

such that the offline dataset consists of trajectories sampled from this true distribution. We do not assume access to the data collection policies. Rather, $\pi_{\mathrm{b}}(a_t \mid s_t)$ is the behavior cloning policy obtained from the offline dataset. We want to estimate the RL objective with trajectories sampled from a different distribution $q_\phi(\tau)$:

$$
q_\phi(\tau) = p_0(s_0) e_\phi(z_0 \mid s_0) \prod_{t=0}^{\infty} p(s_{t+1} \mid s_t, a_t) m_\phi(z_{t+1} \mid z_t, a_t) \pi_\phi(a_t \mid z_t),
$$

which leads to an algorithm that avoids sampling high-dimensional observations.

Similar to Theorem 3.1, we want to find an encoder $e_\phi(z_t \mid s_t)$, a policy $\pi_\phi(a_t \mid s_t)$, and a model $m_\phi(z_{t+1} \mid z_t, a_t)$ to maximize the RL objective:

$$
\begin{aligned}
&\max_\phi \log \mathbb{E}_{p_{\phi,\mathrm{b}}(\tau)}\left[(1-\gamma)\sum_t \gamma^t r(s_t,a_t)\right] \\
&\overset{\mathrm{a}}{=} \log \mathbb{E}_{p^K_{\phi,\mathrm{b}}(\tau)}\left[(1-\gamma)\sum_{t=0}^{K-1} \gamma^t r(s_t,a_t) + \gamma^K Q(s_t,a_t)\right] \\
&\overset{\mathrm{c}}{=} \log \iint P_K(H) p_{\phi,\mathrm{b}}(\tau \mid H=H)\,(\Psi)\,d\tau dH \\
&\overset{\mathrm{d}}{\geq} \int P_K(H) \log \int q_\phi(\tau \mid H=H) \frac{p_{\phi,\mathrm{b}}(\tau \mid H=H)}{q_\phi(\tau \mid H=H)}\,(\Psi)\,d\tau dH \\
&\overset{\mathrm{e}}{\geq} \iint P_K(H) q_\phi(\tau \mid H=H) \log\left(\frac{p_{\phi,\mathrm{b}}(\tau \mid H=H)}{q_\phi(\tau \mid H=H)}\,(\Psi)\right) d\tau dH \\
&\overset{\mathrm{f}}{=} \iint P_K(H) q_\phi(\tau \mid H=\infty)\left(\left(\sum_{t=0}^{H} \log(\frac{e_\phi(z_{t+1} \mid s_{t+1})\pi_\mathrm{b}(a_{t+1} \mid s_{t+1})}{m_\phi(z_{t+1} \mid z_t,a_t)\pi_\phi(a_{t+1} \mid z_{t+1})})\right) + \log(\Psi)\right) d\tau dH \\
&\overset{\mathrm{g}}{=} \iint q_\phi(\tau) P_K(H)\left(\left(\sum_{t=0}^{H} \log(\frac{e_\phi(z_{t+1} \mid s_{t+1})\pi_\mathrm{b}(a_{t+1} \mid s_{t+1})}{m_\phi(z_{t+1} \mid z_t,a_t)\pi_\phi(a_{t+1} \mid z_{t+1})})\right) + \log(\Psi)\right) d\tau dH \\
&\overset{\mathrm{h}}{=} \int q_\phi(\tau) \sum_{t=0}^{K-1} \gamma^t \left(\log(\frac{e_\phi(z_{t+1} \mid s_{t+1})\pi_\mathrm{b}(a_{t+1} \mid s_{t+1})}{m_\phi(z_{t+1} \mid z_t,a_t)\pi_\phi(a_{t+1} \mid z_{t+1})})\right) + (1-\gamma)\gamma^t \log r(s_t,a_t) + \gamma^K \log Q(s_K,a_K) d\tau \\
&\overset{\mathrm{i}}{=} \mathbb{E}_{q^K_\phi(\tau)}\left[\sum_{t=0}^{K-1} \gamma^t \underbrace{\log(\frac{e_\phi(z_{t+1} \mid s_{t+1})}{m_\phi(z_{t+1} \mid z_t,a_t)})}_{\text{KL consistency term}} + \gamma^t \underbrace{\log(\frac{\pi_\mathrm{b}(a_{t+1} \mid s_{t+1})}{\pi_\phi(a_{t+1} \mid z_{t+1})})}_{\text{Behaviour cloning term}} + (1-\gamma)\gamma^t \log r(s_t,a_t) + \gamma^K \log Q(s_K,a_K)\right]
\end{aligned}
$$

The only main difference is between this proof and the proof A.2 of Theorem3.1 is in step (f), where canceling out the common terms of $p(\tau)$ and $q(\tau)$ leaves an additional *behavior cloning* term. This derivation theoretically backs the additional behavior cloning term used in prior representation learning methods (Abdolmaleki et al., 2018; Peters et al., 2010) for the offline RL setting.

