Learning Representations via a Robust Behavioral Metric for Deep Reinforcement Learning

Jianda Chen
Nanyang Technological University, Singapore
jianda001@e.ntu.edu.sg

Sinno Jialin Pan
Nanyang Technological University, Singapore
sinnopan@ntu.edu.sg

Abstract

Learning an informative representation with behavioral metrics is able to accelerate the deep reinforcement learning process. There are two key research issues on behavioral metric-based representation learning: 1) how to relax the computation of a specific behavioral metric, which is difficult or even intractable to compute, and 2) how to approximate the relaxed metric by learning an embedding space for states. In this paper, we analyze the potential relaxation and/or approximation gaps for existing behavioral metric-based representation learning methods. Based on the analysis, we propose a new behavioral distance, the RAP distance, and develop a practical representation learning algorithm on top of it with a theoretical analysis. We conduct extensive experiments on DeepMind Control Suite with distraction, Robosuite, and autonomous driving simulator CARLA to demonstrate new state-of-the-art results.

1 Introduction

Deep reinforcement learning (RL) aims to interactively learn an optimal policy from high-dimensional environmental observations or states in an end-to-end manner. In the literature, it has been demonstrated that a robust representation of states, which are task-relevant and invariant to task-irrelevant background information, is able to significantly speed up the RL process and makes the learned policy more generalizable. Therefore, representation learning has played a key role in Deep RL algorithms and attracts more and more attention in the RL community [4, 23, 17, 20].

Prior work on representation learning is focused on learning embeddings to represent states based on a reconstruction loss [26, 25, 13]. Though promising results have been reported on some RL application domains, policies learned with such representations usually fail to generalize well in a complex environment because minimizing a reconstruction loss may potentially introduce local (visual) features with task-irrelevant information. Another research direction is to apply data augmentation techniques to learn robust state representations for an RL agent [19, 28, 27]. While data augmentation-based approaches may be able to learn more robust feature representations, they do not take the characteristics of Markov Decision Processes (MDPs), e.g., reward signals and dynamics models, into consideration when learning representations. As a result, the learned representations may not be informative for learning an optimal policy for the task of interest. A third research direction is to construct some auxiliary tasks in addition to the prime RL task and learn state representations by learning all the tasks simultaneously [14, 2, 21]. These approaches can be considered as shaping the state representations implicitly by learning the auxiliary tasks, which do not have a guarantee for learning a better policy, especially when the auxiliary tasks fail to benefit the learning of the RL task.

Recently, behavioral metrics, such as the bisimulation metric and its variants [7, 8, 3], have been exploited in representation learning for deep RL. Behavioral metrics are originally proposed to measure the behavioral similarity between states in terms of rewards and dynamics models, e.g, state transition probabilities. The high-level idea is to learn state embeddings by preserving the behavioral

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

similarity between states based on a specific behavioral metric [29, 1, 16, 4]. As behavioral metrics provide a theoretical bound on the difference between the outputs of value function of a pair of states, the learned representation enjoys a theoretical guarantee to capture behavioral structure in the environment for policy learning. However, as behavioral metrics are expensive or even intractable to compute, different approximation approaches and learning objectives have been proposed to make behavioral metric-based representation learning for RL agents more efficient [29, 1, 16, 4].

Though behavioral metric-based representation learning methods have achieved state-of-the-art results on some benchmark RL problems, they suffer from at least one of the following three issues: loss function mismatch, relaxation of dynamics model divergence and the L_1/L_2 norm limitation.

- **Loss function mismatch**. In behavioral metric-based representation learning, a loss function is to measure the difference between the distance between states on the embedding space and their behavioral metric. As behavioral metrics are expensive or even intractable to computers, approximation or relaxation are necessary. However, based on our analysis, state-of-the-art methods adopt improper approximations, which may introduce a bias and make the bound of the value function looser.
- **Relaxation of dynamics model divergence**. The bisimulation metric or its on-policy invariant is one of the most widely used behavioral metrics, which requires estimating the 1-Wasserstein distance between dynamics models. As the 1-Wasserstein distance is usually difficult or intractable to estimate, prior methods propose some relaxations to replace the estimation of the 1-Wasserstein, which may break some theoretical guarantees of the bisimulation metric.
- **The L_1/L_2 norm limitation**. The L_1 and the L_2 norms are commonly-used distances with zero self-distance. However, due to the two gaps mentioned above, the approximations or relaxations of behavior metrics in prior approaches are potentially non-zero self-distance. Using the L_1 or L_2 distance on embedding space is difficult to learn robust representation to preserve the behavioral similarity between states.

To address the aforementioned issues or gaps, in this paper, we first introduce a new behavior metric namely the *Reducing Approximation Gap* (RAP) distance, and then develop a practical approximation algorithm with consistency to its theoretical prototype. In this way, our algorithm is guaranteed to learn a robust state representation to capture the behavioral similarity between states. We conduct extensive experiments on DeepMind Control Suite (DMC) [24] with background distraction, Robosuite [30] and autonomous driving simulator CARLA [6] to demonstrate new state-of-the-art results compared with other behavior metric-based representation learning methods.

The contributions of this paper are two-fold: 1) we analyze the potential approximation gaps for existing behavioral metric-based representation learning approaches, and 2) we introduce a new behavior distance RAP and develop its practical approximation algorithm with theoretical guarantees.

2 Related Work

In RL, early research work has focused on learning state representations by designing and optimizing some auxiliary objectives in addition to the RL task of interest. For instance, Hafner et al. [11] propose to learn a dynamics model to predict future states with a reconstruction loss. Gelada et al. [9] aim to learn state representations by predicting a dynamics model and a reward function on an embedding space. Laskin et al. [17] apply contrastive learning with samples generated by the momentum encoder. Hansen et al. [12] predict an inverse model as an auxiliary task for representation learning. Another type of approaches aims to apply data augmentation techniques to improve representation learning. For instance, Laskin et al. [18] conduct an extensive study of data augmentation for deep RL with pixel-based input. Yarats et al. [28] adopt random crops on pixel-based input and add regularization terms on the Q-function objectives. Lee et al. [19] introduce convolutional neural networks with randomized parameters. Stooke et al. [23] use augmented samples for representation contrastive learning. In contrast to these works, our proposed method aims to encode state representations into a structural metric space based on behavioral metrics.

Recently, behavioral metric-based representation learning has attracted more and more attention in the RL community. For instance, Zhang et al. [29] aim to learn representations by approximating the bisimulation metric [7, 3] on an embedding space. Agarwal et al. [1] propose a behavioral

metric considering a distance between action distributions given states for representation learning. Kemertas and Aumentado-Armstrong [16] propose to improve the robustness of the representation learning method proposed in [29] by adding norm constraints on the embedding space and intrinsic rewards. Castro et al. [4] introduce a new behavioral distance and develop a sampling-based approach to preserve the behavioral similarity between states on the embedding space. Chen and Pan [5] propose to learn neural networks to approximate components in the bisimulation metric on the state embedding space.

3 Preliminaries

Reinforcement Learning We consider a Markov Decision Process (MDP) defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma \rangle$, where $\mathcal{S}$ is the high-dimensional state space, $\mathcal{A}$ is the action space, $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ is the transition distribution that captures the probability of entering a next state $\mathbf{s}' \in \mathcal{S}$ given a current state $\mathbf{s} \in \mathcal{S}$ and an action $a \in \mathcal{A}$, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function and $\gamma \in [0, 1)$ is the discount factor. In the sequel, we use $P^a_{\mathbf{s}}$ and $r^a_{\mathbf{s}}$ to denote $P(\cdot|\mathbf{s}, a)$ and $\mathcal{R}(\mathbf{s}, a)$, respectively. A policy $\pi(a|\mathbf{s})$ is a probability distribution over each action a conditioned on a state $\mathbf{s}$. The value function $V^\pi : \mathcal{S} \to \mathbb{R}$ for a given policy π at a state $\mathbf{s}$ is defined as the expected sum of discounted future rewards,

$$V^\pi(\mathbf{s}) = \mathop{\mathbb{E}}_{\substack{\mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t) \\ \mathbf{s}_{t+1} \sim P^{\mathbf{a}_t}_{\mathbf{s}_t}}} \left[\sum_{t=0}^{\infty} \gamma^t r^{\mathbf{a}_t}_{\mathbf{s}_t} \mid \mathbf{s}_0 = \mathbf{s} \right].$$

The goal of reinforcement learning is to find an optimal policy $\pi^* = \arg\max_\pi V^\pi$ that maximizes the expected future rewards. In the scope of representation learning for deep RL, a state encoder $\phi : \mathcal{S} \to \mathbb{R}^n$ maps a high-dimensional state to a low-dimensional vector, with which a policy $\pi(a|\phi(\mathbf{s}))$ is learned.

Bisimulation Metrics The bisimulation metric [7, 8] defines a pseudometric $d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ to measure the behavioral distance between states. Recently, a variant of the bisimulation metric, *on-policy* bisimulation metric (or π-bisimulation metric) is proposed [3], which focuses on behaviors relative to a particular policy π. The π-bisimulation metric consists of a reward difference term and a Wasserstein distance in dynamics models between states.

Theorem 3.1 (π-bisimulation metric [3])**.** *Let $\mathbb{M}$ be the set of all pseudometrics on space $\mathcal{S}$. A pseudometric transformation function $\mathcal{F}^\pi_B : \mathbb{M} \to \mathbb{M}$ is defined as,*

$$\mathcal{F}^\pi_B(d)(\mathbf{s}_i, \mathbf{s}_j) = |\, \mathbb{E}_{\mathbf{a}_i \sim \pi}\, r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j \sim \pi}\, r^{\mathbf{a}_j}_{\mathbf{s}_j} | + \gamma W_1(d)(P^\pi_{\mathbf{s}_i}, P^\pi_{\mathbf{s}_j}) \tag{1}$$

where $\mathbb{E}_{\mathbf{a}_i \sim \pi}\, r^{\mathbf{a}_i}_{\mathbf{s}_i} = \mathbb{E}_{\mathbf{a}_i \sim \pi(\cdot|\mathbf{s}_i)}\, r^{\mathbf{a}_i}_{\mathbf{s}_i}$, $P^\pi_{\mathbf{s}_i} = \mathbb{E}_{a \sim \pi(\cdot|\mathbf{s}_i)}\, P^a_{\mathbf{s}_i}$ and W_1 is the 1-Wasserstein distance. $\mathcal{F}^\pi_B$ has a least fixed point d^π_B and d^π_B is a π-bisimulation metric.

The following theorem shows that the difference in value function is bounded by d^π_B.

Theorem 3.2 (Value difference bound [3])**.** *Given states $\mathbf{s}_i$ and $\mathbf{s}_j$, and policy π, we have*

$$|V^\pi(\mathbf{s}_i) - V^\pi(\mathbf{s}_j)| \leq d^\pi_B(\mathbf{s}_i, \mathbf{s}_j). \tag{2}$$

MICo distance The MICo distance [4] is a variant of the π-bisimulation metric, which measures the distribution distance between dynamics models by computing the distance between sampled next states from the dynamics models in order to avoid the computation of the Wasserstein distance.

Theorem 3.3 (MICo distance [4])**.** *Let $\mathbb{M}$ be the space of distance function $d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$, if the MICo metric function $\mathcal{F}^\pi_M : \mathbb{M} \to \mathbb{M}$ is defined as,*

$$\mathcal{F}^\pi_M(d)(\mathbf{s}_i, \mathbf{s}_j) = |\, \mathbb{E}_{\mathbf{a}_i \sim \pi}\, r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j \sim \pi}\, r^{\mathbf{a}_j}_{\mathbf{s}_j} | + \gamma \mathop{\mathbb{E}}_{\substack{\mathbf{s}'_i \sim P^\pi_{\mathbf{s}_i} \\ \mathbf{s}'_j \sim P^\pi_{\mathbf{s}_j}}} d(\mathbf{s}'_i, \mathbf{s}'_j), \tag{3}$$

then $\mathcal{F}^\pi_M$ has a unique fixed point d^π_M.

