CONTROL-AWARE REPRESENTATIONS FOR MODEL-BASED REINFORCEMENT LEARNING

Brandon Cui
Facebook AI Research
bcui@fb.com

Yinlam Chow
Google Research
yinlamchow@google.com

Mohammad Ghavamzadeh
Google Research
ghavamza@google.com

ABSTRACT

A major challenge in modern reinforcement learning (RL) is efficient control of dynamical systems from high-dimensional sensory observations. Learning controllable embedding (LCE) is a promising approach that addresses this challenge by embedding the observations into a lower-dimensional latent space, estimating the latent dynamics, and utilizing it to perform control in the latent space. Two important questions in this area are how to learn a representation that is amenable to the control problem at hand, and how to achieve an end-to-end framework for representation learning and control. In this paper, we take a few steps towards addressing these questions. We first formulate a LCE model to learn representations that are suitable to be used by a policy iteration style algorithm in the latent space. We call this model *control-aware representation learning* (CARL). We derive a loss function and three implementations for CARL. In the *offline* implementation, we replace the locally-linear control algorithm (e.g., iLQR) used by the existing LCE methods with a RL algorithm, namely model-based soft actor-critic, and show that it results in significant improvement. In *online* CARL, we interleave representation learning and control, and demonstrate further gain in performance. Finally, we propose *value-guided* CARL, a variation in which we optimize a weighted version of the CARL loss function, where the weights depend on the TD-error of the current policy. We evaluate the proposed algorithms by extensive experiments on benchmark tasks and compare them with several LCE baselines.

1 INTRODUCTION

Control of non-linear dynamical systems is a key problem in control theory. Many methods have been developed with different levels of success in different classes of such problems. The majority of these methods assume that a model of the system is known and its underlying state is low-dimensional and observable. These requirements limit the usage of these techniques in controlling dynamical systems from high-dimensional raw sensory data (e.g., image), where the system dynamics is unknown, a scenario often seen in modern reinforcement learning (RL).

Recent years have witnessed a rapid development of a large arsenal of model-free RL algorithms, such as DQN (Mnih et al., 2013), TRPO (Schulman et al., 2015), PPO (Schulman et al., 2017), and SAC (Haarnoja et al., 2018), with impressive success in solving high-dimensional control problems. However, most of this success has been limited to simulated environments (e.g., computer games), mainly due to the fact that these algorithms often require a large number of samples from the environment. This restricts their applicability in real-world physical systems, for which data collection is often a difficult process. On the other hand, model-based RL algorithms, such as PILCO (Deisenroth & Rasmussen, 2011), MBPO (Janner et al., 2019), and Visual Foresight (Ebert et al., 2018), despite their success, still face difficulties in learning a model (dynamics) in a high-dimensional (pixel) space.

To address the problems faced by model-free and model-based RL algorithms in solving high-dimensional control problems, a class of algorithms have been developed, whose main idea is to first learn a low-dimensional latent (embedding) space and a latent model (dynamics), and then use this model to control the system in the latent space. This class has been referred to as *learning controllable embedding* (LCE) and includes algorithms, such as E2C (Watter et al., 2015), RCE (Banijamali et al., 2018), SOLAR (Zhang et al., 2019), PCC (Levine et al., 2020), Dreamer (Hafner et al., 2020a;b), PC3 (Shu et al., 2020), and SLAC (Lee et al., 2020). The following two properties are extremely important in designing LCE models and algorithms. **First**, to learn a representation that is the most suitable for the control problem at hand. This suggests incorporating the control algorithm in the process of learning representation. This view of learning control-aware representations is aligned with the value-aware and policy-aware model learning, VAML (Farahmand, 2018) and PAML (Abachi et al., 2020), frameworks that have been recently proposed in model-based RL. **Second**, to interleave the representation learning and control, and to update them both, using a unifying objective function. This allows to have an end-to-end framework for representation learning and control.

LCE methods, such as SOLAR, Dreamer, and SLAC, have taken steps towards the second objective by performing representation learning and control in an online fashion. This is in contrast to offline methods like E2C, RCE, PCC, and PC3 that learn a representation once and then use it in the entire control process. On the other hand, methods like PCC and PC3 address the first objective by adding a term to their representation learning loss function that accounts for the curvature of the latent dynamics. This term regularizes the representation towards smoother latent dynamics, which are suitable for the locally-linear controllers, e.g., iLQR (Li & Todorov, 2004), used by these methods.

In this paper, we take a few steps towards the above two objectives. We first formulate a LCE model to learn representations that are suitable to be used by a policy iteration (PI) style algorithm in the latent space. We call this model *control-aware representation learning* (CARL) and derive a loss function for it that exhibits a close connection to the prediction, consistency, and curvature (PCC) principle for representation learning (Levine et al., 2020). We derive three implementations of CARL: *offline*, *online*, and *value-guided*. Similar to offline LCE methods, such as E2C, RCE, PCC, and PC3, in *offline* CARL, we first learn a representation and then use it in the entire control process. However, in offline CARL, we replace the locally-linear control algorithm (e.g., iLQR) used by these LCE methods with a PI-style (actor-critic) RL algorithm. Our choice of RL algorithm is the model-based implementation of soft actor-critic (SAC) (Haarnoja et al., 2018). Our experiments show significant performance improvement by replacing iLQR with SAC. *Online* CARL is an iterative algorithm in which at each iteration, we first learn a latent representation by minimizing the CARL loss, and then perform several policy updates using SAC in this latent space. Our experiments with online CARL show further performance gain over its offline version. Finally, in *value-guided* CARL (V-CARL), we optimize a weighted version of the CARL loss function, in which the weights depend on the TD-error of the current policy. This would help to further incorporate the control algorithm in the representation learning process. We evaluate the proposed algorithms by extensive experiments on benchmark tasks and compare them with several LCE baselines: PCC, SOLAR, and Dreamer.

2 Problem Formulation

We are interested in learning control policies for non-linear dynamical systems, where the states $s \in \mathcal{S} \subseteq \mathbb{R}^{n_s}$ are not fully observed and we only have access to their high-dimensional observations $x \in \mathcal{X} \subseteq \mathbb{R}^{n_x},\ n_x \gg n_s$. This scenario captures many practical applications in which we interact with a system only through high-dimensional sensory signals, such as image and audio. We assume that the observations x have been selected such that we can model the system in the observation space using a Markov decision process (MDP)[1] $\mathcal{M}_\mathcal{X} = \langle \mathcal{X}, \mathcal{A}, r, P, \gamma \rangle$, where $\mathcal{X}$ and $\mathcal{A}$ are observation and action spaces; $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the reward function with maximum value $R_{\max}$, defined by the designer of the system to achieve the control objective;[2] $P : \mathcal{X} \times \mathcal{A} \to \mathbb{P}(\mathcal{X})$ is the unknown transition kernel; and $\gamma \in (0, 1)$ is the discount factor. Our goal is to find a mapping from observations to control signals, $\mu : \mathcal{X} \to \mathbb{P}(\mathcal{A})$, with maximum expected return, i.e., $J(\mu) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) \mid P, \mu]$.

Since the observations x are high-dimensional and the observation dynamics P is unknown, solving the control problem in the observation space may not be efficient. As discussed in Section 1, the class of *learning controllable embedding* (LCE) algorithms addresses this by learning a low-dimensional latent (embedding) space $\mathcal{Z} \subseteq \mathbb{R}^{n_z},\ n_z \ll n_x$, together with a latent dynamics, and controlling the system there. The main idea behind LCE is to learn an *encoder* $E : \mathcal{X} \to \mathbb{P}(\mathcal{Z})$, a *latent space dynamics* $F : \mathcal{Z} \times \mathcal{A} \to \mathbb{P}(\mathcal{Z})$, and a *decoder* $D : \mathcal{Z} \to \mathbb{P}(\mathcal{X})$,[3] such that a good or optimal controller (policy) in $\mathcal{Z}$ performs well in the observation space $\mathcal{X}$. This means that if we model the control problem in $\mathcal{Z}$ as a MDP $\mathcal{M}_\mathcal{Z} = \langle \mathcal{Z}, \mathcal{A}, \bar{r}, F, \gamma \rangle$ and solve it using a model-based RL algorithm to obtain a policy $\pi : \mathcal{Z} \to \mathbb{P}(\mathcal{A})$, the image of π back in the observation space, i.e.,

[1]A method to ensure observations are Markovian is to buffer them for several time steps (Mnih et al., 2013).

[2]For example, in a goal tracking problem in which the agent (robot) aims at finding the shortest path to reach the observation goal x_g (the observation corresponding to the goal state s_g), we may define the reward for each observation x as the negative of its distance to x_g, i.e., $-\|x - x_g\|^2$.

[3]Some recent LCE models, such as PC3 (Shu et al., 2020), are advocating latent models without a decoder. Although we are aware of the merits of such approach, we use a decoder in the models proposed in this paper.

Algorithm 1 Latent Space Learning with Policy Iteration (LSLPI)

```
1: Inputs: $E^{(0)}, F^{(0)}, D^{(0)}$;
2: Initialization: $\mu^{(0)}$ = random policy;    $\mathcal{D} \leftarrow$ samples generated from $\mu^{(0)}$;
3: for $i = 0, 1, \ldots$ do
4:     Compute $\pi^{(i)}$ as the projection of $\mu^{(i)}$ in the latent space w.r.t. $D_{\mathrm{KL}}(\pi \circ E \,||\, \mu)$;  # $\mu^{(i)} \approx \pi^{(i)} \circ E^{(i)}$
5:     Compute the value function of $\pi^{(i)}$ and set $V^{(i)} = V_{\pi^{(i)}}$;    # policy evaluation (critic)
6:     Compute the greedy policy w.r.t. $V^{(i)}$ and set $\pi_+^{(i)} = \mathcal{G}[V^{(i)}]$;    # policy improvement (actor)
7:     Set $\mu^{(i+1)} = \pi_+^{(i)} \circ E^{(i)}$;    # project the improved policy $\pi_+^{(i)}$ back into the observation space
8:     Learn $(E^{(i+1)}, F^{(i+1)}, D^{(i+1)}, \bar{r}^{(i+1)})$ from $\mathcal{D}$, $\pi^{(i)}$, and $\pi_+^{(i)}$;    # representation learning
9:     Generate samples $\mathcal{D}^{(i+1)} = \{(x_t, a_t, r_t, x_{t+1})\}_{t=1}^n$ from $\mu^{(i+1)}$;    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}^{(i+1)}$;
10: end for
```

$(\pi \circ E)(a|x) = \int_z dE(z|x)\pi(a|z)$, should have high expected return. Thus, the loss function to learn $\mathcal{Z}$ and (E, F, D) from observations $\{(x_t, a_t, r_t, x_{t+1})\}$ should be designed to comply with this goal.

This is why in this paper, we propose a LCE framework that tries to incorporate the control algorithm used in the latent space in the representation learning process. We call this model, *control-aware representation learning* (CARL). In CARL, we set the class of control (RL) algorithms used in the latent space to approximate policy iteration (PI), and more specifically to soft actor-critic (SAC) (Haarnoja et al., 2018). Before describing CARL in details in the following sections, we present a number of useful definitions and notations here.

For any policy μ in $\mathcal{X}$, we define its value function U_μ and Bellman operator T_μ as

$$U_\mu(x) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_\mu(x_t) \mid P_\mu, x_0 = x], \qquad T_\mu[U](x) = \mathbb{E}_{x' \sim P_\mu(\cdot|x)}[r_\mu(x) + \gamma U(x')], \tag{1}$$

for all $x \in \mathcal{X}$ and $U : \mathcal{X} \to \mathbb{R}$, where $r_\mu(x) = \int_a d\mu(a|x) r(x,a)$ and $P_\mu(x'|x) = \int_a d\mu(a|x) P(x'|x,a)$ are the reward function and dynamics induced by μ. Similarly, for any policy π in $\mathcal{Z}$, we define its induced reward function and dynamics as $\bar{r}_\pi(z) = \int_a d\pi(a|z)\bar{r}(z,a)$ and $F_\pi(z'|z) = \int_a d\pi(a|z) F(z'|z,a)$. We also define its value function V_π and Bellman operator T_π as

$$V_\pi(z) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \bar{r}_\pi(z_t) \mid F_\pi, z_0 = z], \qquad T_\pi[V](z) = \mathbb{E}_{z' \sim F_\pi(\cdot|z)}[\bar{r}_\pi(z) + \gamma V(z')]. \tag{2}$$

For any policy π and value function V in the latent space $\mathcal{Z}$, we denote by $\pi \circ E$ and $V \circ E$, their image in the observation space $\mathcal{X}$, given encoder E, and define them as

$$(\pi \circ E)(a|x) = \int_z dE(z|x)\pi(a|z), \qquad (V \circ E)(x) = \int_z dE(z|x) V(z). \tag{3}$$

3 CARL Model: A Control Perspective

In this section, we formulate our LCE model, which we refer to as *control-aware representation learning* (CARL). As described in Section 2, CARL is a model for learning a low-dimensional latent space $\mathcal{Z}$ and the latent dynamics, from data generated in the observation space $\mathcal{X}$, such that this representation is suitable to be used by a policy iteration (PI) style algorithm in $\mathcal{Z}$. In order to derive the loss function used by CARL to learn $\mathcal{Z}$ and its dynamics, i.e., $(E, F, D, \bar{r})$, we first describe how the representation learning can be interleaved with PI in $\mathcal{Z}$. Algorithm 1 contains the pseudo-code of the resulting algorithm, which we refer to as *latent space learning policy iteration* (LSLPI).

Each iteration i of LSLPI starts with a policy $\mu^{(i)}$ in the observation space $\mathcal{X}$, which is the mapping of the improved policy in $\mathcal{Z}$ in iteration $i-1$, i.e., $\pi_+^{(i-1)}$, back in $\mathcal{X}$ through the encoder $E^{(i-1)}$ (Lines 6 and 7). We then compute $\pi^{(i)}$, the current policy in $\mathcal{Z}$, as the image of $\mu^{(i)}$ in $\mathcal{Z}$ through the encoder $E^{(i)}$ (Line 4). Note that $E^{(i)}$ is the encoder learned at the end of iteration $i-1$ (Line 8). We then use the latent space dynamics $F^{(i)}$ learned at the end of iteration $i-1$ (Line 8), and first compute the value function of $\pi^{(i)}$ in the policy evaluation or *critic* step, i.e., $V^{(i)} = V_{\pi^{(i)}}$ (Line 5), and then use $V^{(i)}$ to compute the improved policy $\pi_+^{(i)}$, as the greedy policy w.r.t. $V^{(i)}$, i.e., $\pi^{(i+1)} = \mathcal{G}[V^{(i)}]$, in the policy improvement or *actor* step (Line 6). Using the samples in the buffer $\mathcal{D}$, together with the current policies in $\mathcal{Z}$, i.e., $\pi^{(i)}$ and $\pi_+^{(i)}$, we learn the new representation $(E^{(i+1)}, F^{(i+1)}, D^{(i+1)}, \bar{r}^{(i+1)})$ (Line 8). Finally, we generate samples $\mathcal{D}^{(i+1)}$ by following $\mu^{(i+1)}$, the image of the improved policy $\pi_+^{(i)}$ back in $\mathcal{X}$ using the old encoder $E^{(i)}$ (Line 7), and add it to the buffer $\mathcal{D}$ (Line 9), and the algorithm iterates. It is important to note that both critic and actor operate in the low-dimensional latent space $\mathcal{Z}$.

LSLPI is a PI algorithm in $\mathcal{Z}$. However, what is desired is that it also acts as a PI algorithm in $\mathcal{X}$, i.e., it results in (monotonic) policy improvement in $\mathcal{X}$, i.e., $U_{\mu^{(i+1)}} \geq U_{\mu^{(i)}}$. Therefore, we define the representation learning loss function for CARL, such that it ensures LSLPI also results in policy improvement in $\mathcal{X}$. The following theorem, whose proof is reported in Appendix A, shows the relationship between the value functions of two consecutive polices generated by LSLPI in $\mathcal{X}$.