B Comparison to prior work on lower bound for RL (MnM)

Our approach of deriving an evidence lower bound on the RL objective is similar to prior work (Eysenbach et al., 2021a). In this section we briefly go over the connection between our method and Eysenbach et al. (2021a). The lower bound presented in (Eysenbach et al., 2021a) is a special case of the lower bound in 3.1. By taking the limit $K \to \infty$ and assuming an identity function for the encoder, we exactly reach the lower bound presented in Eysenbach et al. (2021a). By using a bottleneck policy (policy with an encoder), ALM(K) learns to represent the observations according to their importance in the control problems rather than trying to reconstruct every observation feature with high fidelity. This is supported by the fact that ALM solves Humanoid-v2 and Ant-v2, which were not solvable by prior methods like MBPO and MnM. By using the model for K steps, we have a parameter to explicitly control the planning / Q-learning bias. Hence, the policy faces a penalty (intrinsic reward term) only for the first K steps rather than the entire horizon (Eysenbach et al., 2021a), which could lead to lower returns on the training tasks (Eysenbach et al., 2021b).

C Additional learning details

Estimating the Q-function and reward function. Unlike most MBRL algorithms, the Q function $Q_\theta(z_t, a_t)$ is learned using transitions s_t, a_t, r_t, s_{t+1} from the *real environment* only, using the

Table 2: A default set of hyper-parameters used in our experiments.

Hyperparameters	Value
Discount (γ)	0.99
Warmup steps	5000
Soft update rate (τ)	0.005
Weighted target parameter (λ)	0.95
Replay Buffer	10^6 for humanoid 10^5 otherwise
Batch size	512
Learning rate	1e-4
Max grad norm	100.0
Latent dimension	50
Coefficient of classifier rewards	0.1
Exploration stddev. clip	0.3
Exploration stddev. schedule	linear(1.0 , 0.1, 100000)

standard TD loss:

$$\mathcal{L}_{Q_\theta}(z_t \sim e_\phi(\cdot \mid s_t), a_t, r_t, z_{t+1} \sim e_\phi(\cdot \mid s_{t+1})) = (Q_\theta(z_t, a_t) - (r_t + \gamma Q_{\theta_{\text{targ}}}(z_{t+1}, \pi(z_{t+1}))))^2. \tag{10}$$

The TD target is computed using a target Q-function (Fujimoto et al., 2018). We learn the reward function $r_\theta(z_t, a_t)$ using data s_t, a_t, r_t from the *real environment* only. For $r_\theta(z_t, a_t)$, by minimizing the mean squared error between true and predicted rewards :

$$\mathcal{L}_{r_\theta}(z_t \sim e_\phi(\cdot \mid s_t), a_t, r_t) = (r_\theta(z_t, a_t) - r_t)^2. \tag{11}$$

Unlike prior representation learning methods (Zhang et al., 2020), we do not use the reward or Q function training signals to train the encoder. The encoder, model, and policy are optimized using the principled joint objective only, as described in the next paragraph.

D IMPLEMENTATION DETAILS

We implement ALM using DDPG (Lillicrap et al., 2015) as the base algorithm. Following prior svg methods (Amos et al., 2020), we parameterize the encoder, model, policy, reward, classifier and Q-function as 2-layer neural networks, all with 512 hidden units except the model which has 1024 hidden units. The model and the encoder output a multivariate Gaussian distribution over the latent-space with diagonal covariance. Like prior work (Hansen et al., 2022; Yarats et al., 2021), we apply layer normalization (Ba et al., 2016) to the value function and rewards. Similar to prior work (Schulman et al., 2015; Hafner et al., 2019), we reduce variance of the policy objective in Equation 8, by computing an exponentially-weighted average of the objective for rollouts of length 1 to K. This average is also a lower bound (Appendix A.3). To train the policy, reward, classifier and Q-function we use the representation sampled from the target encoder. For exploration, we use added normal noise with a linear schedule for the std (Yarats et al., 2021). All hyperparameters are listed in Table 2. The brief summary of all the neural networks, their loss functions and the inputs to their loss functions are listed in Table 3.

Analyzing the learned representations. Because the ALM objective optimizes the encoder and model to be self-consistent (see Sec. 3.4), we expect the ALM dynamics model to remain accurate for longer rollouts than alternative methods. We test this hypothesis using an optimal trajectory from the HalfCheetah-v2 task. Starting with the representation of the initial state, we autoregressively unroll the learned model, comparing each prediction z_t to the true representation (obtained by applying the encoder to observation s_t). Fig. 6 (left) visualizes the first coordinate of the representation, and shows that the model learned via ALM remains accurate for ~ 20 steps. The ablation of ALM that removes the KL term diverges after just two steps.