Theorem 3.4 (Value difference bound [4])**.** *Given states $\mathbf{s}_i$ and $\mathbf{s}_j$, and policy π, we have*

$$|V^\pi(\mathbf{s}_i) - V^\pi(\mathbf{s}_j)| \leq d^\pi_M(\mathbf{s}_i, \mathbf{s}_j). \tag{4}$$

4 Approximation Gaps in Behavioral Metric-based Representation Learning

The high-level idea of behavioral metric-based representation learning is to learn an embedding space such that after mapping states onto the embedding space, the behavioral similarity can be preserved. We denote the state encoder by $\phi_\omega : \mathcal{S} \to \mathbb{R}^n$ with parameters ω and the distance between states on the embedding space $\mathbb{R}^n$ by $\hat{d}(\phi_\omega(\mathbf{s}_i), \phi_\omega(\mathbf{s}_j))$, e.g., $\hat{d}$ can be the L_2 norm distance. The problem of representation learning in terms of ω can be cast as a minimization problem of the following expected squared difference or loss between the distance on the embedding space, $\hat{d}(\phi_\omega(\mathbf{s}_i), \phi_\omega(\mathbf{s}_j))$, and the corresponding behavior metric, $d^\pi(\mathbf{s}_i, \mathbf{s}_j)$, between any pair of states $\mathbf{s}_i$ and $\mathbf{s}_j$:

$$\mathcal{L}(\phi_\omega) = \mathbb{E}\left[\left(\hat{d}(\phi_\omega(\mathbf{s}_i), \phi_\omega(\mathbf{s}_j)) - d^\pi(\mathbf{s}_i, \mathbf{s}_j)\right)^2\right]. \tag{5}$$

To develop a practical algorithm to minimize the above objective, prior approaches adopt different approximation or relaxation strategies to make the resultant optimization problem computationally tractable. As discussed above, there are three common issues underlying most prior approaches, which introduce gaps between the practical algorithms and their theoretical prototypes, i.e., (5), namely loss function mismatch, relaxation of dynamics model divergence, and the L_1 or the L_2 norm limitation. We discuss these 3 gaps in detail in the following sections.

4.1 Loss Function Mismatch

In general, to learn the encoder ϕ_ω by minimizing (5) is computationally intractable or expensive because of the computation of the behavior metric d^π. We take MICo (MICo-based representation learning [4]) as an example. By specifying the term $d^\pi(\mathbf{s}_i, \mathbf{s}_j)$ in (5) as the MICo distance defined in (3), the loss of MICo becomes,

$$\mathcal{L}(\phi_\omega) = \mathbb{E}\left[\left(\hat{d}(\phi_\omega(\mathbf{s}_i), \phi_\omega(\mathbf{s}_j)) - \left|\mathbb{E}_{\mathbf{a}_i\sim\pi}\, r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j\sim\pi}\, r^{\mathbf{a}_j}_{\mathbf{s}_j}\right| - \gamma\mathbb{E}_{\substack{\mathbf{s}'_i\sim P^\pi_{\mathbf{s}_i}\\ \mathbf{s}'_j\sim P^\pi_{\mathbf{s}_j}}} \hat{d}(\phi_{\bar{\omega}}(\mathbf{s}_i), \phi_{\bar{\omega}}(\mathbf{s}_j))\right)^2\right], \tag{6}$$

where $\bar{\omega}$ is a copy of parameters for the target network. However, the reward expectations $\mathbb{E}_{\mathbf{a}_i\sim\pi}\, r^{\mathbf{a}_i}_{\mathbf{s}_i}$ and $\mathbb{E}_{\mathbf{a}_j\sim\pi}\, r^{\mathbf{a}_j}_{\mathbf{s}_j}$ in the 2nd term in (6) are computationally intractable and also difficult to estimate even based on sampling. Thus, Castro et al. [4] propose to approximate the loss in (6) with the following alternative loss,

$$\mathcal{L}(\phi_\omega) = \mathop{\mathbb{E}}_{\substack{\mathbf{a}_i\sim\pi, \mathbf{a}_j\sim\pi\\ \mathbf{s}'_i\sim P^{\mathbf{a}_i}_{\mathbf{s}_i}, \mathbf{s}'_j\sim P^{\mathbf{a}_j}_{\mathbf{s}_j}}} \left[\left(\hat{d}(\phi_\omega(\mathbf{s}_i), \phi_\omega(\mathbf{s}_j)) - \left|r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}\right| - \gamma\hat{d}(\phi_{\bar{\omega}}(\mathbf{s}_i), \phi_{\bar{\omega}}(\mathbf{s}_j))\right)^2\right]. \tag{7}$$

Note that in (7), the expectation operator on rewards is moved out of the absolute value operator of the difference between rewards to avoid the estimation of expectation over rewards, $\mathbb{E}_{\mathbf{a}_i\sim\pi}\, r^{\mathbf{a}_i}_{\mathbf{s}_i}$ and $\mathbb{E}_{\mathbf{a}_j\sim\pi}\, r^{\mathbf{a}_j}_{\mathbf{s}_j}$, and enable sampling more efficient. However, such a revision introduces a gap between solutions of minimizing (7) and (6), because in (7) the reference behavioral metric is no longer the MICo distance but the "shift" MICo distance defined as follows.

Definition 4.1 (Shift MICo distance)**.** The shift MICo distance function $\tilde{\mathcal{F}}^\pi_M$ is defined as

$$\tilde{\mathcal{F}}^\pi_M(d)(\mathbf{s}_i, \mathbf{s}_j) = \mathop{\mathbb{E}_{\mathbf{a}_i\sim\pi}}_{\mathbf{a}_j\sim\pi} |r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}| + \gamma\mathbb{E}_{\substack{\mathbf{s}'_i\sim P^\pi_{\mathbf{s}_i}\\ \mathbf{s}'_j\sim P^\pi_{\mathbf{s}_j}}} d(\mathbf{s}'_i, \mathbf{s}'_j).$$

Lemma 4.2 (Fixed-point)**.** *The shift MICo distance function $\tilde{\mathcal{F}}^\pi_M$ has a unique fixed-point $\tilde{d}^\pi_M$.*

Proof. (Sketch) This can be proved by following the proof of Theorem 3.3 by using Banach's fixed-point theorem. A detailed proof can be found in Appendix A.1. □

The above lemma shows that the approximated distance on the embedding space $\hat{d}(\phi_\omega(\mathbf{s}_i), \phi_\omega(\mathbf{s}_j))$ still converges to the Shift MICo distance, $\tilde{d}^\pi_M$, by minimizing (7). However, as $\mathbb{E}_{\substack{\mathbf{a}_i\sim\pi\\ \mathbf{a}_j\sim\pi}} |r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}| \geq |\,\mathbb{E}_{\mathbf{a}_i\sim\pi}\, r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j\sim\pi}\, r^{\mathbf{a}_j}_{\mathbf{s}_j}|$, we have the following theorem, whose proof can be found in Appendix A.2.

Theorem 4.3 (Looser value difference bound)**.** *Given states $\mathbf{s}_i$ and $\mathbf{s}_j$, and a policy π, we have*

$$|V^\pi(\mathbf{s}_i) - V^\pi(\mathbf{s}_j)| \leq d^\pi_M(\mathbf{s}_i, \mathbf{s}_j) \leq \tilde{d}^\pi_M(\mathbf{s}_i, \mathbf{s}_j). \tag{8}$$

Based on the above theorem, as the Shift MICo distance has looser value difference bound, it may be less relevant to the value function. As a result, the learned representation may not be able to encode the behavioral similarity between states accurately. Apart from MICo, other behavioral metric-based methods, which consist of the on-policy reward difference term, such as DBC [29, 16] and AMBS [5], also suffer from a similar approximation gap.

4.2 Relaxation of Dynamics Models Divergence

The π-bisimulation metric needs to compute the 1-Wasserstein distance W_1 between dynamics models to measure the distribution distance. In π-bisimulation metric-based representation learning, one can learn the encoder ϕ_ω by minimizing the following loss,

$$\mathcal{L}(\phi_\omega) = \mathbb{E}\left[\left(\hat{d}(\phi_\omega(\mathbf{s}_i), \phi_\omega(\mathbf{s}_j)) - \left|\mathbb{E}_{\mathbf{a}_i\sim\pi} r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j\sim\pi} r^{\mathbf{a}_j}_{\mathbf{s}_j}\right| - \gamma W_1(\hat{d})(P^\pi_{\phi_\omega(\mathbf{s}_i)}, P^\pi_{\phi_\omega(\mathbf{s}_j)})\right)^2\right],$$

However, the 1-Wasserstein distance is computationally expensive or intractable. In DBC [29] the 2-Wasserstein distance W_2 is proposed to replace W_1, as W_2 has a convenient closed-form of a Gaussian distribution with respect to the L_2 distance. Specifically, the loss function of DBC with batched sampled transitions is defined as,

$$\mathcal{L}(\phi_\omega) = \mathbb{E}\left[\left(\hat{d}(\phi_\omega(\mathbf{s}_i), \phi_\omega(\mathbf{s}_j)) - \left|r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}\right| - \gamma W_2(\|\cdot\|_2)(\widehat{P}^{\bar{\pi}}_{\phi_\omega(\mathbf{s}_i)}, \widehat{P}^{\bar{\pi}}_{\phi_\omega(\mathbf{s}_j)})\right)^2\right],$$

where $\|\cdot\|_2$ is the L_2 norm, $\widehat{P}$ is a dynamics model on the representation space, and $\bar{\pi}$ is the expected policy output. The use of W_2 almost breaks all theoretical guarantees for the bisimulation metric. The existence of unique fixed-point in the bisimulation metric requires the continuity and monotonicity of W_1 with respect to d [7]. The properties of continuity and monotonicity do not hold with W_2. Therefore there is no more guarantee about the fixed-point existence in DBC except that both the dynamics model and the policy π are deterministic, in which case $W_2(d)$ degenerates to d and Banach's fixed-point exists [16]. However, this assumption may be too strong to hold in practice.

Instead of using the family of Wasserstein distances, in MICo as shown in (3) the sample-based distribution divergence, $\mathbb{E}_{\mathbf{s}'_i\sim P^\pi_{\mathbf{s}_i}, \mathbf{s}'_j\sim P^\pi_{\mathbf{s}_j}} d(\mathbf{s}'_i, \mathbf{s}'_j)$, is used to measure the difference between dynamics models. This sample-based distribution divergence can be considered as a Łukaszyk-Karmowski metric. While a Wasserstein distance has zero self-distance, a Łukaszyk-Karmowski metric does not satisfy the identity of indiscernibles. As a result, the approximated distance on the learned embedding space based on the MICo distance, which involves a Łukaszyk-Karmowski metric to measure distance between dynamics models, may also suffer from the violation issue of the identity of indiscernibles.

4.3 Limitation of Using the L_1 or the L_2 Norm on the Embedding Space

As mentioned in Section 4.1, to avoid the expensive or intractable computation of the expectation over rewards, prior approaches, such as MICo, and DBC, use an alternative term to measure the reward difference between states, $\mathbb{E}_{\substack{\mathbf{a}_i\sim\pi\\ \mathbf{a}_j\sim\pi}}|r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}|$. In the following, we discuss the case that state $\mathbf{s}_i$ and $\mathbf{s}_j$ are identical.

Lemma 4.4. *If $\mathbf{s}_i = \mathbf{s}_j$, then $\mathbb{E}_{\substack{\mathbf{a}_i\sim\pi\\ \mathbf{a}_j\sim\pi}}|r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}| \geq 0$. The equality holds only if the reward function r is constant w.r.t. action a or the policy π is deterministic.*

Note that in most RL tasks, a stochastic policy is widely used for exploration and a reward function is rarely constant w.r.t. actions. Therefore, in practice, $\mathbb{E}_{\substack{\mathbf{a}_i\sim\pi\\ \mathbf{a}_j\sim\pi}}|r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}| > 0$ for most RL tasks. Besides, as mentioned in Section 4.2, a Łukaszyk-Karmowski metric measuring distance between dynamics models does not satisfy the identity of indiscernibles. Therefore, the behavioral metric, which is a sum of the reward difference term and the dynamics model distance term (no matter a Wasserstein distance or a Łukaszyk-Karmowski metric), can be greater than zero on a pair of identical states.

Note that both the L_1 and the L_2 norms satisfy the identity of indiscernibles, i.e. when $\mathbf{x}_i = \mathbf{x}_j$, $\|\mathbf{x}_i - \mathbf{x}_j\|_1 = \|\mathbf{x}_i - \mathbf{x}_j\|_2 = 0$. If we leverage the L_1 or the L_2 norm as the form of distance $\hat{d}$ on the embedding space to approximate the behavioral metric, then the identical states pairs has zero distance on embedding space. However, the regression target, i.e., the behavioral metric, is greater than zero on identical states. As a result, when minimizing regression loss between the L_1 / L_2 norm and the behavioral metric, the representations for similar states, especially identical ones, will be pushed apart in the embedding space.

Recently, AMBS [5] proposes to use neural networks to measure distance on state embedding space rather than using the L_1 or the L_2 norm. However, in this case, the learned "behavior metric" does not have theoretical support such as the fixed-point convergence guarantee. Besides, MICo proposes a new form of distance on the embedding space, which is a sum of angular distance between embeddings and the L_2 norm of the embeddings. Note that the proposed distance has non-zero self-distance.

