# A Metric Space Perspective on Self-Supervised Policy Adaptation

Cristian Bodnar, Karol Hausman, Gabriel Dulac-Arnold, and Rico Jonschkowski

***Abstract*—One of the most challenging aspects of real-world reinforcement learning (RL) is the multitude of unpredictable and ever-changing distractions that could divert an agent from what was tasked to do in its training environment. While an agent could learn from reward signals to ignore them, the complexity of the real-world can make rewards hard to acquire, or, at best, extremely sparse. A recent class of self-supervised methods have shown promise that reward-free adaptation under challenging distractions is possible. However, previous work focused on a short one-episode adaptation setting. In this letter, we consider a long-term adaptation setup that is more akin to the specifics of the real-world and propose a metric space perspective on self-supervised adaptation. We empirically describe the processes that take place in the embedding space during this adaptation process, reveal some of its undesirable effects on performance and show how they can be eliminated. Moreover, we theoretically study how actor-based and actor-free agents can further generalise to the target environment by manipulating the Lipschitz constant of the actor and critic functions.**

***Index Terms*—Continual learning, deep learning methods, reinforcement learning.**

## I. Introduction

REAL-WORLD environments are characterised by an ever-changing set of distractions such as modifications in lighting conditions, object colour variations or evolving backgrounds that are irrelevant for the tasks RL agents should perform. These distractions are often so complex and diverse that they cannot all be anticipated at training time. While further RL training in the target environment could address this problem, RL assumes the existence of a reward signal, which usually requires instrumentation or manual labelling. Another way to address the problem of changing distractions is to have the agent continuously adapt to them – without requiring reward – in a self-supervised manner.

Hansen *et al.* [1] have made important progress in this direction. They propose an RL agent that implicitly adjust its state encoder by training an inverse dynamics network that predicts actions from pairs of abstract state representations. This model is pre-trained via RL in the source environment and then fine-tuned (reward-free) in the target environment that includes the distractions. The fine-tuning adjusts the state representations and improves RL performance is the target domain. While their work opened up this exciting avenue of research, the authors mostly focused on a one-episode adaptation process for a Soft Actor-Critic (SAC) [2] agent.

In this work, we consider a long-term reward-free adaptation scenario both in an actor-critic and actor-free setting and describe the processes that take place in the embedding space during the adaptation phase. Firstly, we demonstrate that while the two environments move towards each other in the embedding space, the original representation of the source environment that the agent was trained on is altered. To address this problem, we propose a parallel training procedure that adjusts the actor and critic weights to compensate for the changes in the state representations. Secondly, we formulate an upper bound on the mismatch between the actions taken between the two environments and show how this can be reduced in practice.

## II. Background

*Problem Statement:* We consider two Partially Observable Markov Decision Processes (POMDPs) [3], [4] $\mathcal{M}_1 = (\mathcal{O}, \mathcal{S}, \mathcal{A}, T, R, \Omega_1, \gamma)$ and $\mathcal{M}_2 = (\mathcal{O}, \mathcal{S}, \mathcal{A}, T, R, \Omega_2, \gamma)$ sharing the same observation space $\mathcal{O}$, state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $T(s'|s,a)$, reward function $R(s,a)$, and discount factor $\gamma$, but with distinct conditional observation densities $\Omega_1(o|s,a)$ and $\Omega_2(o|s,a)$, respectively. $\mathcal{M}_1$ represents the source environment the agent is trained in and $\mathcal{M}_2$ represents the target (adaptation) environment the agent is deployed in. Because we are interested in reward-free adaptation in the target environment, we assume we do not have access to the reward function $R$ when interacting with environment $\mathcal{M}_2$. We formally define our objective as maximizing the expected total reward $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$ in the target environment $\mathcal{M}_2$.

*Environments:* For our experiments, we consider the Distracting Control Suite [5] based on DM Control [6]. We proceed by training in the distraction-free DM Control environments and consider two different environments for adaptation: video backgrounds and random colour changes. In the video background environment, for each episode, a random frame from a set of 10 videos is used in the background. In the colour distraction environment, the colours of all objects are uniformly sampled

Manuscript received October 15, 2020; accepted February 10, 2021. Date of publication March 23, 2021; date of current version April 9, 2021. This letter was recommended for publication by Associate Editor L. Rozo and Editor D. Kulic upon evaluation of the reviewers' comments. *(Corresponding author: Cristian Bodnar.)*

Cristian Bodnar was with the Google Brain, is now with Department of Computer Science and Technology, University of Cambridge, Cambridge CB3 0DG, U.K. (e-mail: cb2015@cam.ac.uk).

Karol Hausman and Rico Jonschkowski are with the Robotics at Google, Mountain View, California 94043 USA (e-mail: karolhausman@google.com; rjon@google.com).

Gabriel Dulac-Arnold is with the Google Research, 75009 Paris, France (e-mail: dulacarnold@google.com).