Table 3: Neural networks used by ALM

Neural Network	Loss Function	Inputs to Loss
Encoder: $e_\phi(s) \to z$ Model: $m_\phi(z, a) \to z'$	joint lower bound (Eq. 5)	$\{s_i, a_i, s_{i+1}\}_{i=t}^{t+K-1}$
Policy: $\pi_\phi(z) \to a$	joint lower bound (Eq. 5)	z_t
Q-Function: $Q_\theta(z, a) \in \mathbb{R}$	TD-loss (Eq. 10)	z_t, a_t, r_t, z_{t+1}
Classifier: $C_\theta(z, a, z') \in (0, 1)$	cross entropy loss (Eq. 9)	z_t, a_t $z_{t+1} \sim e_\phi(s_{t+1})$ $\hat{z_{t+1}} \sim m_\phi(z_t, a_t)$
Reward Function: $r_\theta(z, a) \in \mathbb{R}$	MSE (Eq. 11)	z_t, a_t, r_t

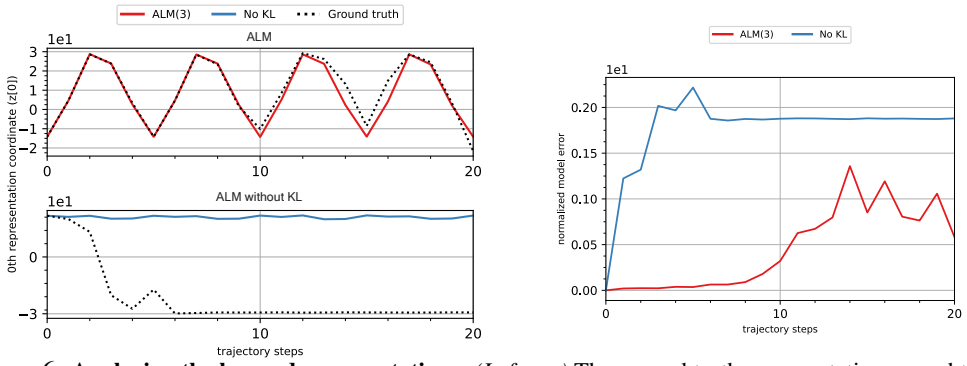


Figure 6: **Analyzing the learned representations**: *(Left-top)* The ground truth representations are obtained from the respective trained encoders on the same optimal trajectory. *(Left-bottom)* Without the KL term, the representations learnt are degenerate, i.e. they correspond to the same value for different states. (*Right*) The KL term in the ALM objective, trains the model to reduce the future K step prediction errors. The latent-space model is accurately able to approximate the true representations upto ~ 20 rollout steps.

D.1 Omission of the log of rewards and Q function is equivalent to scaling KL in Equation 8

While our main results omit the logarithmic transformation of the rewards and Q function, in this section we describe that this omission is approximately equivalent to scaling the KL coefficient in Eq. 8. Using these insights, we applied ALM, with the log of rewards, to a transformed MDP with a shifted reward function. A large enough constant, a, was added to all original rewards(which doesn't change the optimal policy assuming that there are no terminal states)

$$r_{\text{new}} = r + a.$$

Taking a log of this new reward is equal to the reward of the original objective, scaled by the constant, a

$$a(\log(r + a) - \log(a)) \approx r.$$

The additive term can be ignored because it won't contribute to the optimization. We plot both $y = r$ and $y = a(\log(r + a) - \log(a))$, to show that they are very similar for commonly used values of rewards in Figure 7. The scaling constant a can be interpreted as the weight of the log of rewards, relative to the KL term in the ALM objective. Hence changing this value is approximately equivalent to scaling the KL coefficient. The results, shown in Fig 12, show that this version of ALM which includes the logarithm transformation performs at par with the version of ALM without the logarithm. This results shows that we can add back the logarithm to ALM without hurting performance. For both Figure 7 and Figure 12, we used the value of $a = 10000$.

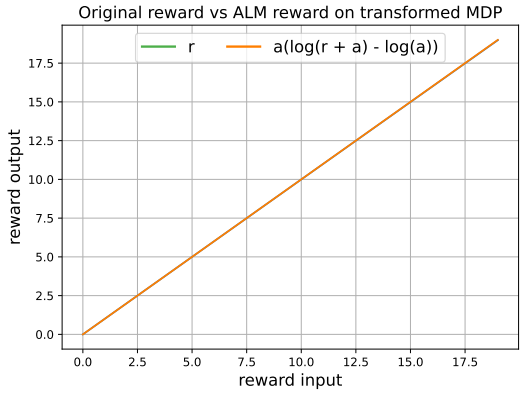


Figure 7:

ALM objective using the transformed reward

$$\stackrel{\text{a}}{=} \mathbb{E}_{q_\phi^K(\tau)}\left[\sum_{t=0}^{K-1}\gamma^t\left(\log e_\phi(z_{t+1}\mid s_{t+1}) - \log m_\phi(z_{t+1}\mid z_t,a_t)\right) + (1-\gamma)\gamma^t\log(r_{\text{new}}(s_t,a_t)) + \gamma^K\log Q_{\text{new}}(s_K,a_K)\right]$$

$$\stackrel{\text{b}}{=} \mathbb{E}_{q_\phi^K(\tau)}\Bigg[[\sum_{t=0}^{K-1}\gamma^t\left(\log e_\phi(z_{t+1}\mid s_{t+1}) - \log m_\phi(z_{t+1}\mid z_t,a_t)\right) + (1-\gamma)\gamma^t\left(\frac{r(s_t,a_t)}{a}+\log(a)\right) + \gamma^K\left(\frac{Q(s_K,a_K)}{a}+\log(a)\right)\Bigg]$$

$$\stackrel{\text{c}}{=} \mathbb{E}_{q_\phi^K(\tau)}\left[[\sum_{t=0}^{K-1}\gamma^t\left(\log e_\phi(z_{t+1}\mid s_{t+1}) - \log m_\phi(z_{t+1}\mid z_t,a_t)\right) + (1-\gamma)\gamma^t(\frac{r(s_t,a_t)}{a}) + \gamma^K(\frac{Q(s_K,a_K)}{a})\right]$$

$$\stackrel{\text{d}}{=} \mathbb{E}_{q_\phi^K(\tau)}\left[[\sum_{t=0}^{K-1}\gamma^t a\left(\log e_\phi(z_{t+1}\mid s_{t+1}) - \log m_\phi(z_{t+1}\mid z_t,a_t)\right) + (1-\gamma)\gamma^t r(s_t,a_t) + \gamma^K Q(s_K,a_K)\right]$$