5 The Proposed RAP Distance

We firstly propose a new behavioral metric to measure the state similarity without computing the Wasserstein distance between dynamics models in Section 5.1. The proposed behavioral metric namely the RAP distance enjoys theoretical properties such as fixed-point existence and a value difference bound. We then present a practical algorithm to learn the state encoder by approximating the RAP distance on the state embedding space in Section 5.2. Our algorithm uses the learned estimation of reward functions and dynamics models to provide distance approximation which is consistent to the behavioral metric. It addresses all the aforementioned approximation gaps and preserves the theoretical guarantee about the value function difference bound. Particularly, the approximation gap of loss function mismatch is addressed in Section 5.2, the relaxation of dynamics model divergence is addressed in Section 5.1, and the limitation of the L_1 or the L_2 norm on the embedding space is addressed in Section 5.3.

5.1 Definition and Properties of the RAP Distance

In order to avoid the high computational cost of W_1 or the approximation gap introduced by relaxation to W_2 as described in Section 4.2, we consider a distance measure between dynamics models $P^\pi_{\mathbf{s}_i}$ and $P^\pi_{\mathbf{s}_j}$ with sampling. To be specific, our on-policy behavioral distance is defined as follows.

Definition 5.1 (the RAP distance)**.** Let $\mathbb{M}$ be the space of distance function $d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$, the RAP distance function $\mathcal{F}^\pi_G : \mathbb{M} \to \mathbb{M}$ is defined as,

$$\mathcal{F}^\pi_G(d)(\mathbf{s}_i, \mathbf{s}_j) = \left|\mathbb{E}_{\mathbf{a}_i \sim \pi}\, r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j \sim \pi}\, r^{\mathbf{a}_j}_{\mathbf{s}_j}\right| + \gamma \mathbb{E}_{\substack{\mathbf{a}_i \sim \pi \\ \mathbf{a}_j \sim \pi}} d(\mathbb{E}[s'_i], \mathbb{E}[s'_j]), \tag{9}$$

where $\mathbb{E}[s'_i] = \mathbb{E}_{s'_i \sim P^{\mathbf{a}_i}_{\mathbf{s}_i}}[s'_i]$ is the expectation value of next state over the dynamics model $P^{\mathbf{a}_i}_{\mathbf{s}_i}$.

We design the behavioral distance by measuring the expected states over dynamics models recursively, which removes the requirement of sampling on $P^{\mathbf{a}_i}_{\mathbf{s}_i}$ and $P^{\mathbf{a}_j}_{\mathbf{s}_j}$ but only performs sampling on the policy π. In practice, the expected next states are generated by a learnable approximated dynamics model as described in Section 5.2.

Theorem 5.2. *$\mathcal{F}^\pi_G$ is a contraction mapping w.r.t. the L_∞ norm and has a unique fixed-point d^π_G.*

Proof. Let $d, d' \in \mathbb{M}$. We have

$$|\mathcal{F}^\pi_G(d)(\mathbf{s}_i, \mathbf{s}_j) - \mathcal{F}^\pi_G(d')(\mathbf{s}_i, \mathbf{s}_j)| = \left|\gamma \sum_{\mathbf{a}_i, \mathbf{a}_j} \pi(\mathbf{a}_i|\mathbf{s}_i)\pi(\mathbf{a}_j|\mathbf{s}_j)(d - d')(\mathbb{E}[s'_i], \mathbb{E}[s'_j])\right| \leq \gamma \|d - d'\|_\infty.$$

Therefore, $\mathcal{F}^\pi_G$ is a contraction mapping w.r.t. the L_∞ norm and there exists a unique fixed-point for $\mathcal{F}^\pi_G$ due to Banach's fixed-point theorem. This completes the proof. □

Theorem 5.2 provides a convergence guarantee for the RAP distance that by iterating $\mathcal{F}^\pi_G$, distance d will converge to the fixed-point d^π_G.

Theorem 5.3 (Value function difference bound)**.** *Given states* $\mathbf{s}_i$ *and* $\mathbf{s}_j$*, and a policy* π*, we have*

$$|V^\pi(\mathbf{s}_i) - V^\pi(\mathbf{s}_j)| \leq d^\pi_G(\mathbf{s}_i, \mathbf{s}_j). \tag{10}$$

The proof can be found in Appendix A.3. Theorem 5.3 demonstrates that the RAP distance between states upper-bounds the difference of their states values.

5.2 Approximation of RAP

A straightforward way to learn a representation encoder to approximate the RAP distance on the embedding space is to minimize the loss in (5). However, the approximation gap of loss function mismatch as mentioned in Section 4.1 still exists if we relax the first term in (9) as $\mathbb{E}_{\substack{\mathbf{a}_i\sim\pi\\ \mathbf{a}_j\sim\pi}}|r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}|$. This leads to the learned metric having a looser value difference bound than the original behavioral metric as proved in Section 4.1. Here, we propose another alternative relaxation of $|\,\mathbb{E}_{\mathbf{a}_i\sim\pi} r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j\sim\pi} r^{\mathbf{a}_j}_{\mathbf{s}_j}|$ to address this issue. Let $r_\mathbf{s}$ be a random variable over the action distribution defined by $p(r_\mathbf{s} = r^\mathbf{a}_\mathbf{s}) = \pi(\mathbf{a}|\mathbf{s})$. We first analyze the difference between $\mathbb{E}_{\substack{\mathbf{a}_i\sim\pi\\ \mathbf{a}_j\sim\pi}}\left[\left|r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}\right|^2\right]$ and $|\,\mathbb{E}_{\mathbf{a}_i\sim\pi} r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j\sim\pi} r^{\mathbf{a}_j}_{\mathbf{s}_j}|^2$,

$$\begin{aligned}
&\mathbb{E}_{\substack{\mathbf{a}_i\sim\pi\\ \mathbf{a}_j\sim\pi}}\left[\left|r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}\right|^2\right] - |\,\mathbb{E}_{\mathbf{a}_i\sim\pi} r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j\sim\pi} r^{\mathbf{a}_j}_{\mathbf{s}_j}|^2 \\
&= \mathbb{E}_{\mathbf{a}_i\sim\pi}\left[(r^{\mathbf{a}_i}_{\mathbf{s}_i})^2\right] + \mathbb{E}_{\mathbf{a}_j\sim\pi}\left[(r^{\mathbf{a}_j}_{\mathbf{s}_j})^2\right] - 2\mathbb{E}_{\substack{\mathbf{a}_i\sim\pi\\ \mathbf{a}_j\sim\pi}}\left[r^{\mathbf{a}_i}_{\mathbf{s}_i} r^{\mathbf{a}_j}_{\mathbf{s}_j}\right] - \left[\mathbb{E}_{\mathbf{a}_i\sim\pi} r^{\mathbf{a}_i}_{\mathbf{s}_i}\right]^2 - \left[\mathbb{E}_{\mathbf{a}_j\sim\pi} r^{\mathbf{a}_j}_{\mathbf{s}_j}\right]^2 + 2\left[\mathbb{E}_{\mathbf{a}_i\sim\pi} r^{\mathbf{a}_i}_{\mathbf{s}_i}\right]\left[\mathbb{E}_{\mathbf{a}_j\sim\pi} r^{\mathbf{a}_j}_{\mathbf{s}_j}\right] \\
&= \mathbb{E}_{\mathbf{a}_i\sim\pi}\left[(r^{\mathbf{a}_i}_{\mathbf{s}_i})^2\right] - \left[\mathbb{E}_{\mathbf{a}_i\sim\pi} r^{\mathbf{a}_i}_{\mathbf{s}_i}\right]^2 + \mathbb{E}_{\mathbf{a}_j\sim\pi}\left[(r^{\mathbf{a}_j}_{\mathbf{s}_j})^2\right] - \left[\mathbb{E}_{\mathbf{a}_j\sim\pi} r^{\mathbf{a}_j}_{\mathbf{s}_j}\right]^2 - 2\mathbb{E}_{\substack{\mathbf{a}_i\sim\pi\\ \mathbf{a}_j\sim\pi}}\left[r^{\mathbf{a}_i}_{\mathbf{s}_i} r^{\mathbf{a}_j}_{\mathbf{s}_j}\right] + 2\left[\mathbb{E}_{\mathbf{a}_i\sim\pi} r^{\mathbf{a}_i}_{\mathbf{s}_i}\right]\left[\mathbb{E}_{\mathbf{a}_j\sim\pi} r^{\mathbf{a}_j}_{\mathbf{s}_j}\right] \\
&= Var[r_{\mathbf{s}_i}] + Var[r_{\mathbf{s}_j}] - 2Cov[r_{\mathbf{s}_i}, r_{\mathbf{s}_j}].
\end{aligned} \tag{11}$$

Since $r_{\mathbf{s}_i}$ and $r_{\mathbf{s}_j}$ are independent, $Cov[r_{\mathbf{s}_i}, r_{\mathbf{s}_j}] = 0$. Therefore, we have the reward difference term

$$|\,\mathbb{E}_{\mathbf{a}_i\sim\pi} r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j\sim\pi} r^{\mathbf{a}_j}_{\mathbf{s}_j}| = \sqrt{\mathbb{E}_{\substack{\mathbf{a}_i\sim\pi\\ \mathbf{a}_j\sim\pi}}\left[\left|r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}\right|^2\right] - Var[r_{\mathbf{s}_i}] - Var[r_{\mathbf{s}_j}]}$$

and the revised RAP distance at fixed-point as

$$\begin{aligned}
d^\pi_G(\mathbf{s}_i, \mathbf{s}_j) = &\sqrt{\mathbb{E}_{\substack{\mathbf{a}_i\sim\pi\\ \mathbf{a}_j\sim\pi}}\left[\left|r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}\right|^2\right] - Var[r_{\mathbf{s}_i}] - Var[r_{\mathbf{s}_j}]} \\
&+ \gamma\mathbb{E}_{\substack{\mathbf{a}_i\sim\pi\\ \mathbf{a}_j\sim\pi}} d^\pi_G(\mathbb{E}_{s'_i\sim P^{\mathbf{a}_i}_{\mathbf{s}_i}}[s'_i], \mathbb{E}_{s'_j\sim P^{\mathbf{a}_j}_{\mathbf{s}_j}}[s'_j]).
\end{aligned} \tag{12}$$

In (12), we successfully move the expectation operator on rewards out of the absolute value operator without introducing any approximation gap caused by loss function mismatch. However, there are three issues that need to be further addressed: 1) the square root introduces new bias under sampling, 2) the variances $Var[r_{\mathbf{s}_i}]$ and $Var[r_{\mathbf{s}_j}]$ are intractable to compute, and 3) how to estimate the expected next states $\mathbb{E}_{s'_i\sim P^{\mathbf{a}_i}_{\mathbf{s}_i}}[s'_i]$ and $\mathbb{E}_{s'_j\sim P^{\mathbf{a}_j}_{\mathbf{s}_j}}[s'_j]$.

In order to reduce the bias issue introduced by the square root, we try to remove the square root in the loss of learning the RAP distance. We move the dynamics term in (12) to the left-hand side, then take square on both sides and get

$$\left(d^\pi_G(\mathbf{s}_i, \mathbf{s}_j) - \gamma\mathbb{E}_{\substack{\mathbf{a}_i\sim\pi\\ \mathbf{a}_j\sim\pi}} d^\pi_G(\mathbb{E}_{s'_i\sim P^{\mathbf{a}_i}_{\mathbf{s}_i}}[s'_i], \mathbb{E}_{s'_j\sim P^{\mathbf{a}_j}_{\mathbf{s}_j}}[s'_j])\right)^2 = \mathbb{E}_{\substack{\mathbf{a}_i\sim\pi\\ \mathbf{a}_j\sim\pi}}\left[\left|r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}\right|^2\right] - Var[r_{\mathbf{s}_i}] - Var[r_{\mathbf{s}_j}]. \tag{13}$$

Let $\hat{d}(\phi_\omega(\mathbf{s}_i), \phi_\omega(\mathbf{s}_j))$ be the approximated RAP distance between $\mathbf{s}_i$ and $\mathbf{s}_j$ parameterized by ω. The loss for learning $\hat{d}(\phi_\omega(\mathbf{s}_i), \phi_\omega(\mathbf{s}_j))$ is to minimize the mean squared error between the left-hand side and the right-hand side in (13):

$$\begin{aligned}
\mathcal{L} = \mathbb{E}\Bigg[&\left(\hat{d}(\phi_\omega(\mathbf{s}_i), \phi_\omega(\mathbf{s}_j)) - \gamma\mathbb{E}_{\substack{\mathbf{a}_i\sim\pi\\ \mathbf{a}_j\sim\pi}} d^\pi_G(\mathbb{E}_{s'_i\sim P^{\mathbf{a}_i}_{\mathbf{s}_i}}[s'_i], \mathbb{E}_{s'_j\sim P^{\mathbf{a}_j}_{\mathbf{s}_j}}[s'_j])\right)^2 \\
&- \left(|r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}|^2 - Var[r_{\mathbf{s}_i}] - Var[r_{\mathbf{s}_j}]\right)\Bigg]^2.
\end{aligned} \tag{14}$$

Such a loss consists of a pair of squared difference terms as the regression target, which approximate the squared difference between distances of current states and distances of dynamics models, respectively.