Theorem 1. *Let μ, μ_+, π, π_+, and $(E, F, D, \bar{r})$ be the policies $\mu^{(i)}$, $\mu^{(i+1)}$, $\pi^{(i)}$, $\pi_+^{(i)}$, and the learned latent representation $(E^{(i+1)}, F^{(i+1)}, D^{(i+1)}, \bar{r}^{(i+1)})$ at iteration i of the LSLPI algorithm (Algorithm 1). Then, the following holds for the value functions of μ and μ_+:*

$$\begin{aligned} U_{\mu_+}(x) \geq U_\mu(x) - \Big(& \frac{1}{1-\gamma} \sum_{\widetilde{\pi} \in \{\pi, \pi_+\}} \mathbb{E}_{d^\gamma_{\widetilde{\pi}\circ E}} \left[\Delta(E, F, D, \bar{r}, \widetilde{\pi}, \cdot) | x_0 = x\right] \\ & + \frac{\sqrt{2}\gamma R_{\max}}{1-\gamma} \cdot \mathbb{E}_{d^\gamma_{\pi\circ E}} \big[\underbrace{\sqrt{D_{KL}\big((\pi \circ E)(\cdot'|\cdot) \;||\; \mu(\cdot'|\cdot)\big)}}_{\mathrm{L_{reg}(E,\mu,\pi,\cdot)}} | x_0 = x \big] \Big), \end{aligned} \tag{4}$$

for all $x \in \mathcal{X}$, where $d^\gamma_{\pi\circ E}(x'|x_0) = (1-\gamma) \cdot \sum_{\ell=0}^\infty \gamma^\ell \mathbb{P}(x_\ell = x'|x_0; \pi \circ E)$ is the γ-stationary distribution induced by policy $\pi \circ E$, and the error term Δ for a policy π is given by

$$\begin{aligned} \Delta(E, F, D, \bar{r}, \pi, x) = & \frac{R_{\max}}{1-\gamma} \overbrace{\sqrt{\frac{-1}{2} \int_z dE(z|x) \log D(x|z)}}^{\mathrm{(I)=L_{ed}(E,D,x)}} + \overbrace{2 \left| r_{\pi\circ E}(x) - \int_z dE(z|x) \bar{r}_\pi(z) \right|}^{\mathrm{(II)=L_r(E,\bar{r},\pi,x)}} \\ & + \frac{\gamma R_{\max}}{\sqrt{2}(1-\gamma)} \Big(\underbrace{\sqrt{D_{KL}\big(P_{\pi\circ E}(\cdot|x) \;||\; (D \circ F_\pi \circ E)(\cdot|x)\big)}}_{\mathrm{(III)=L_p(E,F,D,\pi,x)}} + \underbrace{\sqrt{D_{KL}\big((E \circ P_{\pi\circ E})(\cdot|x) \;||\; (F_\pi \circ E)(\cdot|x)\big)}}_{\mathrm{(IV)}} \Big). \end{aligned} \tag{5}$$

It is easy to see that LSLPI guarantees (policy) improvement in $\mathcal{X}$, if the terms in the parentheses on the RHS of (4) are zero. We now describe these terms. The last term on the RHS of (4) is the KL between $\pi^{(i)} \circ E$ and $\mu^{(i)} = \pi^{(i)} \circ E^{(i)}$. This term can be seen as a regularizer to keep the new encoder E close to the old one $E^{(i)}$. The four terms in (5) are: **(I)** The encoding-decoding error to ensure $x \approx (D \circ E)(x)$; **(II)** The error that measures the mismatch between the reward of taking action according to policy $\pi \circ E$ at $x \in \mathcal{X}$, and the reward of taking action according to policy π at the image of x in $\mathcal{Z}$ under E; **(III)** The error in predicting the next observation through paths in $\mathcal{X}$ and $\mathcal{Z}$. This is the error between x' and $\hat{x}'$ shown in Fig. 1(a); and **(IV)** The error in predicting the next latent state through paths in $\mathcal{X}$ and $\mathcal{Z}$. This is the error between z' and $\tilde{z}'$ shown in Fig. 1(b).

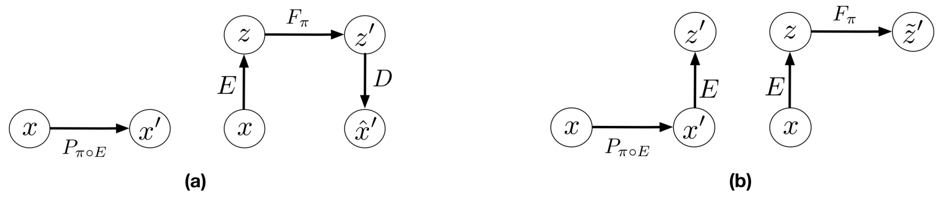


Figure 1: **(a)** Paths from the current observation x to the next one, (left) in $\mathcal{X}$ and (right) through $\mathcal{Z}$. **(b)** Paths from the current observation x to the next latent state, (left) through $\mathcal{X}$ followed by encoding and (right) starting with encoding and then through $\mathcal{Z}$.

Representation Learning in CARL Theorem 1 provides us with a recipe (loss function) to learn the latent space $\mathcal{Z}$ and $(E, F, D, \bar{r})$. In CARL, we propose to learn a representation for which the terms in the parentheses on the RHS of (4) are small. As mentioned earlier, the second term,

$L_{\text{reg}}(E, \mu, \pi, x)$, can be considered as a regularizer to keep the new encoder E close to the old one E_-, when the policy μ is given by $\pi \circ E_-$. Term (I) minimizes the reconstruction error between encoder and decoder, which is standard for training auto-encoders (Kingma & Welling, 2013). Term (II) that measures the mismatch between rewards can be kept small, or even zero, if the designer of the system selects the rewards in a compatible way[4]. Although CARL allows us to learn a reward function in the latent space, similar to several other LCE works (Watter et al., 2015; Banijamali et al., 2018; Levine et al., 2020; Shu et al., 2020), in this paper, we assume that a compatible latent reward function is given. Terms (III) and (IV) are the equivalent of the *prediction* and *consistency* terms in PCC (Levine et al., 2020) for a particular latent space policy π. Since PCC has been designed for an offline setting (i.e., one-shot representation learning and control), its prediction and consistency terms are independent of a particular policy and are defined for state-action pairs. While CARL is designed for an online setting (i.e., interleaving representation learning and control), and thus, its loss function at each iteration depends on the current latent space policies π and π_+. As we will see in Section 4, in our offline implementation of CARL, these two terms are similar to prediction and consistency terms in PCC. Note that (IV) is slightly different than the consistency term in PCC. However, if we upper-bound it using Jensen inequality: (IV) $\leq L_c(E, F, \pi, x) := \int_{x' \in \mathcal{X}} dP_{\pi \circ E}(x'|x) \cdot D_{\text{KL}}\big(E(\cdot|x') \,||\, (F_\pi \circ E)(\cdot|x)\big)$, the resulted loss, $L_{\text{c}}(E, F, \pi, x)$, would be similar to the consistency term in PCC. Similar to PCC, we also add a curvature loss to the loss function of CARL to encourage having a smoother latent space dynamics F_π. Putting all these terms together, we obtain the following loss function for CARL:

$$\begin{aligned} \min_{E,F,D} \sum_{x \sim \mathcal{D}} & \lambda_{\text{ed}} L_{\text{ed}}(E, D, x) + \lambda_{\text{p}} L_{\text{p}}(E, F, D, \pi, x) + \lambda_{\text{c}} L_{\text{c}}(E, F, \pi, x) \\ & + \lambda_{\text{cur}} L_{\text{cur}}(F, \pi, x) + \lambda_{\text{reg}} L_{\text{reg}}(E, \mu, \pi, x), \end{aligned} \tag{6}$$

where $(\lambda_{\text{ed}}, \lambda_{\text{p}}, \lambda_{\text{c}}, \lambda_{\text{cur}}, \lambda_{\text{reg}})$ are hyper-parameters[5] of the algorithm, $(L_{\text{ed}}, L_{\text{p}})$ are the encoding-decoding and prediction losses defined in (5), L_{c} is the consistency loss defined above, $L_{\text{cur}} = \mathbb{E}_{x,u}[\mathbb{E}_\epsilon\big[f_{\mathcal{Z}}(z+\epsilon_z, u+\epsilon_u) - f_{\mathcal{Z}}(z,u) - (\nabla_z f_{\mathcal{Z}}(z,u) \cdot \epsilon_z + \nabla_u f_{\mathcal{Z}}(z,u) \cdot \epsilon_u)\|_2^2\big] \mid E]$ is the curvature loss that regulates the 2nd derivative of $f_{\mathcal{Z}}$, the mean of latent dynamics F, in which ϵ_z, ϵ_u are standard Gaussian noise, and L_{reg} is the regularizer that ensures the new encoder remains close to the old one.

4 Different Implementations of CARL

The CARL loss function in (6) introduces an optimization problem that takes a policy π in $\mathcal{Z}$ as input and learns a representation suitable for its evaluation and improvement. To optimize this loss in practice, similar to the PCC model (Levine et al., 2020), we define $\widehat{P} = D \circ F_\pi \circ E$ as a latent variable model that is factorized as $\widehat{P}(x_{t+1}, z_t, \hat{z}_{t+1}|x_t, \pi) = \widehat{P}(z_t|x_t)\widehat{P}(\hat{z}_{t+1}|z_t, \pi)\widehat{P}(x_{t+1}|\hat{z}_{t+1})$, and use a variational approximation to the interactable negative log-likelihood of the loss terms in (6). The variational bounds for these terms can be obtained similar to Eqs. 6 and 7 in Levine et al. (2020). Below we describe three instantiations of the CARL model in practice. Implementation details can be found in Algorithm 2 in Appendix D. Although CARL is compatible with most PI-style (actor-critic) RL algorithms, we choose soft actor-critic (SAC) (Haarnoja et al., 2018) as its control algorithm. Since most actor-critic algorithms are based on first-order gradient updates, as discussed in Section 3, we regularize the curvature of the latent dynamics F (see Eqs. 8 and 9 in Levine et al. 2020) in CARL to improve its empirical stability and performance in policy learning.

1. Offline CARL We first implement CARL in an offline setting, where we generate a (relatively) large batch of observation samples $\{(x_t, a_t, r_t, x_{t+1})\}_{t=1}^N$ using an exploratory (e.g., random) policy. We then use this batch to optimize the CARL's loss function (6) via the variational approximation scheme described above, and learn a latent representation $\mathcal{Z}$ and (E, F, D). Finally, we solve the decision problem in $\mathcal{Z}$ using a model-based RL algorithm, which in our case is model-based SAC[6]. The learned policy $\hat{\pi}^*$ in $\mathcal{Z}$ is then used to control the system from observations as $a_t \sim (\hat{\pi}^* \circ E)(\cdot|x_t)$. This is the setting that has been used in several recent LCE works, such as E2C (Watter et al., 2015), RCE (Banijamali et al., 2018), PCC (Levine et al., 2020), and PC3 (Shu et al., 2020). Our offline

[4]For example, in goal-based RL problems, a compatible reward function can be the one that measures the negative distance between a latent state and the image of the goal in the latent space.

[5]Theorem 1 provides a high-level guideline for selecting the hyper-parameters of the loss function: $\lambda_{\text{ed}} = 2R_{\max}/(1-\gamma)^2$, $\lambda_{\text{c}} = \lambda_{\text{p}} = \sqrt{2}\gamma R_{\max}/(1-\gamma)^2$, and $\lambda_{\text{reg}} = \sqrt{2}\gamma R_{\max}/(1-\gamma)$.

[6]By model-based SAC, we refer to learning a latent policy with SAC using synthetic trajectories generated by unrolling the learned latent dynamics model F, similar to the MBPO algorithm (Janner et al., 2019).

implementation is different than those in which **1)** we replace their locally-linear control algorithm, namely iterative LQR (iLQR) (Li & Todorov, 2004), with model-based SAC, which results in significant performance improvement, as shown in Section 5, and **2)** we optimize the CARL loss function, that despite close connection, is still different than the one used by PCC.

The CARL loss function presented in Section 3 has been designed for an online setting in which at each iteration, it takes a policy as input and learns a representation that is suitable for evaluating and improving this policy. However, in the offline setting, the learned representation should be good for any policy generated in the course of running the PI-style control algorithm. Therefore, we marginalize out the policy from the (online) CARL's loss function and use the RHS of the following corollary (proof in Appendix B) to construct the CARL loss function used in our offline experiments.

Corollary 2. *Let μ and μ_+ be two consecutive policies in $\mathcal{X}$ generated by a PI-style control algorithm in the latent space constructed by $(E,F,D,\bar{r})$. Then, the following holds for the value functions of μ and μ_+, where Δ is defined by (5) (in modulo replacing sampled action $a\sim\pi\circ E$ with action a):*

$$U_{\mu_+}(x) \geq U_\mu(x) - \frac{2}{1-\gamma}\cdot \max_{x,\in\mathcal{X},a\in\mathcal{A}} \Delta(E,F,D,\bar{r},a,x),\ \forall x\in\mathcal{X}. \tag{7}$$

2. Online CARL In the online implementation of CARL, at each iteration i, the current policy $\pi^{(i)}$ is the improved policy of the last iteration, $\pi_+^{(i-1)}$. We first generate a relatively (to offline CARL) small batch of samples using the image of the current policy in $\mathcal{X}$, i.e., $\mu^{(i)} = \pi^{(i)}\circ E^{(i-1)}$, and then learn a representation $(E^{(i)},F^{(i)},D^{(i)})$ suitable for evaluating and improving the image of $\mu^{(i)}$ in $\mathcal{Z}$ under the new encoder $E^{(i)}$. This means that with the new representation, the current policy that was the image of $\mu^{(i)}$ in $\mathcal{Z}$ under $E^{(i-1)}$, should be replaced by its image $\pi^{(i)}$ under the new encoder, i.e., $\pi^{(i)}\circ E^{(i)}\approx\mu^{(i)}$. In online CARL, we address this by the following *policy distillation* step in which we minimize the following loss:[7]

$$\pi^{(i)} \in \arg\min_\pi \sum_{x\sim\mathcal{D}} D_{\text{KL}}\big((\pi\circ E^{(i)})(\cdot|x)\ ||\ (\pi_+^{(i-1)}\circ E^{(i-1)})(\cdot|x)\big). \tag{8}$$

After the current policy $\pi^{(i)}$ is set, we perform multiple steps of (model-based) SAC in $\mathcal{Z}$ using the current model, $(F^{(i)},\bar{r}^{(i)})$, and then send the resulting policy $\pi_+^{(i)}$ to the next iteration.

3. Value-Guided CARL (V-CARL) While Theorem 1 shows that minimizing the loss in (6) guarantees performance improvement, this loss does not contain any information about the performance of the current policy μ, and thus, the LCE model trained with this loss may have low accuracy in regions of the latent space that are crucial for learning good RL policies. In V-CARL, we tackle this issue by modifying the loss function in a way that the resulted LCE model has more accuracy in regions with higher anticipated future returns.

To derive the V-CARL's loss function, we use the variational model-based policy optimization (VMBPO) framework by Chow et al. (2020) in which the *optimal* dynamics for model-based RL can be expressed in closed-form as $P^*(x'|x,a) = P(x'|x,a)\cdot\exp\big(\frac{\tau}{\gamma}(r(x,a)+\gamma\tilde{U}_\mu(x') - \tilde{W}_\mu(x,a))\big)$, where $\tilde{U}_\mu(x) := \frac{1}{\tau}\log\mathbb{E}\left[\exp\left(\tau\sum_{t=0}^\infty\gamma^t r_{\mu,t}\right)|P_\mu,x_0=x\right]$ and $\tilde{W}_\mu(x,a) := r(x,a) + \frac{\gamma}{\tau}\log\mathbb{E}_{x'\sim P(\cdot|x,a)}\left[\exp(\tau U_\mu(x'))\right]$ are the *optimistic* value and action-value functions[8] of policy μ, and $\tau>0$ is a temperature parameter. Note that in the VMBPO framework, the optimal dynamics P^* is value-aware, because it re-weighs P with an *exponential-twisting* weight $\exp(\frac{\tau}{\gamma}w(x,a,x'))$, where $w(x,a,x') := r(x,a)+\gamma\tilde{U}_\mu(x')-\tilde{W}_\mu(x,a)$ is the temporal difference (TD) error.

In V-CARL, we use the VMBPO framework to modify the CARL's prediction loss $L_{\text{p}}(E,F,D,\pi,x)$. Since the regularizer loss $L_{\text{reg}}(E,\mu,\pi,x)$ in CARL forces policies $\pi\circ E$ and μ to be close to each other, we may replace the transition dynamics $P_{\pi\circ E}$ with P_μ in L_{p}. This makes minimizing L_p equivalent to maximizing the log-likelihood $\int_{x'} dP_\mu(x'|x)\cdot\log(D\circ F_\pi\circ E)(x'|x)$. Finally, we replace P_μ with P_μ^* in this log-likelihood and obtain $\int_a d\mu(a|x)\int_{x'} dP(x'|x,a)\cdot\exp(\frac{\tau}{\gamma}\cdot w(x,a,x'))\cdot\log(D\circ F_\pi\circ E)(x'|x)$, which is a weighted (by the exponential TD $w(x,a,x')$) log-likelihood function (w.r.t. P). Note

[7]Our experiments reported in Appendix F.1 show that adding distillation improves the performance in online CARL. Thus, all our results for online CARL and V-CARL, unless mentioned, are with policy distillation.

[8]We refer to $\tilde{U}_\mu$ as the *optimistic* value function (Ruszczyński & Shapiro, 2006), because it models the right tail of the return via the exponential utility $\rho_\tau(U(\cdot)|x,a) = \frac{1}{\tau}\log\mathbb{E}_{x'\sim P(\cdot|x,a)}[\exp(\tau\cdot U(x'))]$.

that this weight depends on the optimistic value functions $\tilde{U}_\mu$ and $\tilde{W}_\mu$. When $\tau > 0$ is small (see Appendix C for more details), these value functions can be approximated by their standard counterparts, i.e., $\tilde{U}_\mu(x) \approx U_\mu(x)$ and $\tilde{W}_\mu(x,a) \approx W_\mu(x,a) := r(x,a) + \int_{x'} dP(x'|x,a)U_\mu(x')$, which can be further approximated by their latent-space counterparts, i.e., $U_\mu(x) \approx (V_\pi \circ E)(x)$ and $W_\mu(x,a) \approx (Q_\pi \circ E)(x,a)$, according to Lemma 5 in Appendix A.1. Since the latent reward function $\bar{r}$ is defined such that $r(x,a) \approx (\bar{r} \circ E)(x,a)$, we may write the TD-error $w(x,a,x')$ in terms of the encoder E and the latent value functions as $\widehat{w}(x,a,x') := \int_{z,z'} dE(z|x) \cdot dE(z'|x') \cdot (\bar{r}(z,a) - Q_\pi(z,a) + \gamma V_\pi(z'))$.

5 Experimental Results

In this section, we experiment with the following continuous control domains: (i) Planar System, (ii) Inverted Pendulum (Swingup), (iii) Cartpole, (iv) Three-link Manipulator (3-Pole), and compare the performance of our CARL algorithms with three LCE baselines: PCC (Levine et al., 2020), SOLAR (Zhang et al., 2019), SLAC (Lee et al., 2020), and two implementations of Dreamer (Hafner et al., 2020a) (described below).[9] These tasks have underlying start and goal states that are "not" observable, instead, the algorithms only have access to the start and goal observations. We report the detailed setup of the experiments in Appendix E, in particular, the description of the domains in Appendix E.1 and the implementation of the algorithms in Appendix E.3.