Digital Object Identifier 10.1109/LRA.2021.3068100

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

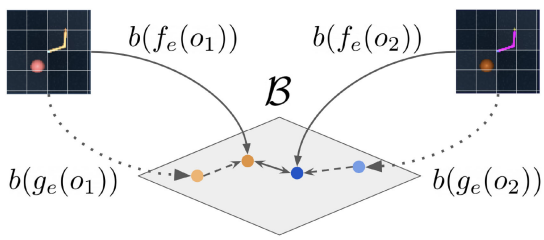


Fig. 1. The evolution of the representations of two matching observations during self-supervised adaptation. The dotted lines indicate the original position of the representations before adaptation. The dashed lines indicate how they move in the embedding space and approach each other.

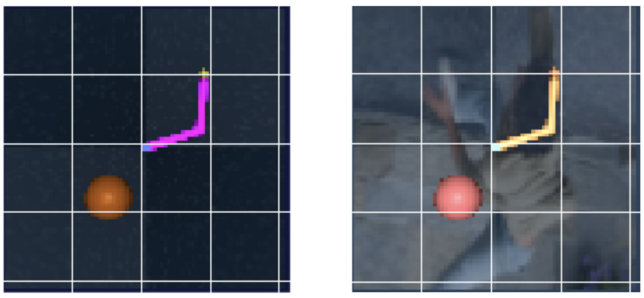

Fig. 2. Color (left) and background (right) distraction environments for the "reacher-easy" task.

from the original colour $\pm 0.5$ in each episode. One frame from each of the environments can be seen in Fig. 2.

*Self-Supervised Adaptation:* Hansen *et al.* [1] consider a soft-actor critic (SAC) [2] model with an auxiliary inverse dynamics loss. The agent is composed of an actor-network ($f_a$), a critic ($f_c$) and an inverse dynamics prediction network ($f_i$), all sharing a convolutional image encoder ($f_e$). Importantly, this architecture allows the policy and the self-supervised prediction network to share features. At training time, the whole model is trained in the source domain using the usual SAC loss combined with an auxiliary inverse dynamics prediction loss, effectively conditioning the shared features on the inverse dynamics of the environment. At deployment time, the SAC objective is dropped and the agent is adapted to the target domain by minimizing only the inverse dynamics loss for the pairs of consecutive observations it encounters in the target environment. The gradients of this loss are propagated only through the inverse dynamics network and the common encoder. The actor and the critic are left untouched by this adaptation procedure (Figure 3). As a result of this fine-tuning, the shared encoder features are adapted to the distractions present in the target environment. Ultimately, this was shown to improve RL performance in the target environment without ever having access to the reward signal. In this work, we aim to improve the understanding of this method while analysing ways the adaptation process could be improved while maintaining the same training procedure. Additionally, we extend our analysis to QT-Opt [7], an actor-free algorithm extensively used in robotic applications.

*Preliminaries:* For our analysis and experiments, we are interested in adapting both in an actor-critic and in an actor-free setting. For the first setting, we use Soft Actor-Critic

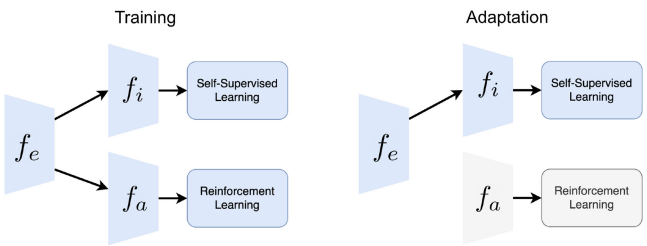


Fig. 3. Schematic of the self-supervised adaptation procedure from [1] (image adapted from the letter). At training time, the SAC agent is minimizing an RL objective alongside an inverse dynamics loss. At adaptation time, the pre-trained agent continues to optimize only the inverse-dynamics loss for the transitions encountered in the target environment.

(SAC) [2], a popular choice in model-free RL. For the latter, we use QT-Opt [7], a Q-Learning [8] based algorithm whose real-world generalization in robotic applications has been well-demonstrated [7], [9].

In our experiments, we use the same neural network architecture based on Yarats *et al.* [10]: We employ an encoder $f_e : \mathcal{O} \to \mathcal{E}$ with eight convolutional layers and ReLU activations that maps from the observation space to the embedding space $\mathcal{E}$. The encoder, is shared by three similar neural network heads $f_i : \mathcal{E} \times \mathcal{E} \to \mathcal{A}$ (inverse dynamics), $f_c : \mathcal{E} \times \mathcal{A} \to \mathbb{R}$ (critic), $f_a : \mathcal{E} \to \mathcal{A}$ (actor – used only in SAC). The actor and critic heads can be seen as a composition of two functions $f_a = h_a \circ b_a$ and $f_c = h_c \circ b_c$. The functions $b_{\{a,c\}} : \mathcal{E} \to \mathcal{B} = [-1, 1]^{100}$ compute a low-dimensional and normalised "bottleneck" vector, while $h_a : \mathcal{B} \to \mathcal{A}$, $h_c : \mathcal{B} \times \mathcal{A} \to \mathbb{R}$. $b_{\{a,c\}}$ compute the action to be taken and the predicted $Q$ value, respectively. The functions $b_{\{a,c\}}$ are formed of 4 convolutional layers with the last containing a layer normalised [11] and tanh-activated bottleneck of dimension 100. We treat the bottleneck space $\mathcal{B}$ as a proxy for underlying state of the observations. $h_{\{a,c\}}$ follow the the bottleneck with two more ReLU activated hidden layers with 1000 neurons each and an output layer of the corresponding dimension for each function. As an additional piece of notation, we use $g$ to refer to state of the networks before adaptation is started (i.e., $g_e$, $g_i$, $g_a$, $g_c$).