The reward variance $Var[r_{\mathbf{s}}]$ is computationally intractable, but we can learn a neural network approximator to estimate it by assuming that the reward $r_{\mathbf{s}}$ on state $\mathbf{s}$ is Gaussian distributed. Let r_ψ be the learned reward function approximation parameterized by ψ, which outputs a Gaussian distribution, $r_\psi(\mathbf{s}) = \{\widehat{\mu}(r_{\mathbf{s}}), \widehat{\sigma}(r_{\mathbf{s}})\}$, where $\widehat{\mu}(r_{\mathbf{s}})$ and $\widehat{\sigma}(r_{\mathbf{s}})$ are the mean and the standard deviation, respectively.

Similarly, in order to estimate the expected next states $\mathbb{E}_{s' \sim P^{\mathbf{a}}_{\mathbf{s}}}[s']$ with a neural network approximator on the embedding space, we learn a dynamics model $\widehat{P}$ taking input of state embedding $\phi(\mathbf{s})$ and action $\mathbf{a}$ and outputs a Gaussian distribution over the next state embedding, $\widehat{P}(\phi_\omega(\mathbf{s}), \mathbf{a}) = \{\widehat{\mu}(\widehat{P}^{\mathbf{a}}_{\phi_\omega(\mathbf{s})}), \widehat{\sigma}(\widehat{P}^{\mathbf{a}}_{\phi_\omega(\mathbf{s})})\}$, where $\widehat{\mu}(\widehat{P}^{\mathbf{a}}_{\phi_\omega(\mathbf{s})})$ and $\widehat{\sigma}(\widehat{P}^{\mathbf{a}}_{\phi_\omega(\mathbf{s})})$ are the mean vector and the standard deviation vector for predictive embeddings of the next state, respectively.

The RAP distance between expected next states, $d^\pi_G(\mathbb{E}[s'_i], \mathbb{E}[s'_j])$, can be approximated by $\hat{d}(\widehat{\mu}(\widehat{P}^{\mathbf{a}_i}_{\phi_\omega(\mathbf{s}_i)}), \widehat{\mu}(\widehat{P}^{\mathbf{a}_j}_{\phi_\omega(\mathbf{s}_j)}))$. The variance $Var[r_{\mathbf{s}}]$ can be replaced by the learned reward function output as $(\widehat{\sigma}(r_{\mathbf{s}}))^2$. By replacing the dynamics term and reward variance terms in (14), we propose the RAP loss defined as

$$\mathcal{L}_{RAP}(\phi_\omega) = \mathbb{E}_{\mathcal{D}}\Bigg[\Big(\hat{d}(\phi_\omega(\mathbf{s}_i), \phi_\omega(\mathbf{s}_j)) - \gamma \hat{d}(\widehat{\mu}(\widehat{P}^{\mathbf{a}_i}_{\phi_{\bar{\omega}}(\mathbf{s}_i)}), \widehat{\mu}(\widehat{P}^{\mathbf{a}_j}_{\phi_{\bar{\omega}}(\mathbf{s}_j)}))\Big)^2 - \Big(|r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}|^2 - (\widehat{\sigma}(r_{\mathbf{s}_i}))^2 - \big(\widehat{\sigma}(r_{\mathbf{s}_j})\big)^2\Big) \Bigg]^2, \quad (15)$$

where $\bar{\omega}$ is a copy of parameter with stop gradient and $\phi_{\bar{\omega}}$ is the encoder in the target network and $\mathcal{D}$ is the replay buffer or the set of data that RL algorithm, e.g. SAC [10], learns from. The loss (15) is trained over the transitions sampled from $\mathcal{D}$.

5.3 Explicit Form of Distance on the Embedding Space

As discussed in Section 4.3, the behavioral metric is usually with non-zero self-distance. Besides, the distance measured between the next-state distributions in the RAP distance is a Łukaszyk-Karmowski metric [22], which also has non-zero self-distance. Here, we adopt the approximation form of distance on the embedding space as proposed in MICo [4].

Definition 5.4 (MICo approximation)**.** Let $\hat{d}_G : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a distance function on representation space $\mathbb{R}^n$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $k > 0$. $\hat{d}_G$ is defined as $\hat{d}_G(\mathbf{x}, \mathbf{y}) = \|\mathbf{x}\|_2^2 + \|\mathbf{y}\|_2^2 + k\theta(\mathbf{x}, \mathbf{y})$, where θ is angular function evaluating the absolute angle between vectors $\mathbf{x}$ and $\mathbf{y}$, and k is a hyperparameter. In practice, we set $k = 10^{-5}$ for our method.

As the above form of distance produces non-zero self-distance, with $\hat{d}_G$ the approximation of the RAP distance will not push apart similar states on the embedding space.

Lemma 5.5 (Non-zero self-distance)**.** *The self-distance of $\hat{d}_G$ is not restrict to zero: $\hat{d}_G(\mathbf{x}, \mathbf{x}) = 2\|\mathbf{x}\|_2^2 \geq 0$. The equality holds only if all the elements of $\mathbf{x}$ are zero.*

5.4 Implementation

The network architecture of our method is shown in Figure 1. Our method is built upon SAC [10]. Actor and critic networks take input of state representation $\phi_\omega(\mathbf{s})$. The SAC objectives and the RAP regression loss $\mathcal{L}_{RAP}(\phi_\omega)$ are optimized jointly. More implementation details can be found in Appendix B. The source code is available at `https://github.com/jianda-chen/RAP_distance`.

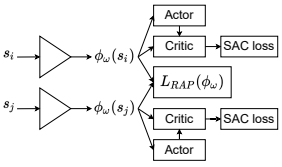
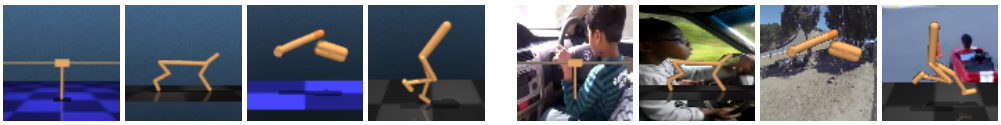


Figure 1: Network architecture of our method.

6 Experiment

In this section, we evaluate the efficiency, robustness and generalization ability of the representation learned by our method on three RL benchmarks: 1) Distracting DeepMind Control Suite (DMC) [24], 2) Robosuite [30], and 3) CARLA [6]. These are all control tasks in continuous action spaces with visual input. We compare our method with several baselines and state-of-the-art methods including metric-based representation learning and data augmentation: 1) **SAC** [10], a baseline RL method for continuous control, 2) **MICo** [4], a sample-based behavioral metric-based representation learning method for RL, 3) **DBC** [29], a representation learning method by approximating the bisimulation metric with the L_1 norm, 4) **RobustDBC** [16], a DBC-styled method with intrinsic rewards and inverse dynamics, and 5) **DrQ** [28], an image augmentation method on pixel inputs RL.



(a) (b)

Figure 2: Illustrations of observations in DMC: **(a)** original background setting; and **(b)** natural video background setting. From **left** to **right**: cartpole-swingup, cheetah-run, finger-spin, and walker-walk.

Distracting DeepMind Control Suite To evaluate the generalization ability and robustness of our method, we perform experiments on 2 settings, original background and natural video background settings, in DMC [24]. We render 84×84 pixels as observation at each time step and stack frames as state. In the **original background** setting, we use the default background provided by DMC (as shown in Figure 2a). For the **natural video background** distracting setting, we follow [29] to replace the background with natural video sampled from Kinetics dataset [15] (as shown in Figure 2b). We sample 1000 continuous frames from the video dataset as background for training RL agents and evaluate agents on another 1000 continuous frames. The video background is considered as distraction to RL algorithm. For each setting, we train and evaluate on 4 tasks: cartpole-swingup, cheetah-run, finger-spin, and walker-walk. Each task is trained using 1 million environment steps.

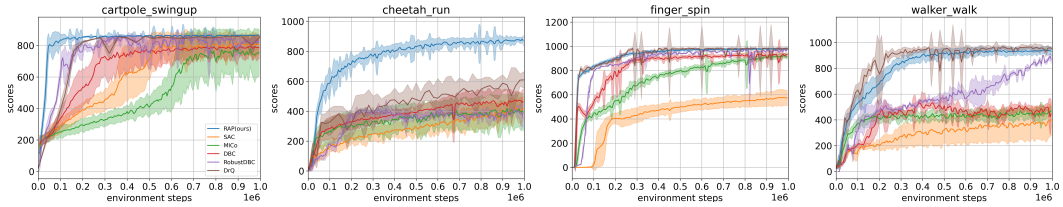


Figure 3: Experimental results on DMC with original background. Each curve is average on 3 runs.

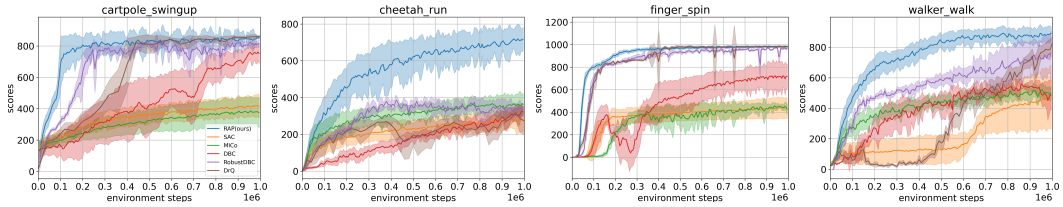


Figure 4: Results on natural video background settings in DMC. Each curve is average on 3 runs.

Figure 3 shows the training curves in original background settings. It confirms that our method RAP can accelerate the training and has comparable results to data augmentation method DrQ. Figure 4 shows the training curves in natural video background settings. Our method RAP outperforms DBC and MICo on all 4 tasks. It also converges to higher score than DrQ in 1 task and learns much faster in 3 tasks. The new SOTA results verify that our method of reducing the gaps in approximation of behavioral distance can improve the efficiency and robustness for deep RL and they also confirm that our method is able to learn generalizable state representation in complex environments.

Robosuite Robosuite [30] is a robotic simulator with various manipulation tasks based on MuJoCo engine. To evaluate the robustness of methods, we use the distraction settings for Robosuite by randomizing the colors of robot arms and table, the lighting source and luminance, and the camera position. The color, lighting and camera position are changed at the beginning of an episode but are consistent within the episode. Our experiment is performed by controlling the Panda arms. The control rate is 20Hz and the episode length is 500 steps. We evaluate on two tasks: Door Opening and Two Arm Peg-In-Hole. Figure 5 shows illustrations of observations of two tasks. We compare our method with behavioral metric methods MICo and DBC, and data augmentation method DrQ. Table 1 shows the training scores in 500k environment steps. Our method RAP outperforms the others. The converge scores are around 7× and 1.4× compared to the second-best method in Door Opening and Two Arm Peg-In-Hole, respectively. It shows the robustness of our RAP in environments with random distractions.

Figure 5: Illustrations in Robosuite. **Left**: Door Opening. **right**: Two Arm Peg-In-Hole.

Table 1: Experimental results on Robosuite trained with 500K environment steps.

Task	RAP(Ours)	SAC	MICo	DBC	DrQ
Door Opening	**102.19 ± 26.11**	8.84±9.89	6.06±6.57	3.15±3.54	14.63±19.57
Two Arm Peg-In-Hole	**307.27±25.70**	191.29±34.88	123.69±23.25	219.56±36.87	156.86±33.57

CARLA In order to validate the generalization ability and learning efficiency in natural scenarios, we construct experiments on an autonomous driving simulator CARLA [6], which provides 3D realistic on-world scenarios. The goal of this task is to control a vehicle driving as far as possible on a high-way map (Town 4) in 1000 time steps. The reward function followed [29] is designed to encourage driving far and avoid collisions with other vehicles. The observation is formed as 420 × 84 pixels of a 300 degrees ego-centric view, constructed by concatenating five cameras. In order to evaluate the generalization ability, we randomly sample a kind of weather in each episode starts. The weather (sunlight, rain, etc.), which affects the visual observation, can be considered as real-world distraction to RL agents. Figure 6 shows the comparison results of our RAP and other SOTA methods. Our method achieves comparable scores with data augmentation method DrQ, and its score is higher than behavioral metric DBC and MICo. It shows that our RAP is able to generalize well in tasks with real-world scenarios.