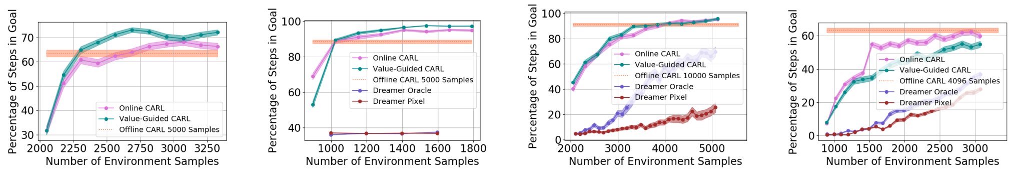
To evaluate the performance of the algorithms, similar to Levine et al. (2020), we report the %-time spent in the goal. The initial policy that is used for data generation is uniformly random (see Appendix E.2 for more details). To measure performance reproducibility for each experiment, we (i) train 25 models, and (ii) perform 10 control tasks for each model. For SOLAR, due to its high computation cost, we only train and evaluate 10 different models. Besides the average results, we also report the results from the best LCE models, averaged over the 10 control tasks.

General Results Table 1 shows the means and standard errors of %-time spent in goal, averaged over all models and control tasks, and averaged over all control tasks for the best model. To compare data efficiency, we also report the number of samples required to train the latent space and controller in each algorithm. We also show the training curves (performance vs. number of samples) of the algorithms in Fig. 2. We report more experiments and ablation studies in Appendix F.

Below summarizes our main observations of the experiments. **First,** offline CARL that uses model-based SAC as its control algorithm achieves significantly better performance than PCC that uses iLQR in all tasks. This can be attributed to the advantage that SAC is more robust and effective in non-(locally)-linear environments. We report more detailed comparison between PCC and offline CARL in Appendix F.3, where we explicitly compare their control performance and latent representation maps. **Second,** in all tasks, online CARL is more data-efficient than its offline counterpart, i.e., it achieves similar or better performance with fewer samples. In particular, online CARL is notably superior in Planar, Cartpole, and Swingup, in which it achieves similar performance to offline CARL with 2, 2.5, and 4 times less samples, respectively (see Fig. 2). In Appendix F.3, we show how the latent representation of online CARL progressively improves through the iterations of the algorithm (in particular, see Fig. 11). **Third,** in the simpler tasks (Planar, Swingup, Cartpole), V-CARL performs even better than online CARL. This corroborates our hypothesis that CARL can achieve extra improvement when its LCE model is more accurate in the regions of the latent space with higher temporal difference (regions with higher anticipated future return). In 3-pole, the performance of V-CARL is worse than online CARL. This is likely due to the instability in representation learning resulted from sample variance amplification by the exponential-TD weight. **Fourth,** SOLAR requires significantly more samples to learn a reasonable latent space for control, and with limited data it fails to converge to a good policy. Even with the fine-tuned latent space from Zhang et al. (2019), its performance is incomparable to those of CARL variants and Dreamer. We report more experiments with SOLAR in Appendix F.5, in which we show that SOLAR can perform better, especially in Planar when we fix the start and goal locations. However, the improved performance is still incomparable with those of CARL and Dreamer. **Fifth,** we include an ablation study in Appendix F.2 to demonstrate how each term of the CARL's loss function impacts policy learning. It shows the importance of the prediction and consistency terms, without which the resulting algorithms struggle, and the (relatively) minor role of the curvature and encoder-decoder terms in the performance of the algorithms.

[9]We did not include E2C and RCE in our experiments, because Levine et al. (2020) has previously shown that PCC outperforms them.

Dreamer As described in Section 2, most LCE algorithms, including E2C, PCC, and CARL variants, assume the observation space $\mathcal{X}$ is selected such that the system is Markovian there. In contrast, Dreamer does not make this assumption and has been designed for more general class of control problems that can be modeled as POMDPs. Thus, it is expected that it performs inferior (requires more samples to achieve the same performance) to CARL when the system is Markov in the observation space. Moreover, CARL and other LCE methods define the reward as the negative distance to the goal in the latent space. This cannot be done in Dreamer, where the encoder is an RNN that takes an entire observation trajectory as input. To address this, we propose two methods to train the Dreamer's reward function in the latent space, which we refer to as Dreamer Pixel and Dreamer Oracle. While Dreamer Pixel uses the negative distance to the goal in the observation space $\mathcal{X}$ as the signal to train the reward function, Dreamer Oracle uses the negative distance in the (unobserved) underlying state space $\mathcal{S}$. Thus, it is more fair to compare the CARL algorithms with Dreamer Pixel than Dreamer Oracle that has the advantage of having access to the underlying state space (see Appendix F.6 for more details). As it was expected, our results show that although both Dreamer's implementations learn reasonably-performing policies for most tasks (except Planar), they require twice to 100-times more samples to achieve the same performance as the CARL algorithms. We report longer (more samples) experiments with Dreamer on all tasks in Appendix F.6 (Fig. 12).



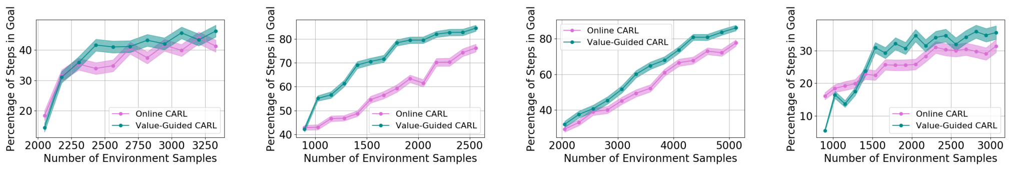
(a) Planar (b) Swingup (c) Cartpole (d) Three-pole

Figure 2: Training curves of offline CARL, online CARL, V-CARL, and two implementations of Dreamer. The shaded region represents mean $\pm$ standard error.

Environment	**Algorithm**	**Number of Samples**	**Avg %-Goal**	**Best %-Goal**
Planar	PCC	5000	38.85 ± 2.45	62.5 ± 10.42
Planar	Offline CARL	5000	63.43 ± 2.78	$\mathbf{79.51 \pm 0.38}$
Planar	Online CARL	3072	68.03 ± 1.69	79.02 ± 0.38
Planar	V-CARL	3200	$\mathbf{71.05 \pm 1.46}$	$\mathbf{79.51 \pm 0.38}$
Planar	SOLAR	5000 (VAE) + 16000 (Control)	5.82 ± 2.50	9.13 ± 3.54
Swingup	PCC	5000	86.60 ± 1.00	97.40 ± 0.61
Swingup	Offline CARL	5000	88.43 ± 2.02	$\mathbf{98.50 \pm 0.0}$
Swingup	Online CARL	1408	95.04 ± 0.96	$\mathbf{98.50 \pm 0.0}$
Swingup	V-CARL	1408	$\mathbf{96.50 \pm 0.25}$	$\mathbf{98.50 \pm 0.0}$
Swingup	SOLAR	5200 (VAE) + 40000 (Control)	16.1 ± 0.69	22.45 ± 1.96
Swingup	Dreamer Pixel	180895	70.35 ± 0.62	$\mathbf{98.5 \pm 0.0}$
Swingup	Dreamer Oracle	183084	94.65 ± 0.20	98.25 ± 0.0
Cartpole	PCC	10000	83.64 ± 0.63	$\mathbf{100.0 \pm 0.0}$
Cartpole	Offline CARL	10000	91.11 ± 1.50	$\mathbf{100.0 \pm 0.0}$
Cartpole	Online CARL	5120	95.34 ± 1.17	$\mathbf{100.0 \pm 0.0}$
Cartpole	V-CARL	5120	95.79 ± 1.06	$\mathbf{100.0 \pm 0.0}$
Cartpole	SOLAR	5000 (VAE) + 40000 (Control)	10.61 ± 2.58	12.33 ± 2.96
Cartpole	Dreamer Pixel	96941	95.59 ± 3.77	$\mathbf{100.0 \pm 0.0}$
Cartpole	Dreamer Oracle	14474	$\mathbf{97.77 \pm 1.525}$	$\mathbf{100.0 \pm 0.0}$
Three-pole	PCC	4096	4.41 ± 0.75	36.20 ± 7.06
Three-pole	Offline CARL	4096	63.20 ± 1.77	88.55 ± 0.0
Three-pole	Online CARL	2944	62.17 ± 2.28	$\mathbf{90.05 \pm 0.0}$
Three-pole	V-CARL	2816	55.06 ± 2.42	89.05 ± 0.0
Three-pole	SOLAR	2000 (VAE) + 20000 (Control)	0 ± 0	0 ± 0
Three-pole	Dreamer Pixel	6245	61.93 ± 2.30	$\mathbf{90.00 \pm 0.0}$
Three-pole	Dreamer Oracle	6245	$\mathbf{71.07 \pm 2.45}$	88.40 ± 0.0

Table 1: Mean $\pm$ standard error results (%-goal) and samples used for different LCE algorithms.

Results with Environment-biased Sampling In the previous experiments, all the online LCE algorithms are warm-started with data collected by a uniformly random policy over the entire environment. With sufficient data the latent dynamics is accurate enough on most parts of the state space for control, therefore we do not observe a significant difference between online CARL and V-CARL. To further illustrate the advantage of V-CARL over online CARL, we modify the experimental setting by gathering initial samples only from a specific region of the environment (see Appendix E.1 for more details). Fig. 3 shows the learning curves of online CARL and V-CARL in this case. As expected, with biased data, both algorithms experience a certain level of performance degradation, yet, V-CARL clearly outperforms online CARL — this verifies our conjecture that control-aware LCE models are more robust to initial data distribution and superior in policy optimization.



(a) Planar (b) Swingup (c) Cartpole (d) Three-pole

Figure 3: Training curves of Online CARL and V-CARL with environment-biased initial samples.

6 Conclusions

In this paper, we argued for incorporating control in the representation learning process and for the interaction between control and representation learning in learning controllable embedding (LCE) algorithms. We proposed a LCE model called *control-aware representation learning* (CARL) that learns representations suitable for policy iteration (PI) style control algorithms. We proposed three implementations of CARL that combine representation learning with model-based soft actor-critic (SAC), as the controller, in offline and online fashions. In the third implementation, called *value-guided* CARL, we further included the control process in representation learning by optimizing a weighted version of the CARL loss function, in which the weights depend on the TD-error of the current policy. We evaluated the proposed algorithms on benchmark tasks and compared them with several LCE baselines. The experiments show the importance of SAC as the controller and of the online implementation. Future directions include **1)** investigating other PI-style algorithms in place of SAC, **2)** developing LCE models suitable for value iteration style algorithms, and **3)** identifying other forms of bias for learning an effective embedding and latent dynamics.

References

R. Abachi, M. Ghavamzadeh, and A. Farahmand. Policy-aware model learning for policy gradient methods. *preprint arXiv:2003.00030*, 2020.

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. *arXiv preprint arXiv:1705.10528*, 2017.

E. Banijamali, R. Shu, M. Ghavamzadeh, H. Bui, and A. Ghodsi. Robust locally-linear controllable embedding. In *Proceedings of the Twenty First International Conference on Artificial Intelligence and Statistics*, pp. 1751–1759, 2018.

V. Borkar. Q-learning for risk-sensitive control. *Mathematics of operations research*, 27(2):294–311, 2002.

M. Breivik and T. Fossen. Principles of guidance-based path following in 2D and 3D. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 627–634, 2005.

Y. Chow, B. Cui, M. Ryu, and M. Ghavamzadeh. Variational model-based policy optimization. In *arXiv*, 2020.

M. Deisenroth and C. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 465–472, 2011.

F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine. Visual Foresight: Model-based deep reinforcement learning for vision-based robotic control. *preprint arXiv:1812.00568*, 2018.

A. Farahmand. Iterative value-aware model learning. In *Advances in Neural Information Processing Systems 31*, pp. 9072–9083, 2018.

A. Farahmand, A. Barreto, and D. Nikovski. Value-aware loss function for model-based reinforcement learning. In *Artificial Intelligence and Statistics*, pp. 1486–1494, 2017.

K. Furuta, M. Yamakita, and S. Kobayashi. Swing up control of inverted pendulum. In *Proceedings of International Conference on Industrial Electronics, Control and Instrumentation*, pp. 2193–2198, 1991.

S. Geva and J. Sitte. A cartpole experiment benchmark for trainable controllers. *IEEE Control Systems Magazine*, 13(5):40–51, 1993.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1861–1870, 2018.

D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020a.

D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020b.

M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems 32*, pp. 12519–12530. 2019.

D. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014.

D. Kingma and M. Welling. Auto-encoding variational bayes, 2013.

X. Lai, A. Zhang, M. Wu, and J. She. Singularity-avoiding swing-up control for underactuated three-link gymnast robot using virtual coupling between control torques. *International Journal of Robust and Nonlinear Control*, 25(2):207–221, 2015.

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

N. Levine, Y. Chow, R. Shu, A. Li, M. Ghavamzadeh, and H. Bui. Prediction, consistency, curvature: Representation learning for locally-linear control. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

W. Li and E. Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pp. 222–229, 2004.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with deep reinforcement learning. *preprint arXiv:1312.5602*, 2013.

A. Ruszczyński and A. Shapiro. Optimization of convex risk functions. *Mathematics of operations research*, 31(3):433–452, 2006.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1889–1897, 2015.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *preprint arXiv:1707.06347*, 2017.

R. Shu, T. Nguyen, Y. Chow, T. Pham, K. Than, M. Ghavamzadeh, S. Ermon, and H. Bui. Predictive coding for locally-linear control. *preprint arXiv:2003.01086*, 2020.

M. Spong. The swing up control problem for the acrobot. *IEEE Control Systems Magazine*, 15(1): 49–55, 1995.

M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems 28*, pp. 2746–2754. 2015.

M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine. SOLAR: Deep structured representations for model-based reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 7444–7453, 2019.

A Proof of Theorem 1

In this section, we prove the policy improvement bound (characterized by Theorem 1) in the observation space. First, we provide several intermediate results (Lemma 5 and Theorem 7) on approximate latent policy evaluation in Appendix A.1, which will be later utilized to prove the main theorem of policy improvement (Theorem 1) in Appendix A.2.

Theorem 1 characterizes the policy improvement bound of the CARL algorithm in the observation space. Here the main contribution is to connect the CARL algorithm, whose policy optimization is in the latent space, with the observation-space MDP performance. To the best of our knowledge, this is novel because standard policy iteration algorithm only guarantees performance on the same state space that it is applied to, while in this case we execute policy optimization in one space (the latent space $\mathcal{Z}$), while the performance guarantee needs to be in another space (the observation space $\mathcal{X}$). The main challenge here is to keep track of the error propagation due to the discrepancy between the observation-state evolution and the latent-state evolution paths when bounding the difference of the corresponding Bellman residuals in the $\mathcal{Z}$ and $\mathcal{X}$ spaces, and it is tackled by the intermediate results, namely Lemma 5 and Theorem 7, in Appendix A.1.

A.1 Approximate Latent Policy Evaluation

We first draw a connection between the approximate policy evaluation procedure in the observation and latent spaces. For any observation space value function $U : \mathcal{X} \to \mathbb{R}$, observation $x \in \mathcal{X}$, and policy μ, the μ-induced Bellman operator is defined as:

$$T_\mu[U](x) := \int_a d\mu(a|x) \int_{x'} dP(x'|x,a) \cdot \big(r(x,a) + \gamma \cdot U(x')\big).$$

Using the LCE model (E, F, D), we define the approximate μ-induced Bellman operator as

$$T'_\mu[U](x) := \int_z dE(z|x) \cdot \int_a d\pi(a|z) \cdot \int_{z'} dF(z'|z,a) \cdot \int_{x'} dD(x'|z') \cdot \big(r(x,a) + \gamma U(x')\big),$$

where the latent space policy π is defined as $\mu = \pi \circ E$. Note that $\pi \circ E$ corresponds to sampling a latent state z from the encoder $E(\cdot|x)$, followed by taking an action according to the latent space policy π, a transition in the latent space according to the latent dynamics F, and finally projecting back the resulting latent state into the observation space using the decoder D.

For any latent space value function $V : \mathcal{Z} \to \mathbb{R}$, latent state $z \in \mathcal{Z}$, and policy π, the π-induced Bellman operator is defined as

$$\mathcal{T}_\pi[V](z) := \int_a d\pi(a|z) \cdot \int_{z'} dF(z'|z,a) \cdot \big(\bar{r}(z,a) + \gamma \cdot V(z')\big),$$

where $\bar{r}$ is a latent reward function that is defined as an approximation of the observation reward function r as $|\int_a \int_z d\pi(a|z) dE(z|x)(\bar{r}(z,a) - r(x,a))| \approx 0$.

Using Equation (8) of Farahmand et al. (2017), we bound the difference between the exact and approximate μ-induced Bellman operator (the first inequality in the following expression) as

$$\begin{aligned}\big|T_\mu[U](x) - T'_\mu[U](x)\big| &\le \gamma \|U\|_\infty \cdot D_{\text{TV}}\Big(P_{\pi\circ E}(\cdot|x) || (D \circ F \circ \pi \circ E)(\cdot|x)\Big) \\ &\overset{\text{(a)}}{\le} \frac{\gamma \|U\|_\infty}{\sqrt{2}} \cdot \sqrt{D_{\text{KL}}\Big(P_{\pi\circ E}(\cdot|x) || (D \circ F \circ \pi \circ E)(\cdot|x)\Big)}, \ \forall x \in \mathcal{X},\end{aligned} \tag{9}$$

where $\|U\|_\infty = \max_{x\in\mathcal{X}} |U(x)|$ and **(a)** is by the Pinsker inequality. Note that in a γ-discounted MDP, whose immediate reward is bounded by magnitude $R_{\max}$, any value function in this MDP is bounded by $R_{\max}/(1-\gamma)$, i.e., $\|U\|_\infty \le R_{\max}/(1-\gamma)$, $\forall U$. This implies that the difference of these Bellman operators can be bounded by a prediction loss.