In (a), we write the ALM objective using the new reward function. In (b), we use the fact that $\log(r_{\text{new}}(s_t,a_t)) \approx \left(\frac{r(s_t,a_t)}{a}+\log(a)\right)$, and $\log Q_{\text{new}}(s_K,a_K) \equiv \left(\frac{Q(s_K,a_K)}{a}+\log(a)\right)$ for a large enough constant a. (See explanation above). In (c), we remove the extra constants which add up to 0. In (d), we note that scaling the rewards and Q function by $1/a$, is equivalent to scaling the KL term by a, which is a large enough constant.

D.2 Bias and Variance of the Lower Bound

We recreate the experimental setup of (Chen et al., 2021) to evaluate the average and the std value of the bias between the Monte-Carlo returns and the estimated returns (lower bound $\mathcal{L}_\phi^K(s,a)$ for our experiments). Similar to (Chen et al., 2021), since the true returns change as training progresses we analyze the mean and std value of normalized bias, which is defined as: $(\mathcal{L}_\phi^K(s,a) - Q^\pi(s,a))/|E_{s',a'\sim\pi}(s',a')|$. This helps us to evaluate bias relative to the scale of average Monte-Carlo returns of the current policy. Results for Ant-v2 and Walker2d-v2 are presented in Fig. 8.

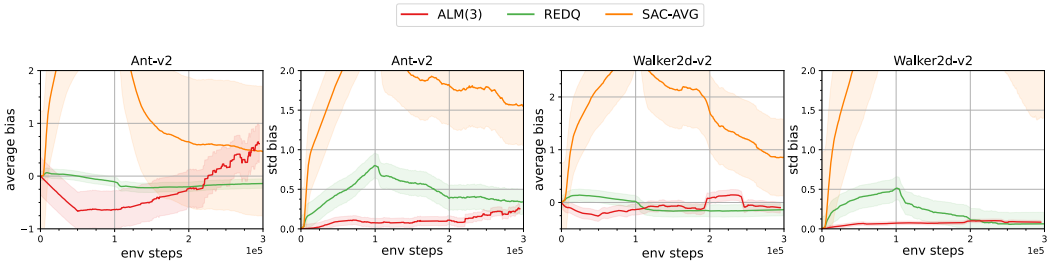


Figure 8: In accordance to what our theory suggests, the joint objective is a biased estimate of the true returns. The std values are uniform throughout training and consistently lower than REDQ, which could be the reason behind the sample efficiency of ALM(3)

E Failed Experiments

Experiments that we tried and that did not help substantially:

- Value expansion: Training the critic using data generated from the model rollouts. The added complexity did not add much benefit.
- Warm up steps: Training the policy using real data for a fixed time-steps at the start of training.
- Horizon scheduling: Scheduling the sequence length from 1 to K at the start of training.
- Exponential discounting: Down-scaling the learning rate of future time-steps using a temporal discount factor, to avoid exploding or vanishing gradients.

Experiments that were tried and found to help:

- Target encoder: Using a target encoder for the KL term in Eq. 7 helped reduce variance in episode returns.
- Elu activation: Switching from Relu to Elu activations for all networks for ALM resulted in more stable and sample efficient performance across all tasks.