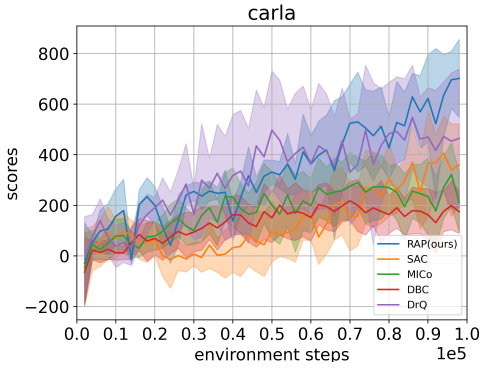


Figure 6: Training curves on CARLA.

7 Conclusion

Representation learning is one of the most critical problems in high-dimensional deep RL. In this paper, we propose a new behavioral metric and a practical representation learning algorithm on top of the new behavioral metric for deep RL. We provide theoretical analysis for our proposed metric as well as the representation learning algorithm. We conduct empirical studies on multiple RL domains to verify the effectiveness of our proposed method.

Acknowledgement and Funding Disclosure

This work is supported by the 2020 Microsoft Research Asia collaborative research grant.

References

[1] Rishabh Agarwal, Marlos C. Machado, Pablo Samuel Castro, and Marc G Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qda7-sVg84.

[2] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 449–458. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/bellemare17a.html.

[3] Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10069–10076, 2020.

[4] Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. MICo: Improved representations via sampling-based state similarity for markov decision processes. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=wFp6kmQELgu.

[5] Jianda Chen and Sinno Pan. Learning generalizable representations for reinforcement learning via adaptive meta-learner of behavioral similarities. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=zBOI9LFpESK.

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[7] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, page 162–169, Arlington, Virginia, USA, 2004. AUAI Press. ISBN 0974903906.

[8] Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous markov decision processes. *SIAM J. Comput.*, 40(6):1662–1714, December 2011. ISSN 0097-5397. doi: 10.1137/10080484X. URL https://doi.org/10.1137/10080484X.

[9] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2170–2179. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/gelada19a.html.

[10] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

[11] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

[12] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=o_V-MjyyGV_.

[13] Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. DARLA: Improving zero-shot transfer in reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine*

Learning Research, pages 1480–1490. PMLR, 06–11 Aug 2017. URL http://proceedings.mlr.press/v70/higgins17a.html.

[14] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks, 2017.

[15] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset, 2017.

[16] Mete Kemertas and Tristan Ty Aumentado-Armstrong. Towards robust bisimulation metric learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=ySFGlFjgIfN.

[17] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5639–5650. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/laskin20a.html.

[18] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19884–19895. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/e615c82aba461681ade82da2da38004a-Paper.pdf.

[19] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJgcvJBFvB.

[20] Kuang-Huei Lee, Ian Fischer, Anthony Liu, Yijie Guo, Honglak Lee, John Canny, and Sergio Guadarrama. Predictive information accelerates learning in rl. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 11890–11901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/89b9e0a6f6d1505fe13dea0f18a2dcfa-Paper.pdf.

[21] Xingyu Lin, Harjatin Baweja, George Kantor, and David Held. Adaptive auxiliary task weighting for reinforcement learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/0e900ad84f63618452210ab8baae0218-Paper.pdf.

[22] Szymon Łukaszyk. A new concept of probability metric and its applications in approximation of scattered data sets. *Computational Mechanics*, 33(4):299–304, 2004.

[23] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9870–9879. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/stooke21a.html.

[24] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite, 2018.

[25] Niklas Wahlström, Thomas B. Schön, and Marc Peter Deisenroth. From pixels to torques: Policy learning with deep dynamical models, 2015.

[26] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/a1afc58c6ca9540d057299ec3016d726-Paper.pdf.

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

[28] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=GY6-6sTvGaf.

[29] Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=-2FCwDKRREu.

[30] Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.

A Proofs

A.1 Proof of Lemma 4.2

Lemma 4.2 (Fixed-point of shift MICo)**.** *The "shift" MICo distance function $\tilde{\mathcal{F}}^\pi_M$ has a unique fixed-point $\tilde{d}^\pi_M$.*

Proof. The shift MICo distance function $\tilde{\mathcal{F}}^\pi_M$ is defined as,

$$\tilde{\mathcal{F}}^\pi_M(d)(\mathbf{s}_i, \mathbf{s}_j) = \mathbb{E}_{\substack{\mathbf{a}_i \sim \pi \\ \mathbf{a}_j \sim \pi}} |r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}| + \gamma \mathbb{E}_{\substack{\mathbf{s}'_i \sim P^\pi_{\mathbf{s}_i} \\ \mathbf{s}'_j \sim P^\pi_{\mathbf{s}_j}}} d(\mathbf{s}'_i, \mathbf{s}'_j).$$

Let $d, d' \in \mathbb{M}$. We have

$$\begin{aligned}
&\left| \tilde{\mathcal{F}}^\pi_M(d)(\mathbf{s}_i, \mathbf{s}_j) - \tilde{\mathcal{F}}^\pi_M(d')(\mathbf{s}_i, \mathbf{s}_j) \right| \\
=&\left| \gamma \sum_{\mathbf{s}'_i, \mathbf{s}'_j} \pi(\mathbf{a}_i|\mathbf{s}_i)\pi(\mathbf{a}_j|\mathbf{s}_j) P^{\mathbf{a}_i}_{\mathbf{s}_i}(\mathbf{s}'_i) P^{\mathbf{a}_j}_{\mathbf{s}_j}(\mathbf{s}'_j)(d - d')(s'_i, s'_j) \right| \\
\leq& \gamma \|d - d'\|_\infty.
\end{aligned}$$

$\tilde{\mathcal{F}}^\pi_M$ is a contraction mapping w.r.t. the L_∞ norm and there exists a unique fixed-point $\tilde{d}^\pi_M$ for $\tilde{\mathcal{F}}^\pi_M$ due to Banach's fixed-point theorem. This completes the proof. □

A.2 Proof of Theorem 4.3

Lemma A.1. $\mathbb{E}_{\substack{\mathbf{a}_i \sim \pi \\ \mathbf{a}_j \sim \pi}} |r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}| \geq |\, \mathbb{E}_{\mathbf{a}_i \sim \pi}\, r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j \sim \pi}\, r^{\mathbf{a}_j}_{\mathbf{s}_j}|\,.$

Proof. Since $r^{\mathbf{a}_i}_{\mathbf{s}_i}$ and $r^{\mathbf{a}_j}_{\mathbf{s}_j}$ are rewards which are scalars, we have $|r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}| \geq r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}$, and $|r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}| \geq r^{\mathbf{a}_j}_{\mathbf{s}_j} - r^{\mathbf{a}_i}_{\mathbf{s}_i}$ by symmetric. By taking the expectation over $\mathbf{a}_i$ and $\mathbf{a}_j$, we get,

$$\mathbb{E}_{\substack{\mathbf{a}_i \sim \pi \\ \mathbf{a}_j \sim \pi}} |r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}| \geq \mathbb{E}_{\substack{\mathbf{a}_i \sim \pi \\ \mathbf{a}_j \sim \pi}} \left[r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j} \right], \tag{16}$$

and

$$\mathbb{E}_{\substack{\mathbf{a}_i \sim \pi \\ \mathbf{a}_j \sim \pi}} |r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}| \geq \mathbb{E}_{\substack{\mathbf{a}_i \sim \pi \\ \mathbf{a}_j \sim \pi}} \left[r^{\mathbf{a}_j}_{\mathbf{s}_j} - r^{\mathbf{a}_i}_{\mathbf{s}_i} \right]. \tag{17}$$

By combining (16) and (17), we have

$$\mathbb{E}_{\substack{\mathbf{a}_i \sim \pi \\ \mathbf{a}_j \sim \pi}} |r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}| \geq \left| \mathbb{E}_{\substack{\mathbf{a}_i \sim \pi \\ \mathbf{a}_j \sim \pi}} \left[r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j} \right] \right| = |\, \mathbb{E}_{\mathbf{a}_i \sim \pi}\, r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j \sim \pi}\, r^{\mathbf{a}_j}_{\mathbf{s}_j}|.$$

□

Lemma A.2 (Lifted MDP of MICo [4])**.** *The MICo metric function $\mathcal{F}^\pi_M$ is the Bellman operator for a Lifted MDP.*

Proof. Define the MDP for RL by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma \rangle$. We consider a lifted MDP constructed by a tuple $\langle \hat{\mathcal{S}}, \hat{\mathcal{A}}, \hat{\mathcal{R}}, \hat{P}, \gamma \rangle$, where state space $\hat{\mathcal{S}} = \mathcal{S} \times \mathcal{S}$, action space $\hat{\mathcal{A}} = \mathcal{A} \times \mathcal{A}$, transition distribution $\hat{P}^{(\mathbf{a}_i, \mathbf{a}_j)}_{(\mathbf{s}_i, \mathbf{s}_j)} = P^{\mathbf{a}_i}_{\mathbf{s}_i} P^{\mathbf{a}_j}_{\mathbf{s}_j}$, and reward function $\hat{\mathcal{R}}((\mathbf{s}_i, \mathbf{s}_j)) = |\, \mathbb{E}_{\mathbf{a}_i \sim \pi}\, r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j \sim \pi}\, r^{\mathbf{a}_j}_{\mathbf{s}_j}|$. The Bellman operator $T^{\hat{\pi}}_M$ under policy $\hat{\pi}(\mathbf{a}_i, \mathbf{a}_j|\mathbf{s}_i, \mathbf{s}_j) = \pi(\mathbf{a}_i|\mathbf{s}_i)\pi(\mathbf{a}_j|\mathbf{s}_j)$ is:

$$\begin{aligned}
T^{\hat{\pi}}_M(d^\pi_M)((\mathbf{s}_i, \mathbf{s}_j)) &= \sum_{(\mathbf{a}_i, \mathbf{a}_j)} \hat{\pi}(\mathbf{a}_i, \mathbf{a}_j|\mathbf{s}_i, \mathbf{s}_j) \sum_{(\mathbf{s}'_i, \mathbf{s}'_j)} \hat{P}^{(\mathbf{a}_i, \mathbf{a}_j)}_{(\mathbf{s}_i, \mathbf{s}_j)}(\mathbf{s}'_i, \mathbf{s}'_j) \left[\hat{\mathcal{R}}((\mathbf{s}_i, \mathbf{s}_j)) + \gamma d^\pi_M(\mathbf{s}'_i, \mathbf{s}'_j) \right] \\
&= |\, \mathbb{E}_{\mathbf{a}_i \sim \pi}\, r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j \sim \pi}\, r^{\mathbf{a}_j}_{\mathbf{s}_j}| + \gamma \mathbb{E}_{\substack{\mathbf{s}'_i \sim P^\pi_{\mathbf{s}_i} \\ \mathbf{s}'_j \sim P^\pi_{\mathbf{s}_j}}} d^\pi_M(\mathbf{s}'_i, \mathbf{s}'_j) \\
&= \mathcal{F}^\pi_M(d^\pi_M)(\mathbf{s}_i, \mathbf{s}_j).
\end{aligned}$$

□

Lemma A.3 (Lifted MDP of shift MICo)**.** *The shift MICo metric function $\tilde{\mathcal{F}}^\pi_M$ is the Bellman operator for a lifted MDP but with a different reward function.*

Proof. We construct the lifted MDP of shift MICo by a tuple $\langle \hat{\mathcal{S}}, \hat{\mathcal{A}}, \tilde{\hat{\mathcal{R}}}, \hat{P}, \gamma \rangle$ which is the same as the lifted MDP in Lemma A.2 except for the reward function $\tilde{\hat{\mathcal{R}}}((\mathbf{s}_i, \mathbf{s}_j), (\mathbf{a}_i, \mathbf{a}_j)) = |r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}|$. The Bellman operator $\tilde{T}^{\hat{\pi}}_M$ under policy $\hat{\pi}$ is:

$$
\begin{aligned}
&\tilde{T}^{\hat{\pi}}_M(\tilde{d}^\pi_M)((\mathbf{s}_i, \mathbf{s}_j)) \\
&= \sum_{(\mathbf{a}_i, \mathbf{a}_j)} \hat{\pi}(\mathbf{a}_i, \mathbf{a}_j | \mathbf{s}_i, \mathbf{s}_j) \sum_{(\mathbf{s}'_i, \mathbf{s}'_j)} \hat{P}^{(\mathbf{a}_i, \mathbf{a}_j)}_{(\mathbf{s}_i, \mathbf{s}_j)}(\mathbf{s}'_i, \mathbf{s}'_j) \left[\tilde{\hat{\mathcal{R}}}((\mathbf{s}_i, \mathbf{s}_j), (\mathbf{a}_i, \mathbf{a}_j)) + \gamma \tilde{d}^\pi_M(\mathbf{s}'_i, \mathbf{s}'_j) \right] \\
&= \mathbb{E}_{\substack{\mathbf{a}_i \sim \pi \\ \mathbf{a}_j \sim \pi}} |r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}| + \gamma \mathbb{E}_{\substack{\mathbf{s}'_i \sim P^\pi_{\mathbf{s}_i} \\ \mathbf{s}'_j \sim P^\pi_{\mathbf{s}_j}}} \tilde{d}^\pi_M(\mathbf{s}'_i, \mathbf{s}'_j) \\
&= \tilde{\mathcal{F}}^\pi_M(\tilde{d}^\pi_M)(\mathbf{s}_i, \mathbf{s}_j).
\end{aligned}
$$