We can upper-bound the TV-divergence in (9) without dependency on the policy π, by considering the worst-case actions as

$$\begin{aligned}D_{\text{TV}}\Big(P_{\pi\circ E}(\cdot|x) || (D \circ F \circ \pi \circ E)(\cdot|x)\Big) &\le \max_{a\in\mathcal{A}} D_{\text{TV}}\Big(P(\cdot|x,a) || (D \circ F \circ E)(\cdot|x,a)\Big) \\ &\le \frac{1}{\sqrt{2}} \max_{a\in\mathcal{A}} \sqrt{D_{\text{KL}}\Big(P(\cdot|x,a) || (D \circ F \circ E)(\cdot|x,a)\Big)}.\end{aligned} \tag{10}$$

This upper bound corresponds to the prediction loss in PCC (Levine et al., 2020).

We now prove a lemma that relates the observation approximate Bellman residual $\big|T'_\mu[U](x) - U(x)\big|$ with the latent Bellman residual $\int_z dE(z|x)\,|\mathcal{T}_\pi[V](z) - V(z)|$ at an arbitrary observation $x \in \mathcal{X}$, where the latent policy π and value function V are related to their observation counterparts μ and U as $\mu = \pi \circ E$, i.e., $\mu(a|x) = \int_z dE(z|x)\pi(a|z)$, and $V = U \circ D$, i.e., $V(z) = \int_{\widetilde{x}} dD(\widetilde{x}|z)U(\widetilde{x})$.

Lemma 3. *For any observation $x \in \mathcal{X}$, encoder-transition-decoder tuple (E, F, D), latent policy π and value function U, the observation approximate and latent Bellman residuals are related to each other as*

$$\begin{aligned}&\big|T'_\mu[U](x) - U(x)\big| \geq \left|\int_{z\in\mathcal{Z}} dE(z|x)(\mathcal{T}_\pi[V](z) - V(z))\right| \qquad (11)\\ &- \left|\int_z dE(z|x)\int_a d\pi(a|z)(r(x,a) - \bar{r}(z,a))\right| - \frac{R_{\max}}{1-\gamma}\sqrt{\frac{-1}{2}\int_z dE(z|x)\log D(x|z)},\ \ \forall x \in \mathcal{X}.\end{aligned}$$

where $\mu = \pi \circ E$ and $V = U \circ D$.

Proof. Using the definitions of Bellman operators, we may write the following chain of inequalities:

$$\begin{aligned}&\left|T'_\mu[U](x) - \int_{z\in\mathcal{Z}} dE(z|x)\mathcal{T}_\pi[V](z)\right|\\ &\overset{(a)}{\leq} \left|\int_z dE(z|x)\int_a d\pi(a|z)\cdot\left(\int_{x'}\int_{z'} dD(x'|z')\cdot dF(z'|z,a)\cdot\gamma\cdot U(x') - \int_{z'} dF(z'|z,a)\cdot\gamma\cdot V(z')\right)\right|\\ &+ \left|\int_z dE(z|x)\int_a d\pi(a|z)\int_{z'} dF(z'|z,a)\cdot\left(\int_{x'} dD(x'|z')r(x,a) - \bar{r}(z,a)\right)\right|\\ &\overset{(b)}{\leq} \left|\int_z dE(z|x)\int_a d\pi(a|z)\big(r(x,a) - \bar{r}(z,a)\big)\right|, \qquad (12)\end{aligned}$$

where **(a)** is from the definitions of T'_μ and $\mathcal{T}_\pi$, and the triangular inequality, and **(b)** comes from the fact that the first term in the first inequality is zero because $V(z) = \int_{\widetilde{x}} dD(\widetilde{x}|z)U(\widetilde{x})$.

Now we bound the observation approximate Bellman residual at any observation $x \in \mathcal{X}$ as

$$\begin{aligned}&\left|T'_\mu[U](x) - U(x)\right|\\ &= \left|T'_\mu[U](x) - \int_z dE(z|x)\big(\mathcal{T}_\pi[V](z) - V(z)\big) + \int_z dE(z|x)\big(\mathcal{T}_\pi[V](z) - V(z)\big) - U(x)\right|\\ &\geq \left|\int_z dE(z|x)\big(\mathcal{T}_\pi[V](z) - V(z)\big)\right| - \left|T'_\mu[U](x) - \int_z dE(z|x)\mathcal{T}_\pi[V](z)\right| - \left|U(x) - \int_z dE(z|x)V(z)\right|\\ &\overset{(a)}{\geq} \left|\int_z dE(z|x)\big(\mathcal{T}_\pi[V](z) - V(z)\big)\right| - \left|\int_z dE(z|x)\int_a d\pi(a|z)\big(r(x,a) - \bar{r}(z,a)\big)\right|\\ &- \left|\int_{\widetilde{x}}\left(\int_z dE(z|x)dD(\widetilde{x}|z) - d\mathbf{1}\{x = \widetilde{x}\}\right)U(\widetilde{x})\right|, \qquad (13)\end{aligned}$$

where **(a)** is from (12) and the definition of $V = U \circ D$. The last term in (13) can be further upper-bounded as

$$\begin{aligned}\left|\int_{\widetilde{x}}\left(\int_z dE(z|x)dD(\widetilde{x}|z) - d\mathbf{1}\{x = \widetilde{x}\}\right)U(\widetilde{x})\right| &\leq \int_{\widetilde{x}}\left|\int_z dE(z|x)D(\widetilde{x}|z) - \mathbf{1}\{x = \widetilde{x}\}\right|\cdot\|U\|_\infty\\ &\overset{(a)}{\leq} \int_z dE(z|x)\sqrt{\frac{1}{2}D_{\text{KL}}(\mathbf{1}\{\cdot = x\}||D(\cdot|z))}\cdot\|U\|_\infty\\ &= \int_z dE(z|x)\sqrt{\frac{-1}{2}\log D(x|z)}\cdot\|U\|_\infty\\ &\overset{(b)}{\leq} \sqrt{\frac{-1}{2}\int_z dE(z|x)\log D(x|z)}\cdot\frac{R_{\max}}{1-\gamma}. \qquad (14)\end{aligned}$$

where **(a)** follows from the Pinsker's inequality and **(b)** follows from the concavity of the $\sqrt{(\cdot)}$ function. The proof is concluded by combining (13) and (14). □

The right-hand-side (RHS) of (11) contains several terms. The first corresponds to the latent Bellman residual error, the second one corresponds to the latent reward estimation error w.r.t. policy π, and the third one is a reconstruction loss in the encoder-decoder path, which is commonly found in training auto-encoders (and is also a regularization term in PCC). Using (9) and Lemma 3, we can further show that for any $U : \mathcal{X} \to \mathbb{R}$ and at any $x \in \mathcal{X}$, with $V = U \circ D$ we may write

$$\begin{aligned}|T_\mu[U](x) - U(x)| &= \left|T_\mu[U](x) - T'_\mu[U](x) + T'_\mu[U](x) - U(x)\right| \\ &\geq \left|T'_\mu[U](x) - U(x)\right| - \left|T_\mu[U](x) - T'_\mu[U](x)\right| \\ &\geq \left|\int_z dE(z|x)(\mathcal{T}_\pi[V](z) - V(z))\right| \\ &\quad - \frac{\gamma R_{\max}}{\sqrt{2}(1-\gamma)} \cdot \sqrt{D_{\mathrm{KL}}\Big(P_{\pi\circ E}(\cdot|x)||(D \circ F \circ \pi \circ E)(\cdot|x)\Big)} \\ &\quad - \left|\int_z dE(z|x) \int_a d\pi(a|z)(r(x,a) - \bar{r}(z,a))\right| \\ &\quad - \frac{R_{\max}}{1-\gamma}\sqrt{\frac{-1}{2}\int_z dE(z|x)\log D(x|z)}. \end{aligned} \tag{15}$$

Conversely we now prove a lemma that relates the observation Bellman residual $|T_\mu[U](x) - U(x)|$ with the latent Bellman residual $\int_z dE(z|x)\,|\mathcal{T}_\pi[V](z) - V(z)|$ at an arbitrary observation $x \in \mathcal{X}$, where the latent policy π and value function V are related to their observation counterparts μ and U as $\mu = \pi \circ E$, i.e., $\mu(a|x) = \int_z dE(z|x)\pi(a|z)$, and $U = V \circ E$, i.e., $U(x) = \int_{\widetilde{z}} dE(\widetilde{z}|x)V(\widetilde{z})$.

Lemma 4. *For any observation $x \in \mathcal{X}$, encoder-transition-decoder tuple (E, F, D), latent policy π and value function V, the observation and latent Bellman residuals are related to each other as*

$$\begin{aligned}\left|\int_z dE(z|x)\cdot\big(\mathcal{T}_\pi[V](z) - V(z)\big)\right| &\geq \left|T_\mu[U](x) - U(x)\right| \\ &\quad - \left|\int_z dE(z|x)\int_a d\pi(a|z)\big(r(x,a) - \bar{r}(z,a)\big)\right| \\ &\quad - \frac{\gamma R_{\max}}{\sqrt{2}(1-\gamma)}\sqrt{\int_{x'} dP_{\pi\circ E}(x'|x)\cdot D_{KL}\big(E(\cdot|x')||(F_\pi \circ E)(\cdot|x)\big)}, \end{aligned} \tag{16}$$

where $\mu = \pi \circ E$ and $U = V \circ E$.

Proof. Using the definition $U(x) = \int_{\widetilde{z}} dE(\widetilde{z}|x)V(\widetilde{z})$, by adding and subtracting $T_\mu[U](x)$, and the triangular inequality, we may write

$$\left|\int_z dE(z|x)\big(\mathcal{T}_\pi[V](z) - V(z)\big)\right| \geq \left|T_\mu[U](x) - U(x)\right| - \left|\int_z dE(z|x)\mathcal{T}_\pi[V](z) - T_\mu[U](x)\right|.$$

To complete the proof, we now bound the second term on the RHS of the above inequality as

$$\begin{aligned}&\left|\int_z dE(z|x)\mathcal{T}_\pi[V](z) - T_\mu[U](x)\right| \leq \left|\int_z dE(z|x)\int_a d\pi(a|z)\big(r(x,a) - \bar{r}(z,a)\big)\right| + \\ &\gamma\left|\int_a d\mu(a|x)\int_{x'} dP(x'|x,a)\int_{z'} dE(z'|x')V(z') - \int_z dE(z|x)\int_a d\pi(a|z)\int_{z'} dF(z'|z,a)V(z')\right|,\end{aligned}$$

in which we used the definitions of $T_\mu[U](x)$ and $\mathcal{T}_\pi[V](z)$, and the assumption that in this lemma $\mu = \pi \circ E$ and $U = V \circ E$. We now further upper-bound the second term on the RHS of the above inequality as

$$\begin{aligned}&\left|\int_a d\mu(a|x)\int_{x'} dP(x'|x,a)\cdot\int_{z'} dE(z'|x')V(z') - \int_z dE(z|x)\cdot\int_a d\pi(a|z)\cdot\int_{z'} dF(z'|z,a)\cdot V(z')\right| \\ &\leq \int_{z'}\left|\int_a d\mu(a|x)\int_{x'} dP(x'|x,a)\cdot E(z'|x') - \int_z dE(z|x)\cdot\int_a d\pi(a|z)\cdot F(z'|z,a)\right|\cdot\|V\|_\infty \\ &\overset{\text{(a)}}{=} D_{\mathrm{TV}}\left(\int_a d\mu(a|x)\int_{x'} dP(x'|x,a)\cdot E(\cdot|x')\;||\;\int_z dE(z|x)\cdot\int_a d\pi(a|z)\cdot F(\cdot|z,a)\right)\cdot\|V\|_\infty \\ &\overset{\text{(b)}}{\leq} \frac{\|V\|_\infty}{\sqrt{2}}\sqrt{\int_{x'} dP_{\pi\circ E}(x'|x)\cdot D_{\mathrm{KL}}\big(E(\cdot|x')\;||\;(F_\pi \circ E)(\cdot|x)\big)},\end{aligned}$$

where **(a)** follows from the definition of total variation (TV) and **(b)** is the result of the Pinsker inequality and joint convexity of $D_{\text{KL}}(x||y)$. The proof can be completed by combining the above results and using the fact that $\|V\|_\infty \le R_{\max}/(1-\gamma)$. □

Note that since both the observation and latent Bellman operators, T_μ and $\mathcal{T}_\pi$, are contraction mappings, there exists a unique solution $U_\mu : \mathcal{X} \to \mathbb{R}$ to the observation fixed-point equation $T_\mu[U](x) = U(x),\ \forall x \in \mathcal{X}$, and a unique solution $V_\pi : \mathcal{Z} \to \mathbb{R}$ to the latent fixed-point equation $\mathcal{T}_\pi[V](z) = V(z),\ \forall z \in \mathcal{Z}$. Together with the result in (15), we can prove the following theorem.

Lemma 5. *Assume that the latent Bellman operator $\mathcal{T}_\pi$ of policy π is defined according to the encoder-transition-decoder tuple (E, F, D). Let $(V_\pi \circ E)(x) = \int_{\widetilde{z}\in\mathcal{Z}} dE(\widetilde{z}|x)V_\pi(\widetilde{z})$ and $\mu = \pi \circ E$ be the images of the fixed-point of $\mathcal{T}_\pi$ and the latent policy π back in the observation space using the encoder E. Then, for any observation $x \in \mathcal{X}$, we have*

$$\begin{aligned}&\big|T_\mu[V_\pi \circ E](x) - V_\pi \circ E(x)\big| \le \frac{R_{\max}}{1-\gamma}\sqrt{\frac{-1}{2}\int_z dE(z|x)\log D(x|z)}\\&\quad + 2\left|\int_z dE(z|x)\int_{a\in\mathcal{A}} d\pi(a|z)\big(r(x,a) - \bar{r}(z,a)\big)\right|\\&\quad + \frac{\gamma R_{\max}}{\sqrt{2}(1-\gamma)}\left\{\sqrt{D_{KL}\big(P_{\pi\circ E}(\cdot|x)||(D\circ F\circ\pi\circ E)(\cdot|x)\big)} + \sqrt{\int_{x'} dP_{\pi\circ E}(x'|x)\cdot D_{KL}\big(E(\cdot|x')||(F_\pi\circ E)(\cdot|x)\big)}\right\}.\end{aligned}$$

Proof. Using Eq. 15 with $U = U_\mu$ and $V_\mu = U_\mu \circ D$, for any $x \in \mathcal{X}$ we have

$$\begin{aligned}&|T_\mu[U_\mu](x) - U_\mu(x)|\\&\ge \left|\int_{z\in\mathcal{Z}} dE(z|x)(\mathcal{T}_\pi[V_\mu](z) - V_\mu(z))\right| - \frac{\gamma R_{\max}}{\sqrt{2}(1-\gamma)}\cdot\sqrt{D_{\text{KL}}\Big(P_{\pi\circ E}(\cdot|x)||(D\circ F\circ\pi\circ E)(\cdot|x)\Big)}\\&-\left|\int_{z\in\mathcal{Z}} dE(z|x)\int_{a\in\mathcal{A}} d\pi(a|z)(r(x,a)-\bar{r}(z,a))\right| - \frac{R_{\max}}{1-\gamma}\sqrt{\frac{-1}{2}\int_{z\in\mathcal{Z}} dE(z|x)\log D(x|z)}.\end{aligned}$$

On the other hand, from the fact that V_π is the fixed-point solution of $\mathcal{T}_\pi$, i.e., $\mathcal{T}_\pi[V_\pi](z) = V_\pi(z)$, we have

$$\left|\int_z dE(z|x)\big(\mathcal{T}_\pi[V_\mu](z) - V_\mu(z)\big)\right| \ge \left|\int_z dE(z|x)\big(\mathcal{T}_\pi[V_\pi](z) - V_\pi(z)\big)\right| = 0.$$

Using Lemma 4 with $V = V_\pi$ and $U = V_\pi \circ E$ and the above inequality, we can further show that

$$\begin{aligned}&\left|\int_z dE(z|x)\big(\mathcal{T}_\pi[V_\mu](z) - V_\mu(z)\big)\right| \ge \big|T_\mu[V_\pi\circ E](x) - (V_\pi\circ E)(x)\big| - \left|\int_z dE(z|x)\int_a d\pi(a|z)\big(r(x,a) - \bar{r}(z,a)\big)\right|\\&- \frac{\gamma R_{\max}}{\sqrt{2}(1-\gamma)}\sqrt{\int_{x'} dP_{\pi\circ E}(x'|x)\cdot D_{\text{KL}}\big(E(\cdot|x')||(F_\pi\circ E)(\cdot|x)\big)}.\end{aligned}$$

Together these inequalities imply that

$$\begin{aligned}&|T_\mu[U_\mu](x) - U_\mu(x)| \ge \big|T_\mu[V_\pi\circ E](x) - (V_\pi\circ E)(x)\big| - 2\left|\int_z dE(z|x)\int_a d\pi(a|z)(r(x,a) - \bar{r}(z,a))\right|\\&- \frac{\gamma R_{\max}}{\sqrt{2}(1-\gamma)}\sqrt{\int_{x'} dP_{\pi\circ E}(x'|x)\cdot D_{\text{KL}}\big(E(\cdot|x')||(F_\pi\circ E)(\cdot|x)\big)}\\&- \frac{\gamma R_{\max}}{\sqrt{2}(1-\gamma)}\sqrt{D_{\text{KL}}\big(P_{\pi\circ E}(\cdot|x)||(D\circ F\circ\pi\circ E)(\cdot|x)\big)} - \frac{R_{\max}}{1-\gamma}\sqrt{\frac{-1}{2}\int_z dE(z|x)\log D(x|z)}.\end{aligned} \tag{17}$$

Recall that U_μ is a fixed-point of T_μ i.e., $T_\mu[U_\mu](x) = U_\mu(x), \forall x \in \mathcal{X}$, we can then complete the proof by setting the LHS of (17) to zero. □

This theory shows that the observation Bellmen residual error w.r.t. value function $V_\pi \circ E$, where V_π is the optimal latent value function (w.r.t. soft Bellman fixed-point equation), depends on (i) the prediction error, (ii) the consistency term, (iii) latent reward estimation error, and (iv) the encoder-decoder reconstruction error. Using analogous derivations as in (10) for the prediction term. Alternatively, we can further derive the following observation Bellman residual error upper bound w.r.t. value function $V_\pi \circ E$ that does not depend on the policy.