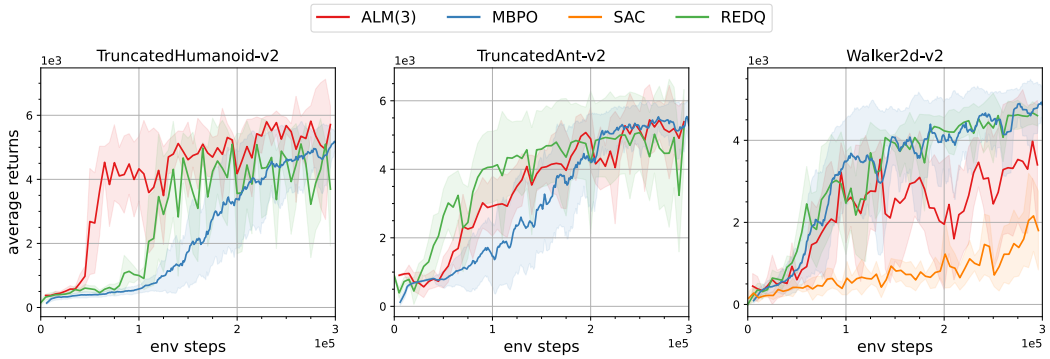
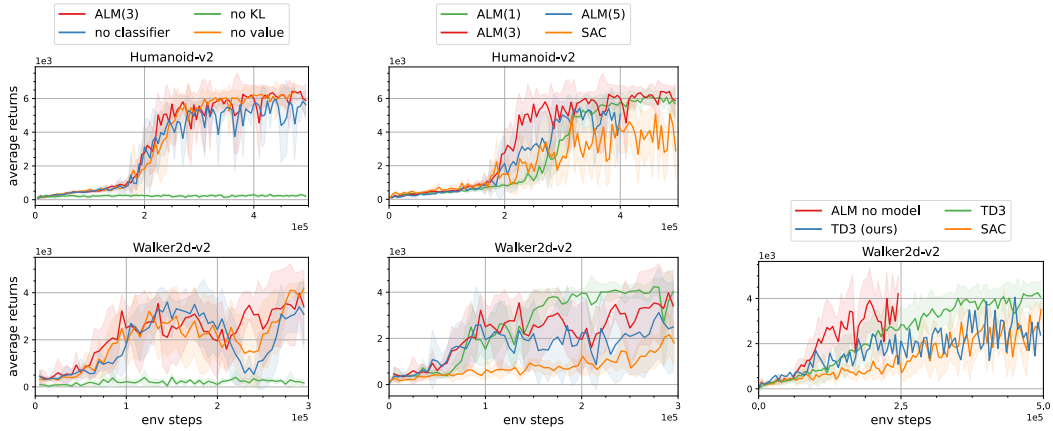


Figure 9: ALM generally matches the sample efficiency of MBPO and REDQ at a fraction of the computation complexity.



(a) Additional results for comparing different components of the joint objective on Humanoid-v2 and Walker-v2. The KL consistency term is the most important contributing factor.

(b) Additional results for comparing different horizon lengths. All horizon lengths generally perform better than sac, but horizon length 3 performs best.

(c) Additional results for comparing ALM(3)'s representations with pure model-free RL methods.

Figure 10: **Additional ablation experiments.**

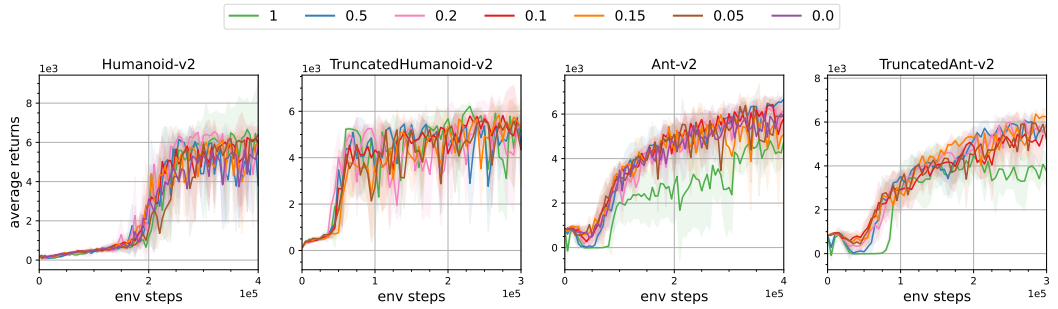


Figure 11: **ALM is robust across different values of the coefficient for the KL term in Equation 8.** In our implementation, we deviate from the value 1, because it leads to relatively gradual increase in returns on some environments. This is expected because higher coefficient to the KL term leads to higher compression Eysenbach et al. (2021b). We add that prior work on variational inference Wenzel et al. (2020) also finds that scaling the KL term can improve results.

Table 4: Where possible, we have tried to use the same hyperparameters and architectures as SAC-SVG. We use the same training hyperparameters like update to data collection ratio, batch size and rollout length when compared to sac-svg. Both methods use the same policy improvement technique: stochastic value gradients. The two exceptions are decisions that simplify our method: unlike SAC-SVG, we use a feedforward dynamics model instead of an RNN; unlike SAC-SVG, we use a simple random noise instead of a more complex entropy schedule for exploration. To test whether the soft actor critic's entropy, used in SAC-SVG can be a confounding factor causing SAC-SVG to perform worse than ALM, we compare a version of ALM which uses a soft actor critic entropy bonus like the SAC-SVG.

Method	T-Humanoid-v2	T-Ant-v2
ALM-ENT(3)	4747 $\pm$ 900	**4921** $\pm$ **128**
ALM(3)	**5306** $\pm$ **437**	4887 $\pm$ 1027
SAC-SVG(3)	501 $\pm$ 307	5306 $\pm$ 437
SAC-SVG(2)	472 $\pm$ 85	3833 $\pm$ 1418