□

Lemma A.4. *Consider an auxiliary MDP specified by a tuple $\langle \hat{\mathcal{S}}, \hat{\mathcal{A}}, \hat{\mathcal{R}}_\Delta, \hat{P}, \gamma \rangle$ where $\hat{\mathcal{R}}_\Delta((\mathbf{s}_i, \mathbf{s}_j), (\mathbf{a}_i, \mathbf{a}_j)) = |r^{\mathbf{a}_i}_{\mathbf{s}_i} - r^{\mathbf{a}_j}_{\mathbf{s}_j}| - |\,\mathbb{E}_{\mathbf{a}_i \sim \pi} r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j \sim \pi} r^{\mathbf{a}_j}_{\mathbf{s}_j}|$. Let $V^{\hat{\pi}}_\Delta$ denote the value function of such MDP over policy $\hat{\pi}$, then*

$$V^{\hat{\pi}}_\Delta((\mathbf{s}_i, \mathbf{s}_j)) \geq 0.$$

Proof. The value function $V^{\hat{\pi}}_\Delta$ can be expanded as,

$$
\begin{aligned}
&V^{\hat{\pi}}_\Delta((\mathbf{s}_i, \mathbf{s}_j)) \\
&= \mathbb{E}_{\hat{\pi}}[\sum_t \gamma^t \hat{\mathcal{R}}_\Delta((\mathbf{s}^{(t)}_i, \mathbf{s}^{(t)}_j), (\mathbf{a}^{(t)}_i, \mathbf{a}^{(t)}_j)) | \mathbf{s}^{(0)}_i = \mathbf{s}_i, \mathbf{s}^{(0)}_j = \mathbf{s}_j] \\
&= \mathbb{E}_{\hat{\pi}} \left[\sum_t \gamma^t \left(\mathbb{E}_{\substack{\mathbf{a}^{(t)}_i \sim \pi \\ \mathbf{a}^{(t)}_j \sim \pi}} \left| r^{\mathbf{a}^{(t)}_i}_{\mathbf{s}^{(t)}_i} - r^{\mathbf{a}^{(t)}_j}_{\mathbf{s}^{(t)}_j} \right| - \left| \mathbb{E}_{\mathbf{a}^{(t)}_i \sim \pi} r^{\mathbf{a}^{(t)}_i}_{\mathbf{s}^{(t)}_i} - \mathbb{E}_{\mathbf{a}^{(t)}_j \sim \pi} r^{\mathbf{a}^{(t)}_j}_{\mathbf{s}^{(t)}_j} \right| \right) \middle| \mathbf{s}^{(0)}_i = \mathbf{s}_i, \mathbf{s}^{(0)}_j = \mathbf{s}_j \right].
\end{aligned}
$$

Due to Lemma A.1, $\mathbb{E}_{\substack{\mathbf{a}^{(t)}_i \sim \pi \\ \mathbf{a}^{(t)}_j \sim \pi}} \left| r^{\mathbf{a}^{(t)}_i}_{\mathbf{s}^{(t)}_i} - r^{\mathbf{a}^{(t)}_j}_{\mathbf{s}^{(t)}_j} \right| - \left| \mathbb{E}_{\mathbf{a}^{(t)}_i \sim \pi} r^{\mathbf{a}^{(t)}_i}_{\mathbf{s}^{(t)}_i} - \mathbb{E}_{\mathbf{a}^{(t)}_j \sim \pi} r^{\mathbf{a}^{(t)}_j}_{\mathbf{s}^{(t)}_j} \right| \geq 0$, therefore,

$$V^{\hat{\pi}}_\Delta((\mathbf{s}_i, \mathbf{s}_j)) \geq 0.$$

□

Theorem 4.3 (Looser value difference bound)**.** *Given states $\mathbf{s}_i$ and $\mathbf{s}_j$, and a policy π, we have*

$$|V^\pi(\mathbf{s}_i) - V^\pi(\mathbf{s}_j)| \leq d^\pi_M(\mathbf{s}_i, \mathbf{s}_j) \leq \tilde{d}^\pi_M(\mathbf{s}_i, \mathbf{s}_j).$$

Proof. Due to Theorem 3.4, which has been proven by Castro et al. [4] in Proposition 4.8 of their paper, we already have the first inequality,

$$|V^\pi(\mathbf{s}_i) - V^\pi(\mathbf{s}_j)| \leq d^\pi_M(\mathbf{s}_i, \mathbf{s}_j). \tag{18}$$

Lemma A.2 shows that d^π_M is the value function of lifted MDP $\langle \hat{\mathcal{S}}, \hat{\mathcal{A}}, \hat{\mathcal{R}}, \hat{P}, \gamma \rangle$. d^π_M can be expanded as the sum of discounted future rewards,

$$d^\pi_M(\mathbf{s}_i, \mathbf{s}_j) = \mathbb{E}_{\hat{\pi}} \left[\sum_t \gamma^t \left(\left| \mathbb{E}_{\mathbf{a}^{(t)}_i \sim \pi} r^{\mathbf{a}^{(t)}_i}_{\mathbf{s}^{(t)}_i} - \mathbb{E}_{\mathbf{a}^{(t)}_j \sim \pi} r^{\mathbf{a}^{(t)}_j}_{\mathbf{s}^{(t)}_j} \right| \right) \middle| \mathbf{s}^{(0)}_i = \mathbf{s}_i, \mathbf{s}^{(0)}_j = \mathbf{s}_j \right].$$

Similarly, due to Lemma A.3, $\tilde{d}^\pi_M$ can be expanded as,

$$\tilde{d}^\pi_M(\mathbf{s}_i, \mathbf{s}_j) = \mathbb{E}_{\hat{\pi}} \left[\sum_t \gamma^t \left(\mathbb{E}_{\substack{\mathbf{a}^{(t)}_i \sim \pi \\ \mathbf{a}^{(t)}_j \sim \pi}} \left| r^{\mathbf{a}^{(t)}_i}_{\mathbf{s}^{(t)}_i} - r^{\mathbf{a}^{(t)}_j}_{\mathbf{s}^{(t)}_j} \right| \right) \middle| \mathbf{s}^{(0)}_i = \mathbf{s}_i, \mathbf{s}^{(0)}_j = \mathbf{s}_j \right].$$

Then we analyze the difference between $\tilde{d}_M^\pi$ and d_M^π. Since $\tilde{d}_M^\pi$ and d_M^π are defined on the same policy $\hat{\pi}$ and dynamics model $\hat{P}$, we have

$$\begin{aligned}
&\tilde{d}_M^\pi(\mathbf{s}_i, \mathbf{s}_j) - d_M^\pi(\mathbf{s}_i, \mathbf{s}_j) \\
&= \mathbb{E}_{\hat{\pi}} \left[\sum_t \gamma^t \left(\mathbb{E}_{\substack{\mathbf{a}_i^{(t)} \sim \pi \\ \mathbf{a}_j^{(t)} \sim \pi}} \left| r^{\mathbf{a}_i^{(t)}}_{\mathbf{s}_i^{(t)}} - r^{\mathbf{a}_j^{(t)}}_{\mathbf{s}_j^{(t)}} \right| - \left| \mathbb{E}_{\mathbf{a}_i^{(t)} \sim \pi} r^{\mathbf{a}_i^{(t)}}_{\mathbf{s}_i^{(t)}} - \mathbb{E}_{\mathbf{a}_j^{(t)} \sim \pi} r^{\mathbf{a}_j^{(t)}}_{\mathbf{s}_j^{(t)}} \right| \right) \middle| \mathbf{s}_i^{(0)} = \mathbf{s}_i, \mathbf{s}_j^{(0)} = \mathbf{s}_j \right] \\
&= V_\Delta^{\hat{\pi}}((\mathbf{s}_i, \mathbf{s}_j)).
\end{aligned}$$

where $V_\Delta^{\hat{\pi}}((\mathbf{s}_i, \mathbf{s}_j))$ is the value function in Lemma A.4 and $V_\Delta^{\hat{\pi}}((\mathbf{s}_i, \mathbf{s}_j)) \geq 0$. Therefore,

$$d_M^\pi(\mathbf{s}_i, \mathbf{s}_j) \leq \tilde{d}_M^\pi(\mathbf{s}_i, \mathbf{s}_j). \tag{19}$$

By combining (18) and (19), we finally prove that,

$$|V^\pi(\mathbf{s}_i) - V^\pi(\mathbf{s}_j)| \leq d_M^\pi(\mathbf{s}_i, \mathbf{s}_j) \leq \tilde{d}_M^\pi(\mathbf{s}_i, \mathbf{s}_j).$$

□

A.3 Proof of Theorem 5.3

Theorem 5.3 (Value function difference bound)**.** *Given states $\mathbf{s}_i$ and $\mathbf{s}_j$, and a policy π, we have*

$$|V^\pi(\mathbf{s}_i) - V^\pi(\mathbf{s}_j)| \leq d_G^\pi(\mathbf{s}_i, \mathbf{s}_j). \tag{20}$$

Proof. We follow Castro et al. [4] (the proof of Proposition 4.8) to prove the value function difference bound. We will show that if $|V^\pi(\mathbf{s}_i) - V^\pi(\mathbf{s}_j)| \leq d(\mathbf{s}_i, \mathbf{s}_j), \forall \mathbf{s}_i, \mathbf{s}_j \in \mathcal{S}$, then $|V^\pi(\mathbf{s}_i) - V^\pi(\mathbf{s}_j)| \leq \mathcal{F}_G^\pi(d)(\mathbf{s}_i, \mathbf{s}_j)$. Suppose $|V^\pi(\mathbf{s}_i) - V^\pi(\mathbf{s}_j)| \leq d(\mathbf{s}_i, \mathbf{s}_j)$ holds, then

$$\begin{aligned}
&V^\pi(\mathbf{s}_i) - V^\pi(\mathbf{s}_j) \\
&= \mathbb{E}_{\mathbf{a}_i \sim \pi} r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j \sim \pi} r^{\mathbf{a}_j}_{\mathbf{s}_j} + \sum_{\mathbf{a}_i} \pi(\mathbf{a}_i|\mathbf{s}_i) \sum_{\mathbf{s}_i'} P^{\mathbf{a}_i}_{\mathbf{s}_i}(\mathbf{s}_i') V^\pi(\mathbf{s}_i') - \sum_{\mathbf{a}_j} \pi(\mathbf{a}_j|\mathbf{s}_j) \sum_{\mathbf{s}_j'} P^{\mathbf{a}_j}_{\mathbf{s}_j}(\mathbf{s}_j') V^\pi(\mathbf{s}_j') \\
&\leq \left| \mathbb{E}_{\mathbf{a}_i \sim \pi} r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j \sim \pi} r^{\mathbf{a}_j}_{\mathbf{s}_j} \right| + \mathbb{E}_{\substack{\mathbf{a}_i \sim \pi \\ \mathbf{a}_j \sim \pi}} \left(\sum_{\mathbf{s}_i'} P^{\mathbf{a}_i}_{\mathbf{s}_i}(\mathbf{s}_i') V^\pi(\mathbf{s}_i') - \sum_{\mathbf{s}_j'} P^{\mathbf{a}_j}_{\mathbf{s}_j}(\mathbf{s}_j') V^\pi(\mathbf{s}_j') \right).
\end{aligned}$$