Corollary 6. *Let $V_\pi \circ E(x) = \int_{\widetilde{z}\in\mathcal{Z}} dE(\widetilde{z}|x)V_\pi(\widetilde{z})$, be the observation value function in which V_π is the solution of the latent fixed-point equation $\mathcal{T}_\pi[V](z) = V(z)$ w.r.t. latent policy π, encoder-transition-decoder tuple (E, F, D), and parameterized observation-based policy $\mu = \pi \circ E$. Then the following statement holds at any $x \in \mathcal{X}$:*

$$\begin{aligned}\big|T_\mu[V_\pi \circ E](x) - V_\pi \circ E(x)\big| \leq{}& \frac{R_{\max}}{1-\gamma}\sqrt{\frac{-1}{2}\int_{z\in\mathcal{Z}} dE(z|x)\log D(x|z)} \\ &+ 2\cdot\max_{a\in\mathcal{A}}\left\{\left|r(x,a) - \int_z dE(z|x)\bar{r}(z,a)\right| + \frac{\gamma R_{\max}}{\sqrt{2}(1-\gamma)}\left\{\sqrt{D_{KL}\Big(P(\cdot|x,a)||(D\circ F\circ E)(\cdot|x,a)\Big)}\right.\right. \\ &\left.\left.+\sqrt{\int_{x'\in\mathcal{X}} dP(x'|x,a)\cdot D_{KL}\left(E(\cdot|x')||F\circ E(\cdot|x,a)\right)}\right\}\right\}.\end{aligned}$$

Combining the above results, we have the following main theorem on approximate policy evaluation that connects the observation value w.r.t. policy $\pi \circ E$ and its latent value counterpart.

Theorem 7. *The observation value function $U_{\pi\circ E}$ w.r.t. policy $\pi \circ E$ satisfies the following bound*

$$|V_\pi \circ E(x) - U_{\pi\circ E}(x)| \leq \frac{1}{1-\gamma}E_{d^\gamma_{\pi\circ E}}\left[\Delta(E,F,D,\bar{r},\pi,\cdot)|x_0 = x\right],\ \forall x \in \mathcal{X}.$$

where the error term is given by

$$\begin{aligned}\Delta(E,F,D,\bar{r},\pi,x) ={}& \frac{R_{\max}}{1-\gamma}\sqrt{\frac{-1}{2}\int_z dE(z|x)\log D(x|z)} \\ &+2\Big|\int_z dE(z|x)\int_a d\pi(a|z)(r(x,a)-\bar{r}(z,a))\Big| + \frac{\gamma R_{\max}}{\sqrt{2}(1-\gamma)}\sqrt{D_{KL}\big(P_{\pi\circ E}(\cdot|x)\ ||\ (D\circ F_\pi\circ E)(\cdot|x)\big)} \\ &+\frac{\gamma R_{\max}}{\sqrt{2}(1-\gamma)}\sqrt{\int_{x'\in\mathcal{X}} dP_{\pi\circ E}(x'|x)\cdot D_{KL}\left(E(\cdot|x')||(F_\pi\circ E)(\cdot|x)\right)}.\end{aligned}$$

Proof. To prove the right side of the approximate policy evaluation inequality, recall from Lemma 5 with $\mu = \pi \circ E$ and LCE-reward models $(E, F, D, \bar{r})$ that

$$T_{\pi\circ E}[V_\pi \circ E](x) \leq V_\pi \circ E(x) + \Delta(E,F,D,\bar{r},\pi,x),\ \forall x \in \mathcal{X}.$$

Applying the Bellman operator $T_{\pi\circ E}$ on both sides, we get

$$\begin{aligned}T^2_{\pi\circ E}[V_\pi \circ E](x) &\leq T_{\pi\circ E}[V_\pi\circ E](x) + \gamma\int_a(\pi\circ E)(a|x)\int_{x'} dP(x'|x,a)\Delta(E,F,D,\bar{r},\pi,x'), \\ &= T_{\pi\circ E}[V_\pi\circ E](x) + \gamma\,\mathbb{E}_{P_{\pi\circ E}}[\Delta(E,F,D,\bar{r},\pi,\cdot)|x],\ \forall x\in\mathcal{X}.\end{aligned}$$

Proceeding similarly, with $P^{\ell-1}_{\pi\circ E}(x'|x) = \mathbb{P}(x_{\ell-1} = x'|x_0 = x; \pi\circ E)$ it follows that

$$T^\ell_{\pi\circ E}[V_\pi\circ E](x) \leq T^{\ell-1}_{\pi\circ E}[V_\pi\circ E](x) + \gamma^{\ell-1}\mathbb{E}_{P^{\ell-1}_{\pi\circ E}}[\Delta(E,F,D,\bar{r},\pi,\cdot)|x],\ \forall x\in\mathcal{X}.$$

Therefore, for every $k > 0$

$$\begin{aligned}T^k_{\pi\circ E}[V_\pi\circ E](x) - V_\pi\circ E(x) &= \sum_{\ell=1}^k\big(T^\ell_{\pi\circ E}[V_\pi\circ E](x) - T^{\ell-1}_{\pi\circ E}[V_\pi\circ E](x)\big) \\ &\leq \sum_{\ell=1}^k\gamma^{\ell-1}\mathbb{E}_{P^{\ell-1}_{\pi\circ E}}[\Delta(E,F,D,\bar{r},\pi,\cdot)|x].\end{aligned}$$

Taking $k \to \infty$, we then obtain

$$U_{\pi\circ E}(x) = T^{\infty}_{\pi\circ E}[V_\pi \circ E](x) \le V_\pi \circ E(x) + \frac{1}{1-\gamma}\mathbb{E}_{d^\gamma_{\pi\circ E}}\left[\Delta(E,F,D,\bar{r},\pi,\cdot)|x_0 = x\right],$$

where $d^\gamma_{\pi\circ E}(x'|x_0) = (1-\gamma)\cdot\sum_{\ell=0}^\infty \gamma^\ell \mathbb{P}(x_\ell = x'|x_0,\, \pi\circ E)$ is the γ-stationary distribution of policy $\pi\circ E$ starting at state x_0. On the other hand, the left-hand-side of the policy evaluation inequality follows analogous arguments. This completes the proof of the approximate policy evaluation. □

A.2 Approximate Policy Iteration with CARL

Given the LCE model (E, F, D) and policy μ, recall from Algorithm 1 the policy iteration procedure:

1. Compute the distilled latent policy by $\pi \leftarrow \arg\min_\mu D_{\text{KL}}(\pi\circ E(\cdot|x)||\mu(\cdot|x))$
2. Compute the π-induced latent value function $V_\pi(z) = \lim_{n\to\infty} T^n_\pi[V](z),\ \forall z$ w.r.t. models $(F,\bar{r})$ and the state-action latent value function $Q_\pi(z,a) := \bar{r}(z,a) + \gamma\int_{z'\in\mathcal{X}} dF(z'|z,a)V_\pi(z')$
3. Compute the updated latent policy $\pi_+(\cdot|z) \in \arg\max_{p\in\Delta}\int_{a\in\mathcal{A}} dp(a)\cdot Q_\pi(z,a)$, and the updated observation policy $\mu_+(\cdot|x) = \pi_+\circ E(\cdot|x)$
4. Update the LCE model (E, F, D) and the latent reward model $\bar{r}$ by minimizing the loss $\Delta(E,F,D,\bar{r},\pi_+)$
5. Repeat step 1 to step 4

Using the intermediate technical results in policy evaluation (Theorem 7), we can now prove the approximate policy improvement result in Theorem 1 for this procedure, which shows the relationship between the value functions of two consecutive polices generated by LSLPI in $\mathcal{X}$.

Proof of Theorem 1. To start with, using the policy evaluation property from Thoerem 7, we have

$$U_{\pi\circ E}(x) \le V_\pi\circ E(x) + \frac{1}{1-\gamma}E_{d^\gamma_{\pi\circ E}}\left[\Delta(E,F,D,\bar{r},\pi,\cdot)|x_0 = x\right],$$

for all $x\in\mathcal{X}$, where the error term Δ for a policy π is given by

$$\Delta(E,F,D,\bar{r},\pi,x) = \frac{R_{\max}}{1-\gamma}\overbrace{\sqrt{\frac{-1}{2}\int_z dE(z|x)\log D(x|z)}}^{\text{(I)=L}_{\text{ed}}\text{(E,D,x)}} + 2\overbrace{\left|r_{\pi\circ E}(x) - \int_z dE(z|x)\bar{r}_\pi(z)\right|}^{\text{(II)=L}_{\text{r}}(\text{E},\bar{\text{r}},\pi,\text{x})} \tag{18}$$
$$+\frac{\gamma R_{\max}}{\sqrt{2}(1-\gamma)}\Big(\underbrace{\sqrt{D_{\text{KL}}\big(P_{\pi\circ E}(\cdot|x)\ ||\ (D\circ F_\pi\circ E)(\cdot|x)\big)}}_{\text{(III)=L}_{\text{p}}(\text{E,F,D},\pi,\text{x})} + \underbrace{\sqrt{D_{\text{KL}}\big((E\circ P_{\pi\circ E})(\cdot|x)\ ||\ (F_\pi\circ E)(\cdot|x)\big)}}_{\text{(IV)}}\Big).$$

Applying Bellman operator $T_{\pi\circ E}$ on both sides and noticing that $T_{\pi\circ E}[U](x) \le T[U](x)$ uniformly for all $x\in\mathcal{X}$ and for any observation value function U, we can then show that for any $x\in\mathcal{X}$,

$$\begin{aligned}
&U_{\pi\circ E}(x) \le V_\pi\circ E(x) + \frac{1}{1-\gamma}\cdot\mathbb{E}_{d^\gamma_{\pi\circ E}}\left[\Delta(E,F,D,\bar{r},\pi,\cdot)|x_0=x\right]\\
&=\int_{z\in\mathcal{Z}} dE(z|x)\mathcal{T}_\pi[V_\pi](z) + \frac{1}{1-\gamma}\cdot\mathbb{E}_{d^\gamma_{\pi\circ E}}\left[\Delta(E,F,D,\bar{r},\pi,\cdot)|x_0=x\right]\\
&\le\int_{z\in\mathcal{Z}} dE(z|x)\mathcal{T}[V_\pi](z) + \frac{1}{1-\gamma}\cdot\mathbb{E}_{d^\gamma_{\pi\circ E}}\left[\Delta(E,F,D,\bar{r},\pi,\cdot)|x_0=x\right]\\
&=\int_{z\in\mathcal{Z}} dE(z|x)\mathcal{T}_{\pi_+}[V_\pi](z) + \frac{1}{1-\gamma}\cdot\mathbb{E}_{d^\gamma_{\pi\circ E}}\left[\Delta(E,F,D,\bar{r},\pi,\cdot)|x_0=x\right]\\
&\le\int_{z\in\mathcal{Z}} dE(z|x)\mathcal{T}_{\pi_+}[V_{\pi_+}](z) + \frac{1}{1-\gamma}\cdot\mathbb{E}_{d^\gamma_{\pi\circ E}}\left[\Delta(E,F,D,\bar{r},\pi,\cdot)|x_0=x\right]\\
&=\int_{z\in\mathcal{Z}} dE(z|x)V_{\pi_+}(z) + \frac{1}{1-\gamma}\cdot\mathbb{E}_{d^\gamma_{\pi\circ E}}\left[\Delta(E,F,D,\bar{r},\pi,\cdot)|x_0=x\right]\\
&\le\underbrace{U_{\pi_+\circ E}(x)}_{U_{\mu_+}(x)} + \frac{\mathbb{E}_{d^\gamma_{\pi\circ E}}\left[\Delta(E,F,D,\bar{r},\pi,\cdot)|x_0=x\right] + \mathbb{E}_{d^\gamma_{\pi_+\circ E}}\left[\Delta(E,F,D,\bar{r},\pi_+,\cdot)|x_0=x\right]}{1-\gamma}.
\end{aligned}\tag{19}$$

The first inequality is based on Theorem 7. The first equality is based on the property that V_π is a unique solution to fixed-point equation $\mathcal{T}_\pi[V](z) = V(z)$. The second inequality is based on the definition of

$$\begin{aligned}\mathcal{T}[V](z) &= \max_{p\in\Delta}\int_a dp(a)\left\{\bar{r}(z,a) + \gamma\int_{z'\in\mathcal{X}} dF(z'|z,a)V_\pi(z')\right\}\\ &\geq \int_a d\pi(a|z)\left\{\bar{r}(z,a) + \gamma\int_{z'\in\mathcal{X}} dF(z'|z,a)V_\pi(z')\right\}.\end{aligned}$$

The second equality is based on the definition of π_+. The third inequality is based on the policy improvement property in latent policy iteration, i.e.,

$$\mathcal{T}[V_\pi] \geq \mathcal{T}_\pi[V_\pi] = V_\pi \implies V_{\pi_+} = \lim_{k\to\infty}\mathcal{T}^k_{\pi_+}[V_\pi] = \lim_{k\to\infty}\mathcal{T}^k[V_\pi] \geq V_\pi,$$

and the monotonicity property of latent Bellman operator. The third equality is based on the fact that V_{π_+} is a unique solution to fixed-point equation $\mathcal{T}_{\pi_+}[V](z) = V(z)$. The fourth inequality is again based on Lemma 5 (when $\pi = \pi_+$).

Furthermore, considering the error from the distillation step, using Lemma 3 of Achiam et al. (2017) and Pinsker's inequality, one can show that

$$U_\mu(x) \leq U_{\pi\circ E}(x) + \frac{\sqrt{2}\gamma R_{\max}}{(1-\gamma)}\cdot\mathbb{E}_{d^\gamma_{\pi\circ E}}\left[\sqrt{D_{\text{KL}}(\pi\circ E(\cdot'|\cdot)||\mu(\cdot'|\cdot))}|x_0 = x\right]. \tag{20}$$

Together this completes the proof of the approximate policy improvement result in Theorem 1. $\square$

B Offline CARL

Given the LCE model (E, F, D) and policy μ, consider the following offline latent policy iteration procedure that optimizes the policy in form of $\mu\circ E$:

1. Compute the distilled latent policy by $\pi \leftarrow \arg\min_\mu D_{\text{KL}}(\pi\circ E(\cdot|x)||\mu(\cdot|x))$
2. Compute the updated latent policy $\pi_+(\cdot|z) \in \arg\max_{p\in\Delta}\int_{a\in\mathcal{A}} dp(a)\cdot Q_\pi(z,a)$, and the updated observation policy $\mu_+(\cdot|x) = \pi_+\circ E(\cdot|x)$
3. Repeat step 1 to step 2

We are now in position to prove the approximate policy improvement result for offline CARL in Corollary 2, which marginalize out the policy from the (online) CARL's loss function.

Proof of Corollary 2. Compared with the approximate policy improvement result of online CARL in Theorem 1, in offline CARL when the LCE model (E, F, D) and the latent reward model $\bar{r}$ do not change online, there is no need for the distillation step. Following analogous arguments as in Corollary 6, to eliminate the dependencies on policies below we replace the PI bound $\Delta(E, F, D, \bar{r}, \pi, x)$ in Theorem 1 with the more conservative bound, which considers the worst-case error term over actions, i.e., $\max_{a\in\mathcal{A}}\Delta(E, F, D, \bar{r}, a, x)$.