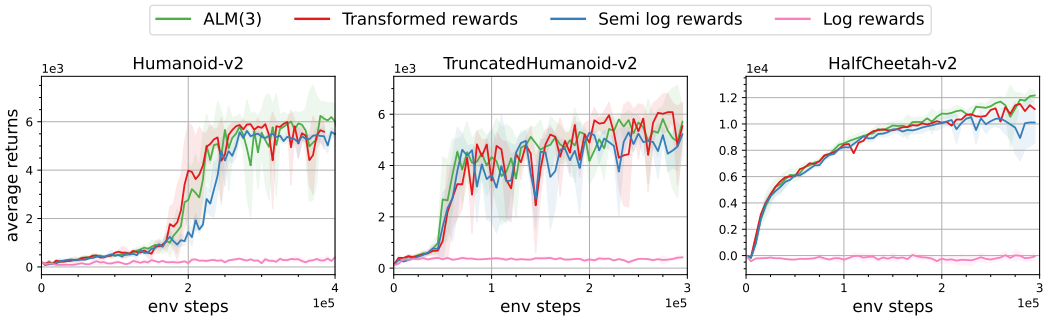


Figure 12: **Using the logarithm does not actually hurt performance.** Our theory suggests that we should take the logarithm of the reward function and Q-function. Naïvely implemented, this logarithmic transformation (pink) performs much worse than omitting the transformation (green). We also see that using a log of rewards for only training the encoder and model does not affect the performance (blue). We hypothesize that the non linearity of log(x) for reward values makes the Q-values similar for different actions. However, by transforming the reward function (which does not change the optimization problem), we are able to include the theoretically-suggested logarithm while retaining high performance (red). See Appendix D.1 for more details.

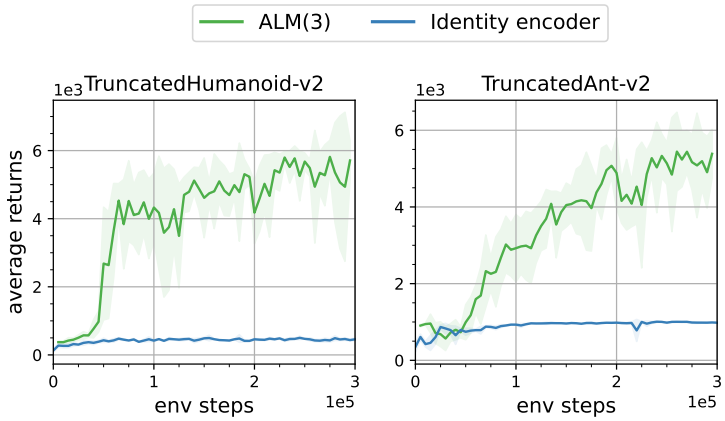


Figure 13: **Comparison with an MnM ablation.** The main difference between ALM and a prior joint optimization method (MnM), is that ALM learns the encoder function. Replacing that learned encoder with an identify function yields a method that resembles MnM, and performs much worse. This result supports our claim that RL methods that use latent-space models can significantly outperform state-space models.

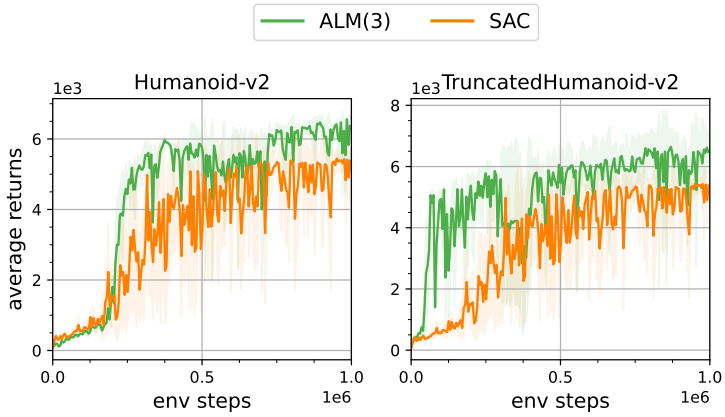


Figure 14: **Asymptotic performance.** Even after 1 million environment steps, ALM still outperforms the SAC baseline.