We make an assumption that $\sum_{\mathbf{s}'} P^{\mathbf{a}}_{\mathbf{s}}(\mathbf{s}') V^\pi(\mathbf{s}') = V^\pi(\mathbb{E}_{s' \sim P^{\mathbf{a}}_{\mathbf{s}}}[s'])$. We make this assumption because we use the learned dynamics model to predict the next states, and the learned dynamics model is assumed as Gaussian distribution with small standard deviation. If this assumption holds, then we have

$$\begin{aligned}
&\left| \mathbb{E}_{\mathbf{a}_i \sim \pi} r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j \sim \pi} r^{\mathbf{a}_j}_{\mathbf{s}_j} \right| + \mathbb{E}_{\substack{\mathbf{a}_i \sim \pi \\ \mathbf{a}_j \sim \pi}} \left(\sum_{\mathbf{s}_i'} P^{\mathbf{a}_i}_{\mathbf{s}_i}(\mathbf{s}_i') V^\pi(\mathbf{s}_i') - \sum_{\mathbf{s}_j'} P^{\mathbf{a}_j}_{\mathbf{s}_j}(\mathbf{s}_j') V^\pi(\mathbf{s}_j') \right) \\
&= \left| \mathbb{E}_{\mathbf{a}_i \sim \pi} r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j \sim \pi} r^{\mathbf{a}_j}_{\mathbf{s}_j} \right| + \mathbb{E}_{\substack{\mathbf{a}_i \sim \pi \\ \mathbf{a}_j \sim \pi}} \left(V^\pi(\mathbb{E}_{s_i' \sim P^{\mathbf{a}_i}_{\mathbf{s}_i}}[s_i']) - V^\pi(\mathbb{E}_{s_j' \sim P^{\mathbf{a}_j}_{\mathbf{s}_j}}[s_j']) \right) \\
&\leq \left| \mathbb{E}_{\mathbf{a}_i \sim \pi} r^{\mathbf{a}_i}_{\mathbf{s}_i} - \mathbb{E}_{\mathbf{a}_j \sim \pi} r^{\mathbf{a}_j}_{\mathbf{s}_j} \right| + \mathbb{E}_{\substack{\mathbf{a}_i \sim \pi \\ \mathbf{a}_j \sim \pi}} d(\mathbb{E}_{s_i' \sim P^{\mathbf{a}_i}_{\mathbf{s}_i}}[s_i'], \mathbb{E}_{s_j' \sim P^{\mathbf{a}_j}_{\mathbf{s}_j}}[s_j']) \\
&= \mathcal{F}_G^\pi(d)(\mathbf{s}_i, \mathbf{s}_j).
\end{aligned}$$

By symmetric,

$$V^\pi(\mathbf{s}_j) - V^\pi(\mathbf{s}_i) \leq \mathcal{F}_G^\pi(d)(\mathbf{s}_i, \mathbf{s}_j).$$

Therefore,

$$|V^\pi(\mathbf{s}_i) - V^\pi(\mathbf{s}_j)| \leq \mathcal{F}_G^\pi(d)(\mathbf{s}_i, \mathbf{s}_j).$$

We have an initial distance $d_0(\mathbf{s}_i, \mathbf{s}_j) = 2 \max_{\mathbf{s},\mathbf{a}} |r^{\mathbf{a}}_{\mathbf{s}}|/(1-\gamma)$ that satisfies $|V^\pi(\mathbf{s}_i) - V^\pi(\mathbf{s}_j)| \leq d(\mathbf{s}_i, \mathbf{s}_j)$, and $\mathcal{F}_G^\pi$ is contraction mapping on d. By repeatedly applying $\mathcal{F}_G^\pi$ on d, d will eventually converge to the fixed-point d_G^π. Because for each iteration $\mathcal{F}_G^\pi(d)$ satisfies $|V^\pi(\mathbf{s}_j) - V^\pi(\mathbf{s}_i)| \leq \mathcal{F}_G^\pi(d)(\mathbf{s}_i, \mathbf{s}_j)$, the fixed-point d_G^π satisfies

$$|V^\pi(\mathbf{s}_i) - V^\pi(\mathbf{s}_j)| \leq d_G^\pi(\mathbf{s}_i, \mathbf{s}_j). \tag{21}$$

□

B Implementation Details

B.1 Objectives

B.1.1 Soft Actor-critic

Our approach is built upon Soft Actor-Critic (SAC) [10]. SAC is an actor-critic algorithm that optimizes a stochastic policy in an off-policy manner. We use the version of SAC that incorporates clipped double-Q trick. Let Q_{θ_i} denote the Q-networks where $i = 1, 2$, $Q_{\bar{\theta}_i}$ be the target Q-networks, and π_θ be the actor network. All Q-networks, target Q-networks and actor network take input of state representation $\phi_\omega(\mathbf{s})$ instead of state $\mathbf{s}$. The loss functions for the Q-networks are

$$\mathcal{L}_Q(Q_{\theta_i}, \phi_\omega) = \mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{D}} \left[\left(Q_{\theta_i}(\phi_\omega(\mathbf{s}), \mathbf{a}) - r - \gamma V_{target}(\mathbf{s}') \right)^2 \right],$$

where $\mathcal{D}$ is the replay buffer, and V_{target} is the target value function. We stop gradients for V_{target} and V_{target} is defined as

$$V_{target}(\mathbf{s}') = \min_{j=1,2} Q_{\bar{\theta}_j}(\phi_{\bar{\omega}}(\mathbf{s}'), \mathbf{a}') - \alpha \log \pi_\theta(\mathbf{a}'|\phi_\omega(\mathbf{s}'))$$

where $\mathbf{a}' \sim \pi_\theta(\cdot|\phi_\omega(\mathbf{s}'))$ is the action at next state. The policy improvement is minimizing the following objective for actor network,

$$\mathcal{J}_\pi(\pi_\theta) = \mathop{\mathbb{E}}_{\mathbf{s}\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{\mathbf{a}\sim\pi_\theta(\cdot|\phi_\omega(\mathbf{s}))} \left[\alpha \log \pi_\theta(\mathbf{a}|\phi_\omega(\mathbf{s})) - \min_{i=1,2} Q_{\theta_i}(\phi_\omega(\mathbf{s}), \mathbf{a}) \right] \right].$$

The loss function for α of SAC is,

$$\mathcal{J}_\alpha(\alpha) = \mathop{\mathbb{E}}_{\mathbf{s}\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{\mathbf{a}\sim\pi_\theta(\cdot|\phi_\omega(\mathbf{s}))} [\alpha \log \pi_\theta(\mathbf{a}|\phi_\omega(\mathbf{s})) - \alpha\bar{\mathcal{H}}] \right], \tag{22}$$

where $\bar{\mathcal{H}} \in \mathbb{R}$ is the target entropy and $\bar{\mathcal{H}} = -|\mathcal{A}|$ in practice.

B.1.2 RAP Distance

We learn the reward function and dynamics model by neural network approximators, in order to approximate RAP distance. Let r_ψ be the learned reward function parameterized by ψ, which outputs a Gaussian distribution, $r_\psi(\mathbf{s}) = \{\widehat{\mu}(r_\mathbf{s}), \widehat{\sigma}(r_\mathbf{s})\}$, where $\widehat{\mu}(r_\mathbf{s})$ and $\widehat{\sigma}(r_\mathbf{s})$ are the mean and the standard deviation, respectively. The loss function for learning r_ψ is

$$\mathcal{L}_r(r_\psi) = \mathop{\mathbb{E}}_{(\mathbf{s},r,)\sim\mathcal{D}} \left[\left(\frac{r - \widehat{\mu}(r_\mathbf{s})}{2\widehat{\sigma}(r_\mathbf{s})} \right)^2 \right].$$

We learn a dynamics model $\widehat{P}_\eta$ parameterized by η to take input of state embedding $\phi(\mathbf{s})$ and action $\mathbf{a}$, and output a Gaussian distribution over the next state embedding, $\widehat{P}_\eta(\phi_\omega(\mathbf{s}), \mathbf{a}) = \{\widehat{\mu}(\widehat{P}^\mathbf{a}_{\phi_\omega(\mathbf{s})}), \widehat{\sigma}(\widehat{P}^\mathbf{a}_{\phi_\omega(\mathbf{s})})\}$, where $\widehat{\mu}(\widehat{P}^\mathbf{a}_{\phi_\omega(\mathbf{s})})$ and $\widehat{\sigma}(\widehat{P}^\mathbf{a}_{\phi_\omega(\mathbf{s})})$ are the mean vector and the standard deviation vector for the predictive next state embedding, respectively. The loss function for $\widehat{P}_\eta$ is

$$\mathcal{L}_P(\widehat{P}_\eta) = \mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} \left[\left(\frac{\phi_\omega(\mathbf{s}') - \widehat{\mu}(\widehat{P}^\mathbf{a}_{\phi_\omega(\mathbf{s})})}{2\widehat{\sigma}(\widehat{P}^\mathbf{a}_{\phi_\omega(\mathbf{s})})} \right)^2 \right].$$

B.2 Learning Algorithm

Algorithm shows the learning steps for learning SAC and RAP jointly.

Algorithm 1 Leaning step for SAC and RAP

Input: Replay Buffer $\mathcal{D}$, Q network Q_{θ_j}, actor pi_ψ, target Q network $Q_{\bar{\theta}_j}$, encoder ϕ_ω. reward function r_ψ, dynamics model $\widehat{P}$
Sample a batch with size B: $\{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)\}_{b=1}^{B} \sim \mathcal{D}$
Update Q network by minimizing $\mathcal{L}_Q(Q_{\theta_i}, \phi_\omega)$
Update actor network by minimizing $\mathcal{J}_\pi(\pi_\theta)$
Update α by minimizing $\mathcal{J}_\alpha(\alpha)$
Update dynamics model $\widehat{P}_\eta$ by minimizing $\alpha_P \mathcal{L}_P(\widehat{P}_\eta)$
Update reward function r_ψ by minimizing $\mathcal{L}_r(r_\psi)$
Update encoder ϕ_ω by minimizing $\alpha_{RAP}\mathcal{L}_{RAP}(\phi_\omega)$
Softly update target Q network: $\bar{\theta}_j = \tau_Q \theta_j + (1 - \tau_Q)\bar{\theta}_j$
Softly update target encoder $\phi_{\bar{\omega}}$: $\bar{\omega} = \tau_\phi \omega + (1 - \tau_\phi)\bar{\omega}$

B.3 Networks and Hyperparameters

We stack convolutional layers followed by a fully-connected layer as state encoder ϕ_ω. The encoder takes the input of a state, where 3 frames are stacked, and outputs a vector, the state representation $\phi_\omega(\mathbf{s})$. The detailed dimensions of each convolutional layer and fully-connected layer are described in Table 2. The Q-network consists of 3 stacked fully-connected layers with 1024 hidden dimensions, and takes the input of state representation $\phi_\omega(\mathbf{s})$ and action $\mathbf{a}$. The actor network takes the input of $\phi_\omega(\mathbf{s})$ and feeds it into three stacked fully-connected layers with 1024 hidden dimensions to output policy π. Both the approximated dynamics model $\widehat{P}$ and the approximated reward function r_ψ are two-layers MLPs with 512 hidden dimensions. ReLU activation is used for every layer. We train the whole model on one NVIDIA A100 GPU. Other hyperparameters are listed in Table 2.

Hyperparameter	DMC	Robosuite	CARLA
Episode length	1000	500	1000
Training steps	1M	500K	100K
Replay buffer capacity	1M	100K	100K
Batch size	128	128	128
Discount factor γ	0.99	0.99	0.99
State dims	9×84×84	9×128×128	9×420×84
Encoder conv kernels	[3,3,3,3]	[3,3,3,3]	[5, 3, 3, 3]
Encoder conv channels	[32,32,32,32]	[32,32,32,32]	[64,64,64,64]
Encoder conv strides	[2,1,1,1]	[2,1,1,1]	[2,2,2,2]
Encoder output dims	100	100	100
Optimizer	Adam	Adam	Adam
Q-networks learning rate	5e-4	1e-3	1e-3
Actor network learning rate	5e-4	1e-3	1e-3
Encoder learning rate	5e-4	1e-3	1e-3
Dynamics model learning rate	5e-4	1e-3	1e-3
Reward function learning rate	5e-4	1e-3	1e-3
log α learning rate	1e-4	1e-4	1e-4
τ_ϕ	0.05	0.05	0.05
τ_Q	0.01	0.01	0.01
Target Q-network update frequency	2	1	2
Actor network update frequency	2	1	2
α_{RAP}	0.5	0.5	0.5
α_P	1e-4	1e-4	1e-4
Actor log std bound	[-10, 2]	[-10, 2]	[-10, 2]

Table 2: Networks dimensions and hyperparameters.

C Additional Details of Experiments

C.1 DeepMind Control Suite

Table 3 shows the action repeat for various DMC tasks.

DMC Task	Action Repeat
Cartpole-Swingup	8
Cheetah-Run	4
Finger-Spin	2
Walker-Walk	2

Table 3: Action repeat used for various tasks in DeepMind Control Suite.

C.2 CARLA

Figure 7 shows an illustration of observation in CARLA. The observation is 420 $\times$ 84 pixels constructed by concatenating five cameras. The action repeat for CARLA is 8.