Denote by $T_{\mu^*}[U](x)$ the observation Bellman operator w.r.t. optimal latent policy π^*. Recall that $\int_{z\in\mathcal{Z}} dE(z|x)\mathcal{T}_{\pi_+}[V_{\pi_+}](z) = V_{\pi_+}\circ E(x)$. Using the results in Theorem 7, we have the following chain on inequalities

$$\begin{aligned}U_{\mu_+}(x) \geq& \int_{z\in\mathcal{Z}} dE(z|x)\mathcal{T}_{\pi_+}[V_{\pi_+}](z) - \frac{1}{1-\gamma}\cdot\mathbb{E}_{d^\gamma_{\pi_+\circ E}}\left[\max_{a\in\mathcal{A}}\Delta(E,F,D,\bar{r},a,\cdot)|x_0 = x\right]\\ \geq& \int_{z\in\mathcal{Z}} dE(z|x)\mathcal{T}_{\pi_+}[V_\pi](z) - \frac{1}{1-\gamma}\cdot\mathbb{E}_{d^\gamma_{\pi_+\circ E}}\left[\max_{a\in\mathcal{A}}\Delta(E,F,D,\bar{r},a,\cdot)|x_0 = x\right]\\ =& \int_{z\in\mathcal{Z}} dE(z|x)\mathcal{T}[V_\pi](z) - \frac{1}{1-\gamma}\cdot\mathbb{E}_{d^\gamma_{\pi_+\circ E}}\left[\max_{a\in\mathcal{A}}\Delta(E,F,D,\bar{r},a,\cdot)|x_0 = x\right]\\ \geq& \int_{z\in\mathcal{Z}} dE(z|x)\mathcal{T}_{\pi^*}[V_\pi](x) - \frac{1}{1-\gamma}\cdot\mathbb{E}_{d^\gamma_{\pi_+\circ E}}\left[\max_{a\in\mathcal{A}}\Delta(E,F,D,\bar{r},a,\cdot)|x_0 = x\right]\\ \geq& T_{\mu^*}[V_\mu\circ E](x) - \frac{1+(1-\gamma)}{1-\gamma}\cdot\max_{x\in\mathcal{X},a\in\mathcal{A}}\Delta(E,F,D,\bar{r},a,x),\\ \geq& T_{\mu^*}[U_\mu](z) - \frac{1+(1-\gamma)+\gamma}{1-\gamma}\cdot\max_{x\in\mathcal{X},a\in\mathcal{A}}\Delta(E,F,D,\bar{r},a,x),\end{aligned}$$

where the first three inequality and equality follows similar argument as in the proof of Theorem 1 (but instead of Lemma 5 we use the results from Corollary 6), the fourth inequality follows from direct algebraic manipulations, and the last inequality follows from Theorem 7 when applied to U_μ, i.e.,

$$\begin{aligned} U_\mu(x) \leq & V_\pi \circ E(x) + \frac{1}{1-\gamma} \cdot \mathbb{E}_{d^\gamma_{\pi\circ E}} \left[\max_{a\in\mathcal{A}} \Delta(E, F, D, \bar{r}, a, \cdot) | x_0 = x \right] \\ \leq & V_\pi \circ E(x) + \frac{1}{1-\gamma} \cdot \max_{x\in\mathcal{X}, a\in\mathcal{A}} \Delta(E, F, D, \bar{r}, a, x) \end{aligned}$$

and from the contraction property of T_{μ^*}, i.e.,

$$T_{\mu^*}[U_\mu](x) \leq T_{\mu^*}[V_\pi \circ E](x) + \frac{\gamma}{1-\gamma} \cdot \max_{x\in\mathcal{X}, a\in\mathcal{A}} \Delta(E, F, D, \bar{r}, a, x).$$

To prove the inequality in equation 7, i.e., the final result of Corollary 2, notice that with U^* equal to the fixed-point solution of Bellman operator T_{μ^*}, we have the following property:

$$-\gamma\|U_\mu - U^*\|_\infty \leq T_{\mu^*}[U_\mu] - T_{\mu^*}[U^*] = T_{\pi^*}[U_\mu] - U^* \leq \gamma\|U_\mu - U^*\|_\infty.$$

Notice that by definition $U_{\mu_+}(x) \leq U^*(x)$, we then have the chain of inequalities

$$\begin{aligned} -\|U_{\mu_+} - U^*\|_\infty = U_{\mu_+}(x) - U^*(x) \geq & T_{\mu^*}[U_\mu](x) - U^*(x) - \frac{2}{1-\gamma} \max_{x\in\mathcal{X}, a\in\mathcal{A}} \Delta(E, F, D, \bar{r}, a, x) \\ \geq & -\gamma\|U_\mu - U^*\|_\infty - \frac{2}{1-\gamma} \cdot \max_{x\in\mathcal{X}, a\in\mathcal{A}} \Delta(E, F, D, \bar{r}, a, x). \end{aligned}$$

In other words, we have the following inequality:

$$\|U_{\mu_+}(x) - U^*(x)\|_\infty \leq \gamma\|U_\mu - U^*\|_\infty + \frac{2}{1-\gamma} \max_{x\in\mathcal{X}, a\in\mathcal{A}} \Delta(E, F, D, \bar{r}, a, x).$$

Thus the proof is completed by taking $i \to \infty$ and noticing that

$$\lim_{i\to\infty} \|U_{\mu_+}(x) - U^*(x)\|_\infty = \lim_{i\to\infty} \|U_{\mu_+}(x) - U^*(x)\|_\infty.$$

□

C Value-Guided CARL (V-CARL)

Based on variational model-based policy optimization (Chow et al., 2020), the "optimal" dynamics for model-based RL has the following closed-form solution:

$$P^*(x'|x,a) = \frac{P(x'|x,a) \cdot \exp\left(\tau \cdot \tilde{U}_\mu(x')\right)}{\exp\left(\tau \cdot (\tilde{W}_\mu(x,a) - r(x,a))/\gamma\right)} = P(x'|x,a) \cdot \exp\left(\tau \cdot \frac{r(x,a) + \gamma\tilde{U}_\mu(x') - \tilde{W}_\mu(x,a)}{\gamma}\right), \tag{21}$$

in which $\tilde{U}_\mu(x)$ is the optimistic observation value function at policy μ, i.e.,

$$\tilde{U}_\mu(x) := \frac{1}{\tau} \log \mathbb{E}\left[\exp\left(\tau \cdot \sum_{t=0}^{\infty} \gamma^t r_\mu(x_t)\right) \mid P_\mu, x_0 = x\right],$$

which is also a unique solution that satisfies the fixed-point property:

$$\tilde{U}_\mu(x) = \int_a d\mu(a|x) \left[r(x,a) + \gamma \cdot \frac{1}{\tau} \cdot \log \mathbb{E}_{x'\sim P(\cdot|x,a)}\left[\exp\left(\tau \cdot \tilde{U}(x')\right)\right]\right],$$

and $\tilde{W}_\mu(x)$ is the optimistic observation state-action value function at policy μ, i.e.,

$$\tilde{W}_\mu(x,a) := r(x,a) + \gamma \cdot \frac{1}{\tau} \cdot \log \mathbb{E}_{x'\sim P(\cdot|x,a)}\left[\exp\left(\tau \cdot U_\mu(x')\right)\right].$$

This modified dynamics model P^* is an exponential twisting of the original transition dynamics P with weight

$$w(x,a,x') = \tau \cdot (r(x,a) + \gamma\tilde{U}_\mu(x') - \tilde{W}_\mu(x,a))/\gamma, \tag{22}$$

which corresponds to the standard discounted TD-error of the optimistic value functions.

To incorporate this model-learning paradigm in V-CARL, one replaces the transition model P in the prediction loss with P^*. Assuming that $\mu \approx \pi \circ E$ (which is enforced by the distillation loss term in CARL), the prediction loss can be approximated as

$$L_p(E, F, D, \pi, x) \approx D_{\text{KL}}\big(P_\mu(\cdot|x) \mid\mid (D \circ F_\pi \circ E)(\cdot|x)\big) = \int_{x'} dP_\mu(x'|x) \log \frac{(D \circ F_\pi \circ E)(x'|x)}{P_\mu(x'|x)}.$$

Since the term $\log P_\mu(x'|x)$ is independent to the LCE model, minimizing L_p is equivalently to maximizing the expected log-likelihood:

$$\int_{x'} dP_\mu(x'|x) \cdot \log(D \circ F_\pi \circ E)(x'|x).$$

Replacing the transition dynamics model P with P^*, this objective function can be re-written as

$$\begin{aligned}\int_{x'} dP^*_\mu(x'|x) \cdot \log(D \circ F_\pi \circ E)(x'|x) &= \int_{x'} dP^*_\mu(x'|x) \cdot \log(D \circ F_\pi \circ E)(x'|x) \\ &= \int_a d\mu(a|x) \int_{x'} dP(x'|x, a) \cdot \exp(w(x, a, x')) \cdot \log(D \circ F_\pi \circ E)(x'|x).\end{aligned} \tag{23}$$

Maximizing this function corresponds to a maximum weighted log-likelihood model-learning approach (w.r.t. P) for which the weight is the exponential TD $w(x, a, x')$

Below we propose ways to efficiently compute the exponential TD weight $w(x, a, x') = \tau \cdot (r(x, a) + \gamma \tilde{U}_\mu(x') - \tilde{W}_\mu(x, a))/\gamma$. For simplicity we consider the case when $\tau > 0$ is small, for which Ruszczyński & Shapiro (2006) shows that $\rho_\tau(U(\cdot)|x, a) \approx \mathbb{E}[U|x, a]$. In other cases when this approximation does not hold, one needs to directly learn the optimistic value function $\tilde{U}_\mu(x)$ and state-action value functions $\tilde{W}_\mu(x, a)$, for which the details can be found in Borkar (2002). Under this approximation, we can approximate the optimistic value functions with their standard counterparts, i.e.,

$$\tilde{U}_\mu(x) \approx U_\mu(x), \qquad \tilde{W}_\mu(x, a) \approx W_\mu(x, a) := r(x, a) + \int_{x'} dP(x'|x, a) U_\mu(x'),\ (x, a) \in \mathcal{X} \times \mathcal{A}.$$

Instead of computing the value functions U_μ and W_μ in the observation space, we can approximate them with their low-dimensional latent-space counterparts. In particular, Lemma 5 implies that

$$|V_\pi \circ E(x) - U_\mu(x)| \le \frac{\gamma}{1-\gamma} \Delta(E, F, D, \bar{r}, \pi, x) + \frac{R_{\max}}{1-\gamma} \cdot D_{\text{KL}}(\pi \circ E(\cdot|x) || \mu(\cdot|x)),\ \forall x \in \mathcal{X}.$$

Since we are minimizing the terms on the right side of the bound for the LCE model, assuming these terms are small, we have $U_\mu(x) \approx V_\pi \circ E(x)$. Following analogous derivations we also have the following error bound for the state-action value function:

$$\begin{aligned}&|Q_\pi \circ E(x, a) - W_\mu(x, a)| \\ \le & \frac{\gamma}{1-\gamma} \Delta(E, F, D, \bar{r}, x, a) + \frac{\gamma R_{\max}}{1-\gamma} \cdot D_{\text{KL}}(\pi \circ E(\cdot|x) || \mu(\cdot|x)),\ \forall x \in \mathcal{X},\ a \in \mathcal{A}.\end{aligned}$$

where the state-action error term is given by

$$\begin{aligned}\Delta(E, F, D, \bar{r}, x, a) := & \frac{R_{\max}}{1-\gamma} \sqrt{\frac{-1}{2} \int_z dE(z|x) \log D(x|z)} \\ & + 2\big|r(x, a) - \bar{r}(z, a)\big| + \frac{\gamma R_{\max}}{\sqrt{2}(1-\gamma)} \sqrt{D_{\text{KL}}\big(P(\cdot|x, a) \mid\mid (D \circ F \circ E)(\cdot|x, a)\big)} \\ & + \frac{\gamma R_{\max}}{\sqrt{2}(1-\gamma)} \sqrt{\int_{x' \in \mathcal{X}} dP(x'|x, a) \cdot D_{\text{KL}}\left(E(\cdot|x') || (F \circ E)(\cdot|x, a)\right)}.\end{aligned}$$

Similarly, the state-action value function can be approximated by $W_\mu(x, a) \approx Q_\pi \circ E(x, a)$.

Finally, recall that latent reward function is learned to minimize the following loss: $\big|\int_{z,a} dE(z|x) d\pi(a|z)(r(x, a) - \bar{r}(z, a))\big|$, such that the reward model satisfies $r(x, a) \approx \bar{r} \circ E(x, a)$. Together, the exponential TD weight can be approximated by the latent reward, latent value function, and latent state-action value function as follows:

$$w(x, a, x') \approx \widehat{w}(x, a, x') := \int_{z,z'} dE(z|x) \cdot dE(z'|x') \cdot (\bar{r}(z, a) - Q_\pi(z, a)) + \gamma V_\pi(z').$$

D CARL Algorithms

Below in Algorithm 2 we present the practical implementation of the CARL algorithm with notation for all of its variants (offline CARL, online CARL, Value-Guided CARL).

Algorithm 2 Control Aware Representation Learning (CARL)

```
 1: Inputs: A dataset $\mathcal{D}_{\text{real}}$ tuples $(x, a, x', x_g)$ from the environment, $Env$. A latent controllable embedding (LCE) network $\mathcal{M}$ consisting of an encoder $E : \mathcal{X} \to \mathcal{Z}$, transition dynamics $F : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$, decoder $D : \mathcal{Z} \to \mathcal{X}$, and backwards encoder $B : \mathcal{X} \times \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$; plus networks for control: latent critic networks $V_{\phi_1}, V_{\phi_2} : \mathcal{Z} \to \mathbb{R}$, $Q_{\theta_1}, Q_{\theta_2} : \mathcal{Z} \times \mathcal{A} \to \mathbb{R}$, and latent actor network $\pi_\psi : \mathcal{Z} \to \mathcal{A}$
 2: #For $T = 0$, the algorithm becomes Offline CARL
 3: for $i = 0, \dots, T$ do
 4:     for $j = 1, \dots,$ num_pcc_epochs do
 5:         #Representation learning.
 6:         Train $\mathcal{M}^{(i)}$ using dataset $\mathcal{D}_{\text{real}}$ model
 7:         For value-guided CARL, the prediction and consistency loss functions requires the exponential twisting weight $\exp(\frac{\tau}{\gamma} \cdot \hat{w}(x, a, x'))$, where $\hat{w}(x, a, x') = \int_{z,z'} E^{(i-1)}(z|x) \cdot E^{(i-1)}(z'|x') \cdot (-||z_g - z'||_2 - Q_{\bar{\phi}}(z, a)) + \gamma V_{\bar{\phi}}(z'))$
 8:     end for
 9:     Initialize a soft actor critic (SAC) policy $\pi^{(i)}$
10:     if Do policy distillation and $i \geq 1$ then
11:         #Corresponds to the policy distillation loss $L_p$
12:         for Each policy distillation epoch do
13:             $\pi_\psi^{(i)} \leftarrow \pi_\psi^{(i)} - \nabla_\psi \mathbb{E}_{x\sim D} \left[ D_{KL} \left( \pi_\psi^{(i)} \left( E^{(i)}(\cdot|x) \right) || \pi_\psi^{(i-1)} \left( E^{(i-1)}(\cdot|x) \right) \right) \right]$
14:         end for
15:     end if
16:     Initialize a latent space buffer $\mathcal{B}_{latent}$
17:     #Learning a latent space policy
18:     for Each soft actor critic step do
19:         Sample real dataset $(x, a, x_g) \sim \mathcal{D}_{\text{real}}^{(i)}$
20:         Generate necessary latent space variables:
            $z \sim E(\cdot|x), z' \sim F(\cdot|z, u), z_g \sim E(\cdot|x_g), r = -||z_g - z'||_2$
21:         Add latent batch to latent buffer $\mathcal{B}_{latent} \leftarrow \mathcal{B}_{latent} \cup (z, a, z', r, z_g)$
22:         Sample latent buffer $(z, a, z', r, z_g) \sim \mathcal{B}_{latent}$
23:         #Train the policy $\pi^{(i)}$ with $(z, u, z', r, z_g)$ with the SAC algorithm
24:         $\theta_i \leftarrow \theta_i - \kappa_Q \nabla_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$          #Update the Q-function weights
25:         $\phi_i \leftarrow \phi_i - \kappa_V \nabla_{\phi_i} J_V(\phi_i)$ for $i \in \{1, 2\}$          #Update the V-function weights
26:         $\psi \leftarrow \psi - \kappa_\pi \nabla_\psi J_\psi(\psi)$          #Update the policy weights
27:         $\bar{\theta}_i \leftarrow \nu\theta_i + (1 - \nu)\bar{\theta}_i$ for $i \in \{1, 2\}$          #Update the Q-target critic networks weights
28:         $\bar{\phi}_i \leftarrow \nu\phi_i + (1 - \nu)\bar{\phi}_i$ for $i \in \{1, 2\}$          #Update the V-target critic networks weights
29:     end for
30:     #Sample the environment for new real data
31:     for Each Interaction with Environment do
32:         Sample Actions $a \sim \pi^{(i)}(E(\cdot|x))$
33:         Interact with the environment $x' \leftarrow Env(x, a)$
34:         Update real data dataset $\mathcal{D}_{\text{real}} \leftarrow \mathcal{D}_{\text{real}} \cup (x, a, x', x_g)$
35:         Update current state $x \leftarrow x'$
36:     end for
37: end for
```

Soft Actor Critic (SAC) Updates The policy parameters ψ are optimized to update the latent space policy towards the exponential of the soft Q-function,

$$J_\pi(\psi) = \underset{z_t\sim\mathcal{D}}{\mathbb{E}} \left[\underset{a_t\sim\pi_\psi}{\mathbb{E}} \left[\alpha log(\pi_\psi(a_t|z_t) - Q_\theta(z_t, a_t) \right] \right] \tag{24}$$

Our updates to the Q network minimize the following loss function:

$$J_Q(\theta) = \mathbb{E}_{(z_t,a_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(z_t,a_t) - \hat{Q}(z_t,a_t)\right)^2\right] \tag{25}$$

where:

$$\hat{Q}_\theta(z_t,a_t) = r(z_t,a_t) + \sum_{i=0}^{k-1}\gamma^{i+1} r(z_{t+i+1},a_{t+i+1})|a_t \sim \pi_\psi(\cdot|z_t), z_{t+1} \sim F(\cdot|z_t,a_t) \tag{26}$$

here, F is the learned latent space transition model and $r(z_t,a_t) = ||z_{goal} - z_{t+1}||_2^2$ where, $z_{t+1} \sim F(\cdot|z_t,a_t)$ and $z_{goal} \sim E(\cdot|x_{goal})$ and x_{goal} is the observation of the environment; additionally, k is a tunable hyperparameter of the number of rollouts in the latent space should we rollout our model to sufficiently approximate the Q-value. As in Haarnoja et al. (2018) we utilize two Q-functions and take the minimum of the Q-functions to generate the value in the actor loss function. We note that we don't have value network updates as we tried adding value networks but were unable to get good results.