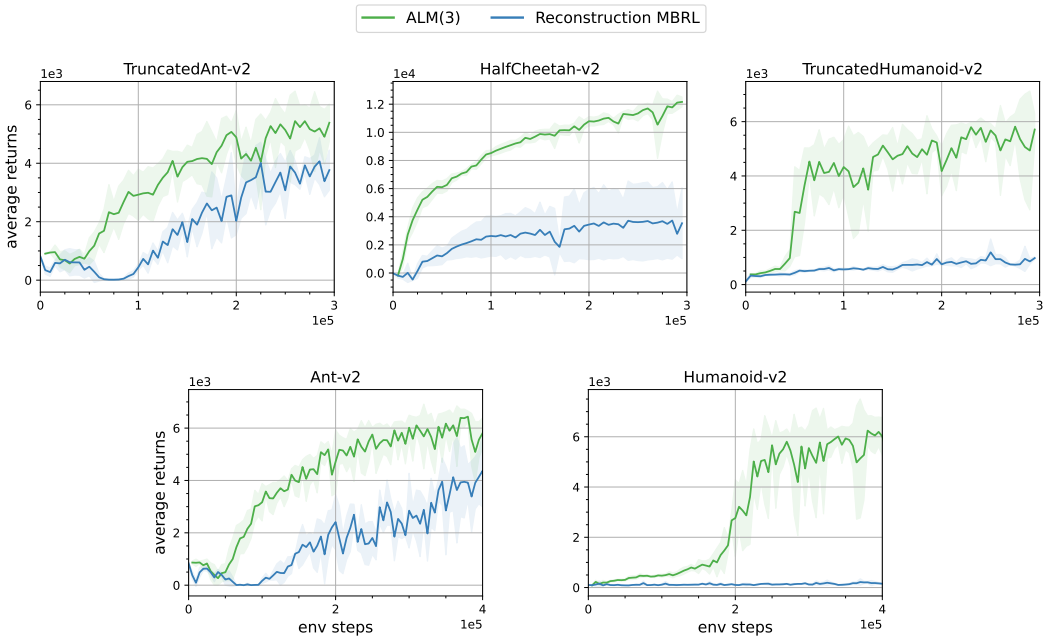


Figure 15: **Importance of an aligned objective.** Comparison with a version of ALM that learns policies using the same objective, but learns representations and latent-space models to maximize likelihood. This further verifies the claims made in Section 5.1 that using an aligned objective for joint optimization leads to superior performance

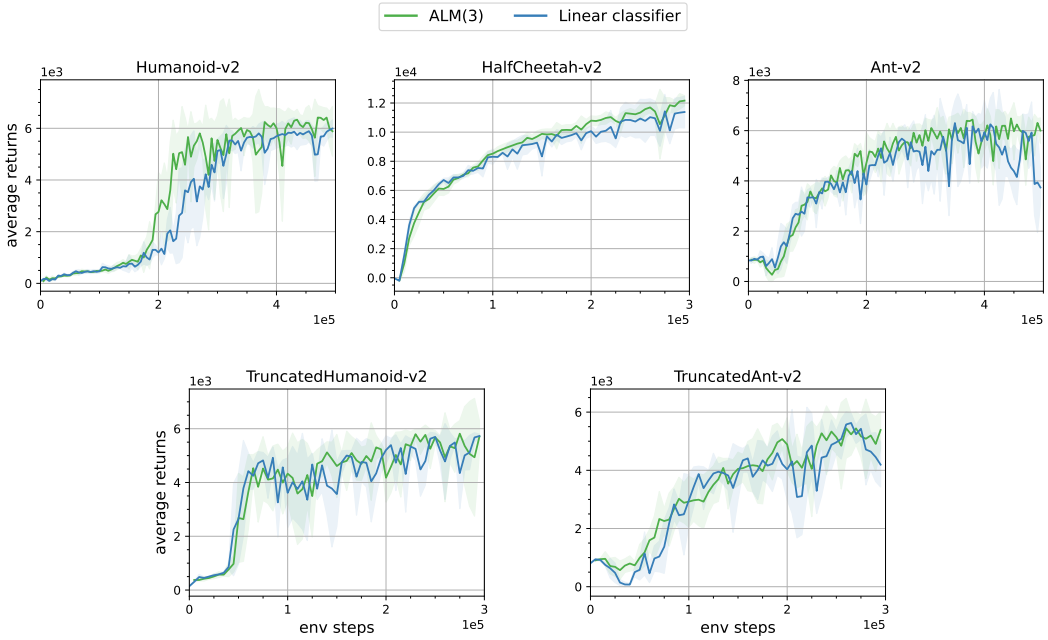


Figure 16: **Robustness to classifier errors.** ALM retains a high degree of performance even when the classifier is restricted to be a linear function, showing how ALM can be robust to errors in the classifier.

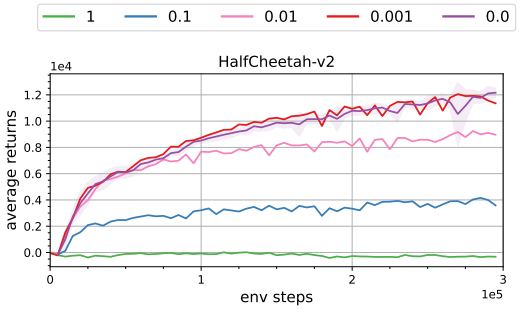


Figure 17: **Robustness towards various levels of aleatoric uncertainty.** We created a version of the HalfCheetah-v2 task with varying levels of aleatoric noise.