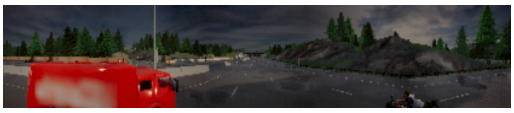


Figure 7: Illustration of observation in CARLA.

D Additional Experiments

D.1 Ablation study on reward correction and learned dynamics

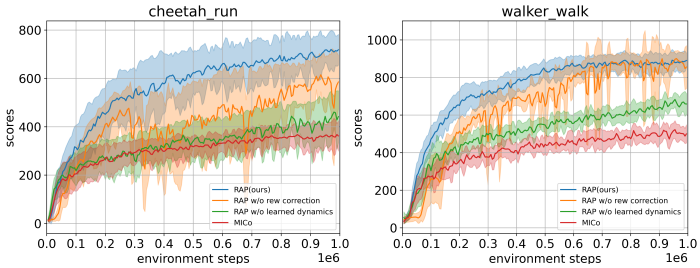


Figure 8: Ablation study results on DMC with natural video background.

We perform this ablation study in order to evaluate the effects of the reward correction and the learned Gaussian dynamics model. Figure 8 shows the experiment results of cheetah-run and walker-walk of DMC with natural video background setting. The curve **RAP w/o rew correction** is our method RAP with removing the reward variance correction. The curve **RAP w/o learned dynamics** is our method but measuring distances with next states that are sampled from MDP instead of a learned dynamics model mean/expectation. This curve can be considered as MICo with reward variance correction. Figure 8 shows that without reward variance correction our method becomes more unstable and gets worse results in cheetah-run. Without a learned dynamics model mean it will meet a larger performance drop but it is still better than MICo.

D.2 Experiment on Simple 2D Highway

We add one more domain which is a 2D highway driving task to further evaluate our method and the baseline methods. The 2D highway task contains a straight road with 4 lanes and 50 other vehicles

running on. Figure 9 shows an illustration of 2D highway and Figure 10 shows the observation which is cropped to a square. The green vehicle is the vehicle that RL agent controls and blue vehicles are controlled by the environment. The RL agent learns to run as far as possible within 600 time steps.

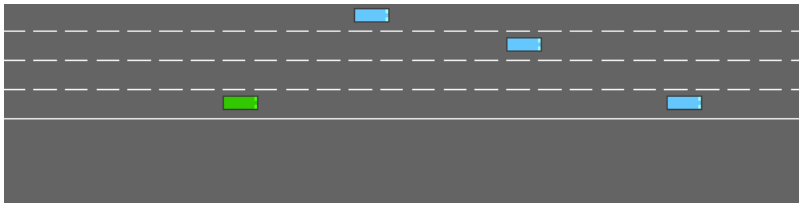


Figure 9: An illustration of 2D highway.

Figure 10: An example of observation in 2D highway task.

Figure 11 shows the learning curves at 500k steps. DBC, DrQ, and the proposed method RAP achieve similar scores. SAC and MICo are slightly worse than the other three. This 2D highway is a simple task so that they converge at around 100K to 200K steps.

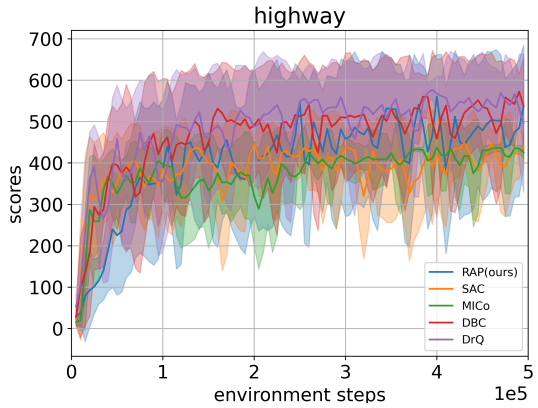


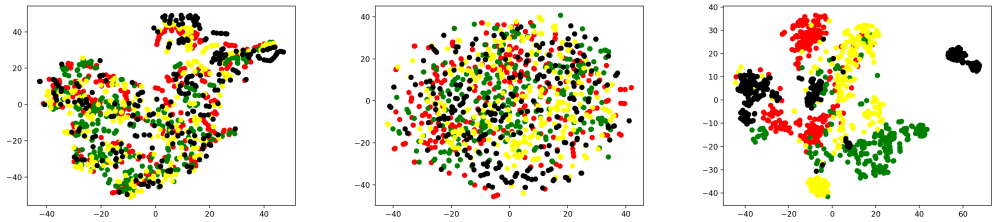
Figure 11: Experimental results on 2D Highway.

E Visualization of Representation Space

We show the t-SNE of the latent representation space of learned RAP, MICo, and DBC. We collect data from cheetah-run with different natural video backgrounds. Figure 12 shows the latent representation space where the data points are collected from four different video backgrounds and marked with four different colors. RAP and MICo can mix different video background which means they can prevent overfit to the background features, while DBC grouping the same color points into several clusters potentially captures the background task-irrelevant features. RAP maps the observations to a structural latent space but MICo maps to a circle latent space which may lose the structure.

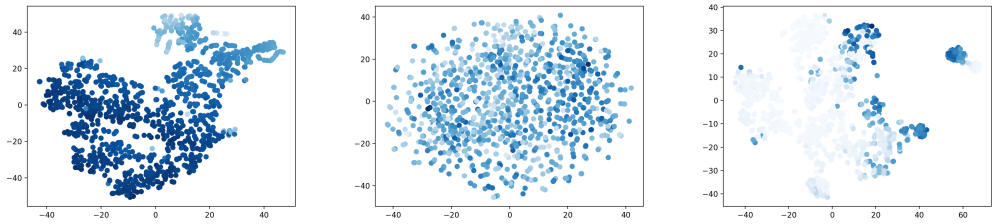
Figure 13 shows the same representation spaces but each data point is colored according to the predicted Q-value. The darker blue indicates higher Q-value and lighter blue means lower Q-value. The representation space of RAP preserves the information for predicting Q-values as it gets dark blue in the bottom-left and light blue in the upper-right. MICo seems to have random Q-values with respect to the representation structure. DBC is likely to predict more extreme high/low Q-values.

Figure 14 shows some representations of the corresponding observations in the latent space. RAP can map similar observations to the nearby locations, while MICo and DBC map them far away from each other.



(a) (b) (c)

Figure 12: t-SNE of the latent representatiinon space of **(a)** RAP (ours), **(b)** MICo, and **(c)** DBC. Different colors indicate different videos in the background of DMC.



(a) (b) (c)

Figure 13: t-SNE of the latent representation space of **(a)** RAP (ours), **(b)** MICo, and **(c)** DBC. The color indicated the predicted Q-value (darker blue means a higher value).

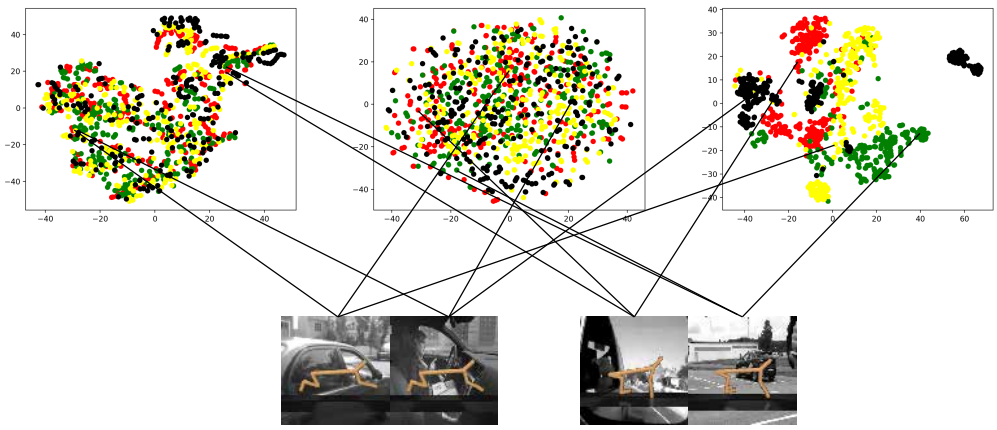


Figure 14: Comparison of the representations of similar observation pairs in latent space. From left to right: RAP, MICo, and DBC

F Explanation of Policy Dependence of RAP Loss

The loss (15) in Section 5.2 should be dependent on the policy. More precisely, the expectation is taken over the transitions sampled from a replay buffer $\mathcal{D}$. The loss (15) should be written as

$$\begin{aligned}\mathcal{L}_{RAP}(\phi_\omega) = \mathbb{E}_{\mathbf{s}_i,\mathbf{s}_j,r^{\mathbf{a}_i}_{\mathbf{s}_i},r^{\mathbf{a}_j}_{\mathbf{s}_j},\mathbf{a}_i,\mathbf{a}_j\sim\mathcal{D}}\Big(\Big(\hat{d}(\phi_\omega(\mathbf{s}_i),\phi_\omega(\mathbf{s}_j))-\gamma\hat{d}(\widehat{\mu}(\widehat{P}^{\mathbf{a}_i}_{\phi_{\bar{\omega}}(\mathbf{s}_i)}),\widehat{\mu}(\widehat{P}^{\mathbf{a}_j}_{\phi_{\bar{\omega}}(\mathbf{s}_j)}))\Big)^2 \\ - \Big(|r^{\mathbf{a}_i}_{\mathbf{s}_i}-r^{\mathbf{a}_j}_{\mathbf{s}_j}|^2-(\widehat{\sigma}(r_{\mathbf{s}_i}))^2-(\widehat{\sigma}(r_{\mathbf{s}_j}))^2\Big)\Big)^2.\end{aligned} \tag{23}$$

The rewards $r^{\mathbf{a}_i}_{\mathbf{s}_i}$ and $r^{\mathbf{a}_j}_{\mathbf{s}_j}$ are sampled from the replay buffer, and the actions $\mathbf{a}_i$ and $\mathbf{a}_i$ that the rewards depend on are sampled from the policy when the rewards are generated from the MDP. Therefore this

	1M	500K	100K	50K	10K
1M	\	6.57	4.53	2.88	2.45
500K	6.57	\	3.43	5.76	5.21
100K	4.53	3.43	\	2.93	2.57
50K	2.88	5.76	2.93	\	1.30
10K	2.45	5.21	2.57	1.30	\

Table 4: Average differences of RAP distances among learned models with different replay buffer sizes.

loss function is dependent on the policy. The learning of the neural network approximated reward variance $\widehat{\sigma}$ is dependent on the data sampled from replay buffer, as described in Appendix B.1.2. This indicates that the learned reward function is also dependent on the policy.

If we use on-policy RL algorithms, then $\mathcal{D}$ would be the set of transitions sampled from the current policy. However, off-policy RL algorithms have better sample efficiency, especially in control tasks. In this paper, we build our method upon SAC which is also an off-policy algorithm. The underlying policy in the replay buffer may not be identical to the current policy. We want to verify that even though the learning of RAP is not dependent on the "truly" on-policy, our approximation is not far away from on-policy and is enough to learn good representations.

We try to reduce the replay buffer capacity. If the replay buffer is small enough, the RL algorithm would become on-policy. However, the smaller replay buffer usually leads to worse performance. We reduce the replay buffer capacity from 1 million to 500K, 100K, 50K, and 10K. Figure 15 shows the learning curves of different replay buffer capacities. It shows that smaller replay buffer sizes lead to lower performance and slower convergence.

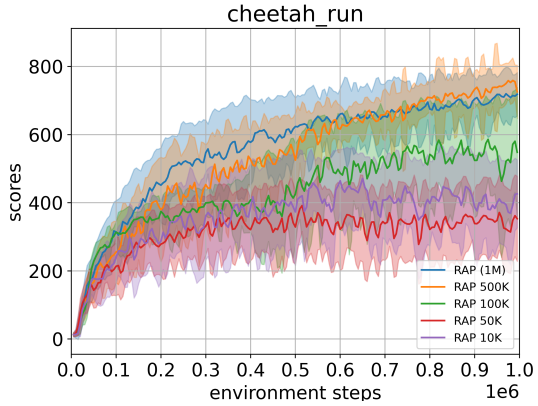


Figure 15: Experiment results in comparison of smaller replay buffer sizes.

We generate trajectories (in total 1K steps) by the optimal policy which is our RAP policy. Then we measure the RAP distance in between the 1K observations by using different models learned by different replay buffer sizes, i.e. 500K, 100K, 50K and 10K. We compare the distances among different replay buffer sizes and show the average differences in Table 4. The differences of distances stay on the same scale. The 1M version which is the proposed method is not far away from the models learned with other buffer sizes. Such difference/variance may be generated by the randomly initialized weights of the neural networks.