E Experimental Setup

In this section, we provide a description of the domains and implementation details used in our experiments. For all the experiments, when using iLQR we define the cost function as $c(z,a) = (z - z_g)^\top Q(z - z_g) + a^\top Ra$, where z and z_g are latent states of the current and goal observation, and $Q = \kappa \cdot I_{n_z}$ with $\kappa = 50$ and $R = I_{n_a}$ are the penalty weights of observation and action. This reward configuration follows exactly from Levine et al. (2020).

E.1 Domains Description

Planar System In this task the main goal is to navigate an agent in a surrounded area on a 2D plane (Breivik & Fossen, 2005), whose goal is to navigate from a corner to the opposite one, while avoiding the six obstacles in this area. The system is observed through a set of 40×40 pixel images taken from the top view, which specifies the agent's location in the area. Actions are two-dimensional and specify the $x - y$ direction of the agent's movement, and given these actions the next position of the agent is generated by a deterministic underlying (unobservable) state evolution function. *Start State:* top-left corner. *Goal State:* one of three corners (excluding top-left corner). *Agent's Objective:* agent is within Euclidean distance of 5 from the goal state. ***For the biased-sampling variant of this experiment***, we uniformly sample a proportion, p, of the total samples within a 30×30 pixel region, which doesn't include any of the goal states, and the other $1 - p$ proportion of the samples are sampled uniformly from the entire underlying state space.

Inverted Pendulum – SwingUp This is the classic problem of controlling an inverted pendulum (Furuta et al., 1991) from 48×48 pixel images. The goal of this task is to swing up an under-actuated pendulum from the downward resting position (pendulum hanging down) to the top position and to balance it. The underlying state s_t of the system has two dimensions: angle and angular velocity, which is unobservable. The control (action) is 1-dimensional, which is the torque applied to the joint of the pendulum. For all PCC based algorithms, we opt to consider each observation x_t as two images generated from consecutive time-frames (the current time and the previous time; this was also done in the original PCC paper (Levine et al., 2020). This is because each image only shows the position of the pendulum and does not contain any information about the velocity. *Start State:* Pole is resting down, *Agent's Objective:* pole's angle is within $\pm\pi/6$ from an upright position. ***For the biased-sampling variant of this experiment***, we sample a proportion, p, of the total samples from when the pendulum is in it's closer to its resting position $[-\pi, -2.0] \cup [2, \pi]$ and the other $1 - p$ samples when the pendulum is within ± 0.5 from an upright position.

CartPole This is the visual version of the classic task of controlling a cart-pole system (Geva & Sitte, 1993). The goal in this task is to balance a pole on a moving cart, while the cart avoids hitting the left and right boundaries. The control (action) is 1-dimensional, which is the force applied to the cart. The underlying state of the system s_t is 4-dimensional, which indicates the angle and angular velocity of the pole, as well as the position and velocity of the cart. Similar to the inverted pendulum, in order to maintain the Markovian property the observation x_t is a stack of two 80×80 pixel images generated from consecutive time-frames. *Start State:* Pole is randomly sampled in $\pm\pi/6$. *Agent's Objective:* pole's angle is within $\pm\pi/10$ from an upright position. ***For the biased-sampling variant of this experiment***, we sample a proportion, p, of the total samples from when the pole's angle is sampled from $[-\pi/6, -\pi/10] \cup [\pi/10, \pi/6]$ and the other $1 - p$ samples are sampled as before uniformly from the given state space.

3-link Manipulator — SwingUp & Balance The goal in this task is to move a 3-link manipulator from the initial position (which is the downward resting position) to a final position (which is the top position) and balance it. In the 1-link case, this experiment is reduced to inverted pendulum. In the 2-link case the setup is similar to that of arcobot (Spong, 1995), except that we have torques applied to all intermediate joints, and in the 3-link case the setup is similar to that of the 3-link planar robot arm domain that was used in the E2C paper, except that the robotic arms are modeled by simple rectangular rods (instead of real images of robot arms), and our task success criterion requires both swing-up (manipulate to final position) and balance.[10] The underlying (unobservable) state s_t of the system is $2N$-dimensional, which indicates the relative angle and angular velocity at each link, and the actions are N-dimensional, representing the force applied to each joint of the arm. The state evolution is modeled by the standard Euler-Lagrange equations (Spong, 1995; Lai et al., 2015). Similar to the inverted pendulum and Cartpole, in order to maintain the Markovian property, the observation state x_t is a stack of two 80×80 pixel images of the N-link manipulator generated from consecutive time-frames. In the experiments we will evaluate the models based on the case of $N = 2$ (2-link manipulator) and $N = 3$ (3-link manipulator). *Start State:* 1st pole with angle π, 2nd pole with angle $2\pi/3$, and 3rd pole with angle $\pi/3$, where angle π is a resting position. *Agent's Objective:* the sum of all poles' angles is within $\pm\pi/6$ from an upright position. ***For the biased-sampling variant of this experiment***, we sample a proportion, p of the total samples of when the 1st pole is within $\pm\pi/2$, the 2nd pole is within angle $\pm\pi/3$, and the 3rd pole is within angle $\pm\pi/6$ of the upright position and the other $1 - p$ samples are sampled as before uniformly from the given state space.

E.2 Data Generation Procedure

For all algorithms that use the PCC framework for representation learning, we always start by sampling triplets of the form (x_t, a_t, x_{t+1}), which is done by (1) uniformly randomly sampling an underlying state s_t from the environment and creating the corresponding observation x_t, (2) uniformly randomly sampling a valid action a_t, and (3) obtaining the next state s_{t+1} through the environment's true dynamics and creating the corresponding observation x_{t+1}.

When interacting with the true underlying MDP, sampling the environment for more data for the iterative online variant of our algorithm, at iteration i of our algorithm, we are following our learned policy $\pi^{(i)}$. We start with an initial observation x_0 and generate our initial action a_0, $a_0 \sim \pi_\psi^{(i)}(E(\cdot|x_0))$ and continue following our learned policy $\pi^{(i)}$ to get our action a_j, $a_j \sim \pi_\psi^{(i)}(E(\cdot|x_j))$. We continue this process until we have reached the end of the episode or the pre-defined number of samples we draw from the environment.

In SOLAR and Dreamer each training sample is an episode $\{x_1, a_1, x_2, \cdots, x_T, a_T, x_{T+1}\}$, where T is the control horizon. We uniformly sample T actions from the action space, apply the dynamics T times, and generate the T corresponding observations.

E.3 Implementation of the Algorithms

In the following we describe architectures and hyper-parameters that were used for training the different algorithms.

E.3.1 Training Hyper-Parameters and Regulizers

SOLAR training specifics, we used their default setting:

- Batch size of 2.

[10]Unfortunately due to copyright issues, we cannot test our algorithms on the original 3-link planar robot arm domain.

- ADAM (Kingma & Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We also use a learning rate $\alpha_{model} = 2 \cdot 10^{-5} \times$horizon for the learning rate $\mathcal{MNIW}$ prior and $\alpha = 10^{-3}$ for other parameters.
- $\beta_{start}, \beta_{end}, \beta_{rate} = (10^{-4}, 10.0, 5 \cdot 10^{-5})$
- Local inference and control:
  - Data strength: 50
  - KL step: 2.0
  - Number of Iterations: 10

PCC training specifics, we use their reported optimal hyperparameters:

- Batch size of 128.
- ADAM with $\alpha = 5 \cdot 10^{-4}$, $\beta_1 = 0.9, \beta_2 = 0.999$, and $\epsilon = 10^{-8}$.
- L2 regularization with a coefficient of 10^{-3}.
- $(\lambda_p, \lambda_c, \lambda_{\text{cur}}) = (1, 7, 1)$, and the additive Gaussian noise in the curvature loss is $\mathcal{N}(0, \sigma^2)$, where $\sigma^2 = 0.01$.
- Additional VAE (Kingma & Welling, 2013) loss term $\ell_{\text{VAE}} = -\mathbb{E}_{q(z|x)}[\log p(x|z)] + D_{\text{KL}}(q(z|x)||p(z))$ with a very small coefficient of 0.01, where $p(z) = \mathcal{N}(0, 1)$.
- Additional deterministic reconstruction loss with coefficient 0.3: given the current observation x, we take the means of the encoder output and the dynamics model output, and decode to get the reconstruction of the next observation.

CARL training specifics:

- Batch size of 128.
- ADAM with $\alpha = 5 \cdot 10^{-4}$, $\beta_1 = 0.9, \beta_2 = 0.999$, and $\epsilon = 10^{-8}$.
- L2 regularization with a coefficient of 10^{-3}.
- The additive Gaussian noise in the curvature loss is $\mathcal{N}(0, \sigma^2)$, where $\sigma^2 = 0.01$.
- As in Levine et al. (2020), we use the deterministic reconstruction loss with coefficient 0.3.

E.3.2 Network Architectures

We next present the specific architecture choices for each domain. For fair comparison, The numbers of layers and neurons of each component were shared across all algorithms. ReLU non-linearities were used between each two layers.

Encoder: composed of a backbone (either a MLP or a CNN, depending on the domain) and an additional fully-connected layer that outputs mean variance vectors that induce a diagonal Gaussian distribution (for PCC, SOLAR, and all CARL variants).

Decoder: composed of a backbone (either a MLP or a CNN, depending on the domain) and an additional fully-connected layer that outputs logits that induce a Bernoulli distribution (for PCC, SOLAR, and all CARL variants)

Dynamical model: the path that leads from $\{z_t, a_t\}$ to $\hat{z}_{t+1}$. Composed of a MLP backbone and an additional fully-connected layer that outputs mean and variance vectors that induce a diagonal Gaussian distribution (for PCC, SOLAR, and all CARL variants).

Backwards dynamical model: the path that leads from $\{\hat{z}_{t+1}, a_t, x_t\}$ to z_t. Each of these inputs goes to fully-connected layer of $\{N_z, N_u, N_x\}$ neurons respectively. These outputs are then concatenated and passed through another layer of N_{joint} neurons, and finally with an additionally fully-connected layer we output the mean and variance vectors that induce a diagonal Gaussian distribution.

SAC Architecture: For all of our environments with all CARL algorithms, we utilized the same SAC architecture as seen in Table 2:

Hyper Parameters for SAC	**Value(s)**
Discount Factor	0.99
Critic Network Architecture	MLP with 2 hidden layers of size 256
Actor Network Architecture	MLP with 2 hidden layers of size 256
Exploration policy	$\mathcal{N}(0, \sigma = 1)$
Exploration noise (σ) decay	0.999
Exploration noise (σ) minimum	0.025
Temperature	0.99995
Soft target update rate (τ)	0.005
Replay memory size	10^6
Minibatch size	128
Number of Rollouts in the Latent space, k in (26)	5
Critic learning rate	0.001
Actor learning rate	0.0005
Neural network optimizer	Adam

Table 2: Hyper-parameters for the SAC controller.

Planar system

- **Input:** 40×40 images.
- **Actions space:** 2-dimensional
- **Latent space:** 2-dimensional
- **Encoder:** 3 Layers: 300 units - 300 units - 4 units (2 for mean and 2 for variance)
- **Decoder:** 3 Layers: 300 units - 300 units - 1600 units (logits)
- **Dynamics:** 3 Layers: 20 units - 20 units - 4 units
- **Backwards dynamics:** $N_z = 5, N_a = 5, N_x = 100$ - $N_{\text{joint}} = 100$ - 4 units
- **Number of control actions:** or the planning horizon $T = 40$
- **Offline and Online CARL hyperparameters:** $\lambda_{ed} = 0.01, \lambda_p = 1, \lambda_c = 7, \lambda_{cur} = 1$
- **Value-Guided CARL hyperparameters:** $\lambda_{ed} = 0.01, \lambda_p = 2, \lambda_c = 11, \lambda_{cur} = 1, \tau = 1/30.0$
- **Proportion of biased samples:** $p = 0.5$
- **Number of samples from the environment per iteration i in Algorithm 2**: 128
- **Initial standard deviation for collecting data (SOLAR):** 1.5 for both global and local training.

Inverted Pendulum – SwingUp

- **Input:** Two 48×48 images.
- **Actions space:** 1-dimensional
- **Latent space:** 3-dimensional
- **Encoder:** 3 Layers: 500 units - 500 units - 6 units (3 for mean and 3 for variance)
- **Decoder:** 3 Layers: 500 units - 500 units - 4608 units (logits)
- **Dynamics:** 3 Layers: 30 units - 30 units - 6 units
- **Backwards dynamics:** $N_z = 10, N_a = 10, N_x = 200$ - $N_{\text{joint}} = 200$ - 6 units
- **Number of control actions:** or the planning horizon $T = 400$
- **Offline and Online CARL environment hyperparameters:** $\lambda_{ed} = 0.01, \lambda_p = 1, \lambda_c = 11, \lambda_{cur} = 1$
- **Value-Guided CARL environment hyperparameters:** $\lambda_{ed} = 0.01, \lambda_p = 1, \lambda_c = 7, \lambda_{cur} = 1, \tau = 1/60.0$
- **Proportion of biased samples:** $p = 0.95$
- **Number of samples from the environment per iteration i in Algorithm 2**: 128

- **Initial standard deviation for collecting data (SOLAR):** 0.5 for both global and local training.

Cart-pole – Balancing

- **Input:** Two 80×80 images.
- **Actions space:** 1-dimensional
- **Latent space:** 8-dimensional
- **Encoder:** 6 Layers: Convolutional layer: $32 \times 5 \times 5$; stride $(1, 1)$ - Convolutional layer: $32 \times 5 \times 5$; stride $(2, 2)$ - Convolutional layer: $32 \times 5 \times 5$; stride $(2, 2)$ - Convolutional layer: $10 \times 5 \times 5$; stride $(2, 2)$ - 200 units - 16 units (8 for mean and 8 for variance)
- **Decoder:** 6 Layers: 200 units - 1000 units - 100 units - Convolutional layer: $32 \times 5 \times 5$; stride $(1, 1)$ - Upsampling $(2, 2)$ - convolutional layer: $32 \times 5 \times 5$; stride $(1, 1)$ - Upsampling $(2, 2)$ - Convolutional layer: $32 \times 5 \times 5$; stride $(1, 1)$ - Upsampling $(2, 2)$ - Convolutional layer: $2 \times 5 \times 5$; stride $(1, 1)$
- **Dynamics:** 3 Layers: 40 units - 40 units - 16 units
- **Backwards dynamics:** $N_z = 10, N_a = 10, N_x = 300$ - $N_{\text{joint}} = 300$ - 16 units
- **Number of control actions:** or the planning horizon $T = 200$
- **Offline and Online CARL environment hyperparameters:** $\lambda_{ed} = 0.01, \lambda_p = 1, \lambda_c = 7, \lambda_{cur} = 1$
- **Value-Guided CARL environment hyperparameters:** $\lambda_{ed} = 0.01, \lambda_p = 1, \lambda_c = 7, \lambda_{cur} = 1, \tau = 1/40.0$
- **Proportion of biased samples:** $p = 0.8$
- **Number of samples from the environment per iteration i in Algorithm 2**: 256
- **Initial standard deviation for collecting data (SOLAR):** 10 for global and 5 for local training.

3-link Manipulator — Swing Up & Balance

- **Input:** Two 80×80 images.
- **Actions space:** 3-dimensional
- **Latent space:** 8-dimensional
- **Encoder:** 6 Layers: Convolutional layer: $32 \times 5 \times 5$; stride $(1, 1)$ - Convolutional layer: $32 \times 5 \times 5$; stride $(2, 2)$ - Convolutional layer: $32 \times 5 \times 5$; stride $(2, 2)$ - Convolutional layer: $10 \times 5 \times 5$; stride $(2, 2)$ - 500 units - 16 units (8 for mean and 8 for variance)
- **Decoder:** 6 Layers: 200 units - 1000 units - 100 units - Convolutional layer: $32 \times 5 \times 5$; stride $(1, 1)$ - Upsampling $(2, 2)$ - convolutional layer: $32 \times 5 \times 5$; stride $(1, 1)$ - Upsampling $(2, 2)$ - Convolutional layer: $32 \times 5 \times 5$; stride $(1, 1)$ - Upsampling $(2, 2)$ - Convolutional layer: $2 \times 5 \times 5$; stride $(1, 1)$
- **Dynamics:** 3 Layers: 40 units - 40 units - 16 units
- **Backwards dynamics:** $N_z = 10, N_a = 10, N_x = 400$ - $N_{\text{joint}} = 400$ - 16 units
- **Number of control actions:** or the planning horizon $T = 200$
- **Offline and Online CARL environment hyperparameters:** $\lambda_{ed} = 0.01, \lambda_p = 1, \lambda_c = 11, \lambda_{cur} = 1$
- **Value-Guided CARL environment hyperparameters:** $\lambda_{ed} = 0.01, \lambda_p = 2, \lambda_c = 11, \lambda_{cur} = 1, \tau = 1/60.0$
- **Proportion of biased samples:** $p = 0.2$
- **Number of samples from the environment per iteration i in Algorithm 2**: 128
- **Initial standard deviation for collecting data (SOLAR):** 1 for global and 0.5 for local training.

F Additional Experiments

F.1 Policy Distillation

In our iterative algorithm, we describe a method to connect policies from two different latent spaces in equation 8. In Figure 4, we show the learning curves for Online CARL with and without policy distillation. In general, when utilizing policy distillation, we achieve similar performance to the iterative variant of our algorithm. Additionally, these results show that with policy distillation, in the three-pole and swingup tasks we are able to achieve faster convergence. Another observed added benefit is that with policy distillation we achieve more stability in the final metrics as we add more samples form our environment across environments.