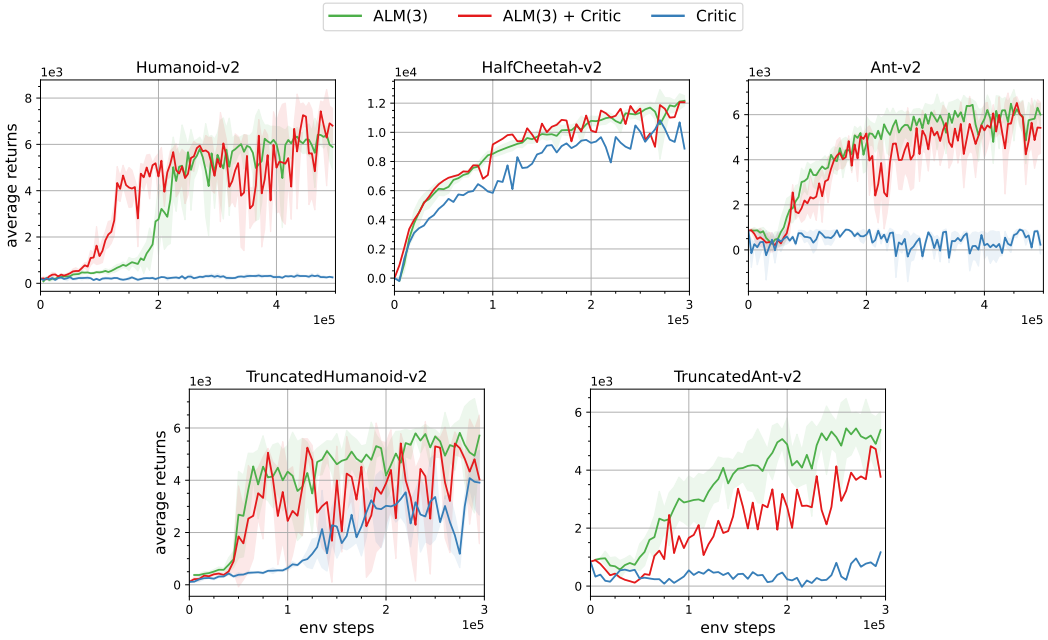


Figure 18: **Effect of learning to predict the value function.**

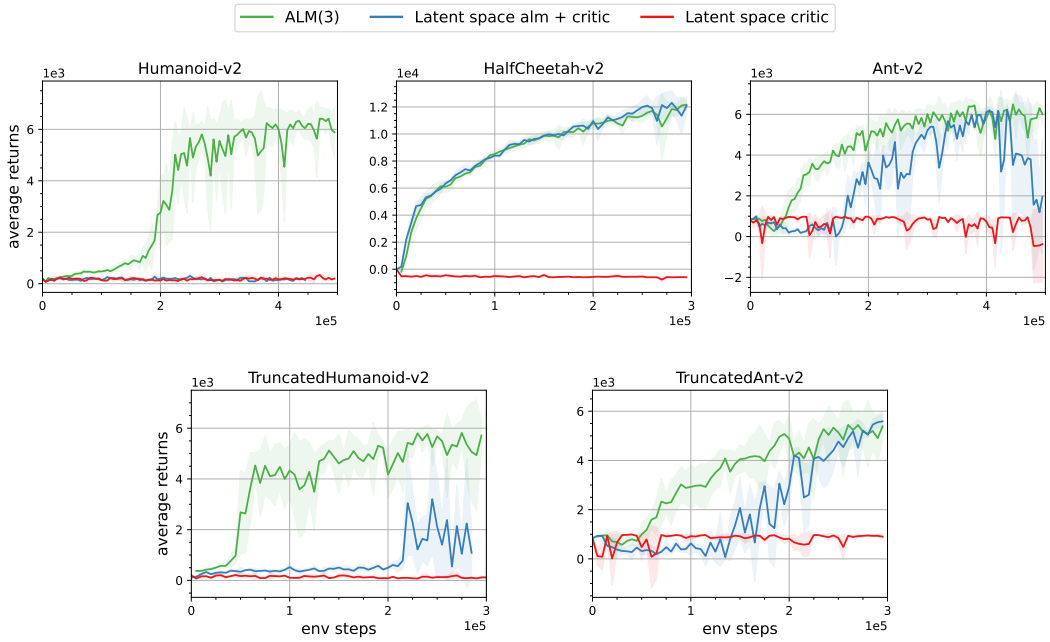


Figure 19: **Effect of learning the encoder as well as latent-space model to predict the value function.**

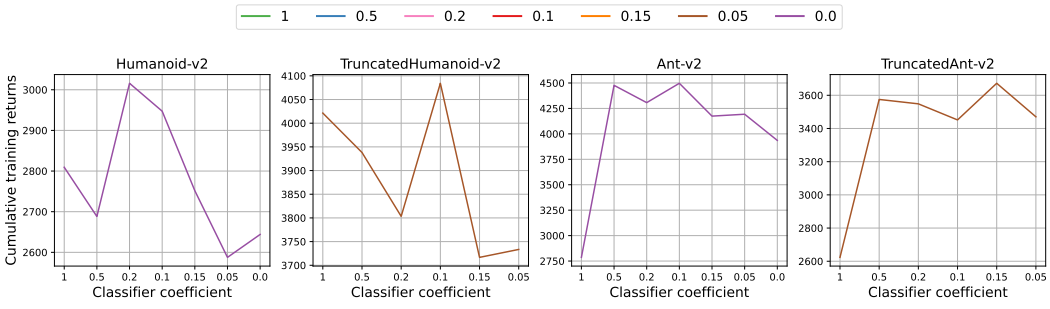


Figure 20: **Returns at different values for classifier coefficient.**

F Comparison to baselines

Here we provide a quick comparison to baselines, conceptually in terms of number of gradient updates per env step, usage of ensembles are used for dynamics model or Q-function, and objectives for representations, model, and policy.

Table 5: Table showing conceptual comparisons of ALM to baselines.

	Ensembles	Representations	Model	Policy	UTD
ALM	✗	Lower bound	Lower bound	Lower bound	3
SAC-SVG	✗	MLE	MLE	Actor-Critic (stochastic value gradients)	4
MBPO	✓	✗	MLE	Soft Actor-Critic (dyna-style)	20-40
REDQ	✓	✗	✗	Soft Actor-Critic	20