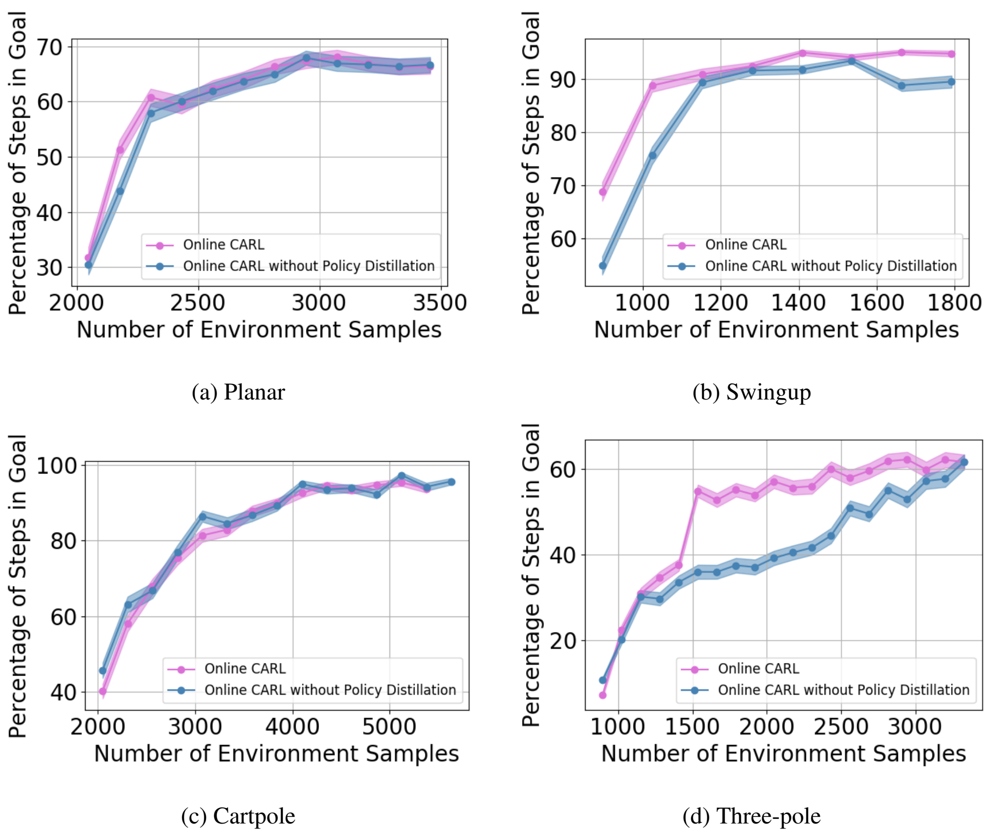


Figure 4: Training curves comparing our online algorithm with and without policy distillation on continuous control environments. The solid curves depict the mean of the experiments and the standard deviations correspond to the standard deviation of the means.

F.2 Ablation Study

In order to better understand different components of the CARL loss function in Eq. 6, we conducted an ablation study over various components of the loss and present the results in Figure F.2.

All the results are generated with the online CARL algorithm for which the hyper-parameters are consistent across all experiments (with the exception on the parameters removed in the loss function due to the ablation). In general our results show a similar trend to that presented in Levine et al. (2020), in which the prediction and consistency loss components are important in guaranteeing good learning performance. On the contrary, both the curvature loss and the encoding-decoding regularization loss in CARL play relatively minor roles in guaranteeing the algorithm's performance. Different from PCC, in which a low-curvature latent dynamics clearly improves the learning of the iLQR control algorithm, in CARL the curvature loss might have a less significant role, because our control algorithm (i.e., soft actor critic) does not necessarily require the (latent) state evolution to be locally linear.

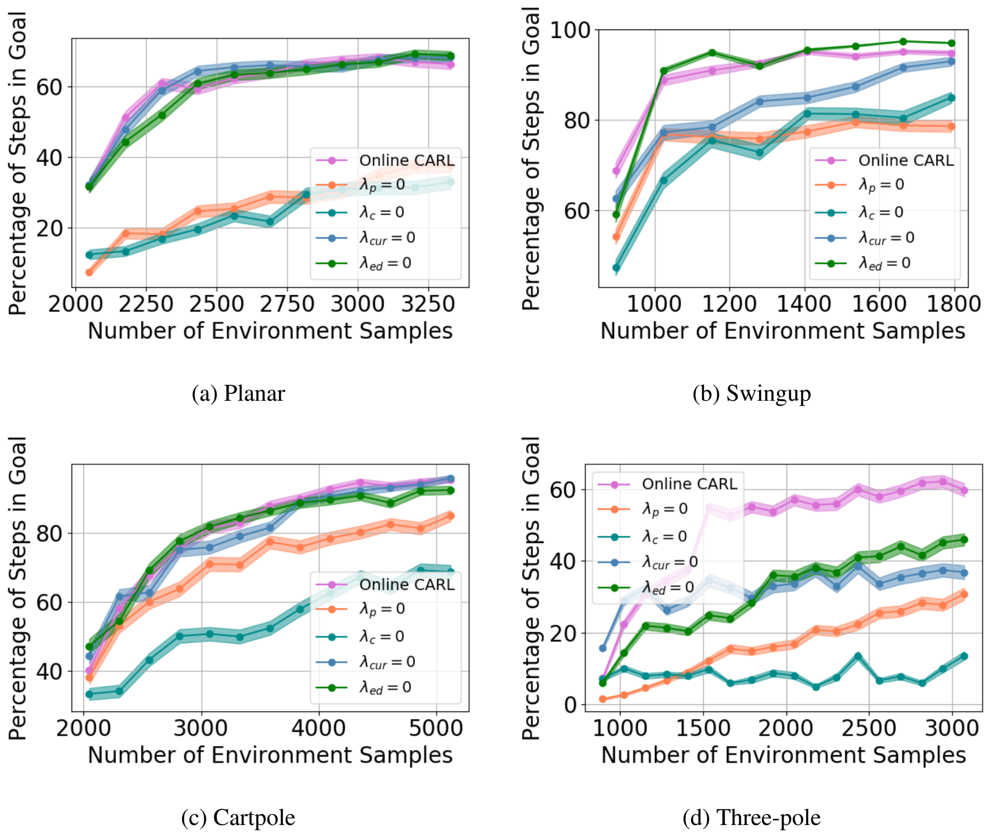


Figure 5: An ablation study on different components of the CARL training loss function

F.3 More Comparisons between PCC and Offline CARL

All of the following figures were trained using either the PCC or CARL framework. We present 5 representations with the worst control performance (Figure 7) and 5 representations that had the best control performance (Figure 8), with PCC. Additionally, we present 5 representations that performed the worst (Figure 9) and 5 representations that performed the best (Figure 10) with offline CARL. We also present a reference figure in Figure 6. These maps were generated by uniformly sampling a state s from the underlying environment, creating a corresponding observation x, and using the encoder create the latent representation $z = \mathbb{E}[E(\cdot|x)]$. All of the latent maps presented in Figures 6-9 were generated with the same hyperparameters.

The latent maps from the best performing representations are all fairly similar and look similar to the reference figure. Additionally, the worst performing maps have some similarities e.g. twisting or folding of the latent space. It is important to note that even though these latent maps are similar, it is clear that there is a large difference in performance in table 3. Importantly, from the visual representations, it is clear that iLQR struggles significantly more than SAC in these non-linear environments as seen in the worst case performance and the corresponding latent maps, where the latent maps contain additional twisting or curvature resulting in poorer performance.

In this case it is obvious that control in several of the latent representations that performed poorly would be difficult as there are regions that are highly non-smooth, non-locally-linear; thus, a locally linear controller such as iLQR is likely to perform poorly. We compare the top and worst 5 representations trained using the PCC framework, with the only difference being the controller (SAC vs. iLQR) in table 3. For almost all tasks, Offline CARL performs better for the worst 5 average results and the top 5 average results, with the exception of the worst case swingup results.

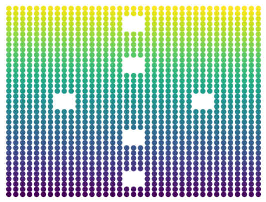


Figure 6: True underlying state representation for the Planar MDP.

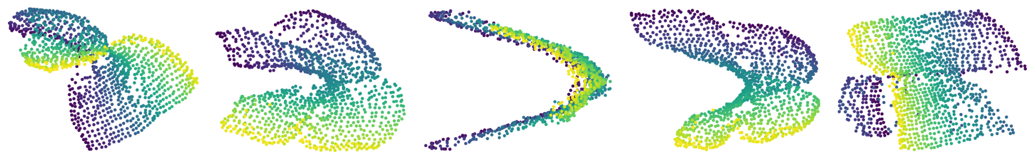


Figure 7: Latent maps for the 5 worst performing representations on average using PCC.

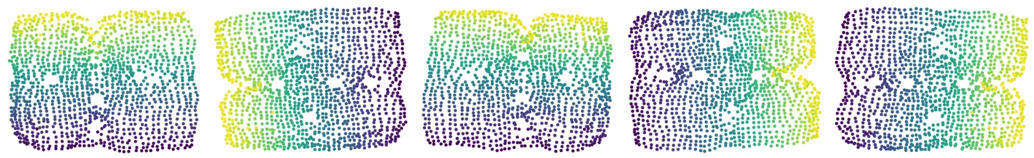


Figure 8: Latent maps for the 5 best performing representations on average using PCC.

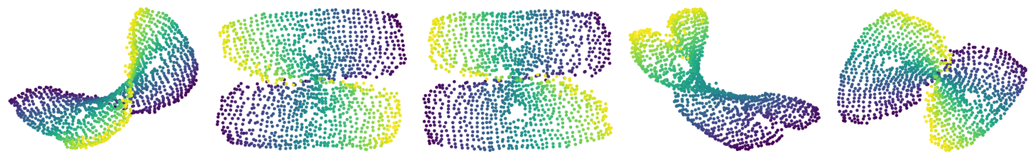


Figure 9: Latent maps for the 5 worst performing representations on average using Offline CARL.

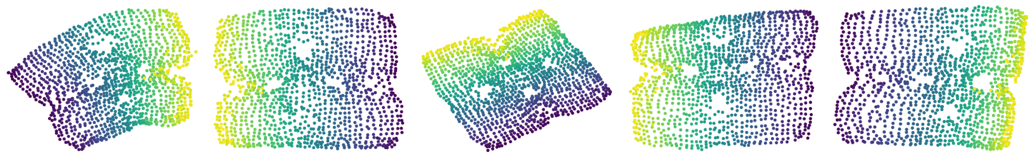


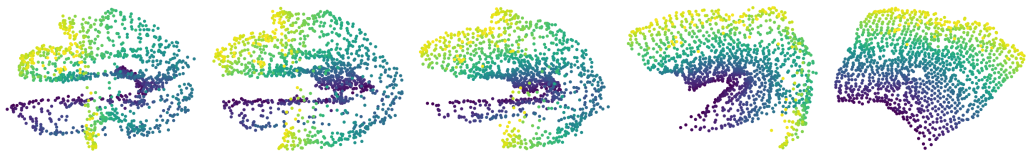
Figure 10: Latent maps for the 5 best performing representations on average using Offline CARL.

Environment	**Algorithm**	**Worst 5 Avg. Results**	**Top 5 Avg. Results**
Planar	PCC	6.15 ± 2.89	62.25 ± 4.45
Planar	Offline CARL	**33.46** $\pm$ **4.61**	**78.87** $\pm$ **0.19**
Swingup	PCC	**68.07** $\pm$ **3.49**	95.71 ± 0.38
Swingup	Offline CARL	57.22 ± 3.71	**98.50** $\pm$ **0.0**
Cartpole	PCC	50.98 ± 5.44	99.85 ± 0.08
Cartpole	Offline CARL	**74.44** $\pm$ **5.28**	**100.0** $\pm$ **0.0**
Three-pole	PCC	0 ± 0	18.42 ± 2.98
Three-pole	Offline CARL	**6.17** $\pm$ **1.71**	**85.77** $\pm$ **0.23**

Table 3: Percentage of steps in goal state; averaged over the 5 worst models and the 5 best models.

F.4 Evolving Latent Space Maps for Online CARL

In Figure 11 we show how the latent space for online CARL in the planar case evolves as we increase the number of iterations in online CARL. The final map looks fairly similar to the reference figure.



(a) $i = 1$ (b) $i = 3$ (c) $i = 6$ (d) $i = 10$ (e) $i = 14$

Figure 11: Evolution of the latent representation of the Planar problem learned by online CARL. Here i represents the number of LCE model-learning episodes (Algorithm 2 in Appendix D).

F.5 Additional SOLAR Experiments

In our experiments with Planar, Swingup, and Cartpole, we start from a point randomly chosen from a region surrounding the start point in the underlying MDP. Additionally in Planar, we randomize the target at every episode. In Table 4, we present results where we fix the start and goal states, to see if there is an improvement in the SOLAR's performance. We also shorten the horizon for Swingup to 100 to see if long horizon has an effect on the SOLAR's performance. We do not present any new results on 3-pole task as the starting and goal states were already fixed in this problem. The results in Table 4 indicate a dramatic improvement for Planar when we fix the start and goal states, and a modest improvement in Cartpole and Swingup. However, these SOLAR results are still incomparable to the performance of any of CARL variants and Dreamer.

Environment	**Algorithm**	**Number of Samples**	**Avg Result**	**Best Result**
Planar	SOLAR	5000 (VAE) + 40000 (Control)	26.70 ± 5.92	41 ± 7.28
Cartpole	SOLAR	10000 (VAE) + 40000 (Control)	14.60 ± 1.70	20.05 ± 2.91
Swingup	SOLAR	20000 (VAE) + 40000 (Control)	22.40 ± 3.07	34.03 ± 2.09

Table 4: Percentage of steps in goal state; averaged over all models and the best model. Additionally, the number of samples used for training for SOLAR are under the condition that there is the same start and same goal state for all episodes.

F.6 Dreamer: Implementation and Additional Results

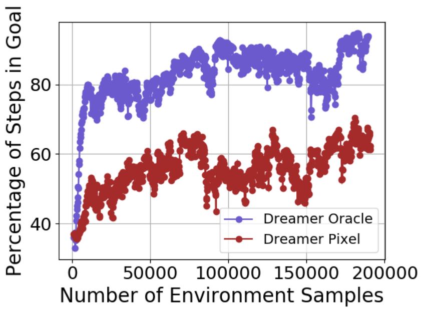
Dreamer Implementation. For dreamer we used the default parameters presented in Hafner et al. (2020a), so all images were presented as (64, 64, 3) images. The only change was for Planar, where a sequence length of 20 was used, otherwise, all other sequence lengths were 50. Different from CARL and other LCE methods, which can directly use the latent distance to goal as the reward to learn the RL policy, in Dreamer this reward construction becomes non-trivial as the encoder is modeled with an RNN that takes an observation trajectory as the input. Therefore, to provide reward samples for the Dreamer training procedure we construct the several environment-based rewards, whose implementation details are given below:

Pixel-based: We use the negative distance between the current observation x_t and the goal observation x_g, i.e., $-||x_t - x_g||_2^2$, as the reward signal that Dreamer uses as input to learn the reward in the resulting latent space.

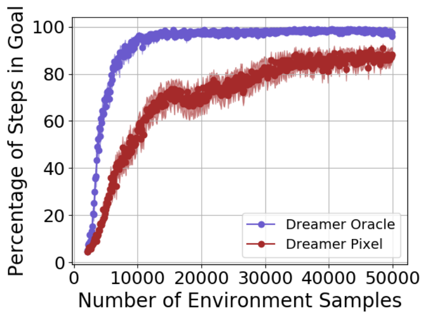
Oracle-based: We use the negative distance between the current (unobserved) state s_t and the goal state s_g, i.e., $-||s_t - s_g||_2^2$, as the reward signal that Dreamer uses as input to learn the reward in the resulting latent space. To facilitate learning in Dreamer in some domains, we only use a subset of the state variables (e.g., only the angle and not the angular velocity in Swingup) to define this reward function.

Since CARL algorithms only assume access to the observation space $\mathcal{X}$, and not to the underlying state space $\mathcal{S}$, it is more fair to compare them with Dreamer-Pixel, and comparison with Dreamer-Oracle is definitely in favor of Dreamer.

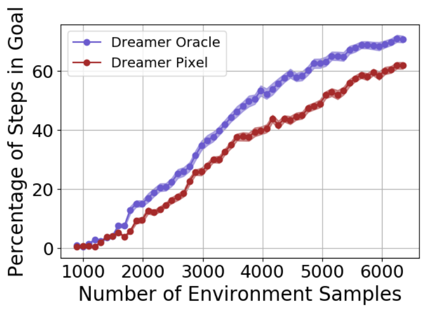
Additional Results. In Fig. 12, we plot dreamer with more environment samples. We see that in general, dreamer will converge for most of our environments, but it takes significantly more time-steps (90x, 3x, 2x) for swingup, cartpole, and three-pole respectively with the oracle-based dreamer. The pixel based rewards for dreamer also leads to generally slower convergence for dreamer, and in the case of the swingup task leads to supoptimal results. We also opt to not present results of Dreamer on Planar as we were unable to obtain good results for Dreamer on this task when the initial and goal states are randomly sampled.



(a) Swingup



(b) Cartpole



(c) Three-pole

Figure 12: Training curves for both implementations of Dreamer with significantly longer timesteps. Almost all oracle implementations converge, but the pixel based reward can lead to suboptimal results.