We train our model in the source environment for 250 thousand steps using two random crop augmentations per state like in DrQ [12] and [1]. Then, we adapt in the target environment for 50 000 transitions, with one gradient step per frame. We use a batch size of $512 = 64 \times 8$ containing 64 states with eight random crop augmentations for each. We adapt from a replay buffer with capacity 50 000 that is filled initially with 3200 transitions collected by the trained policy.

## III. Towards Understanding Self-Supervised Adaptation

### A. The Embedding Space Dynamics

In this section, we analyse the dynamics in the embedding space for the representations of the two environments in an attempt to explain the internal mechanisms of self-supervised adaptation. For the purpose of our analysis, we equip the bottleneck space $\mathcal{B}$ with a metric $d(e_1, e_2)$ providing the distance

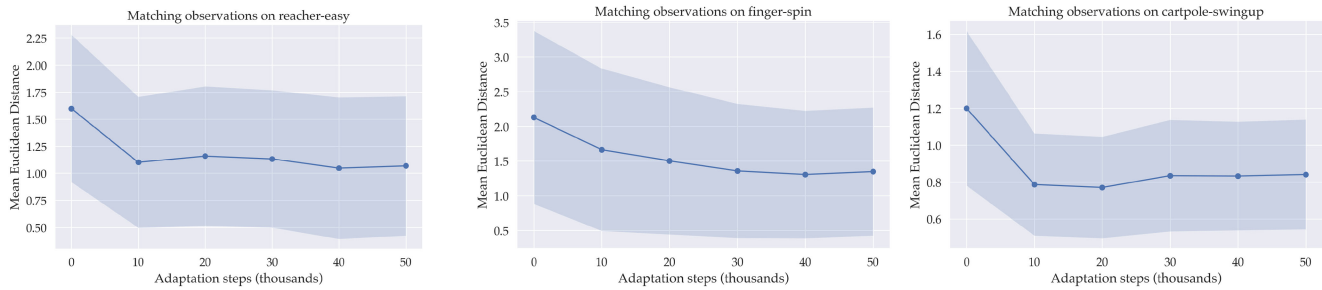


Fig. 4. Distance between the embeddings of matching states in the reacher-easy *(left)*, finger-spin *(middle)* and cartpole-swingup *(right)* environments. We report the mean and standard error over five different episodes. The matching states of the two environments move towards each other during adaptation. Lines show means and shaded areas cover two standard deviations of the mean estimate.

between any two embeddings. Throughout our experiments and theoretical results, we select this distance to be Euclidean distance between the bottleneck representations of the two embeddings such that $d(e_1, e_2) = ||e_1 - e_2||_2$. The use of the Euclidean distance will later be motivated by our theoretical results. The decision to measure it at the bottleneck layer is justified by the fact that the output features of the bottleneck layer are normalised in the range $[-1, 1]$ and, therefore, the Euclidean distance is well behaved. In contrast, this cannot be guaranteed by the convolutions features of the encoder, which are (unbounded) ReLU activations.

Firstly, we hypothesise that the representations corresponding to observations sharing the same underlying state of the two environments become more similar during adaptation. To validate this, we measure the expected distance between the embeddings of observations sharing the same underlying state, formally given by $\mathbb{E}_{s,a}[d(b(f_e(o_1)), b(f_e(o_2)))]$, with $o_1 \sim \Omega_1(s, a)$ and $o_2 \sim \Omega_2(s, a)$. To do so, we collect five matching episodes in the two environments by synchronising the initial state of the two and taking the same actions in both of them.

In Figure 4 we plot this average distance as a function of the adaptation step for the reacher-easy, finger-spin and cartpole-swingup environments with colour distractions. It shows that the auxiliary loss minimization during the adaptation process implicitly minimizes the distance between matching observations of the two environments. Consequently, this allows the agent trained on the source environment to generalize to the target environment. It remains to be examined as part of future work what types of auxiliary objectives implicitly produce a better alignment of the features and how feature alignment could be perhaps explicitly optimised for.

However, even though the two environments move closer to each other in the embedding space as we have just shown, we further hypothesise that the original representations of the source environment are progressively forgotten. A large perturbation in the original representations would cause catastrophic forgetting of the actions learned in the source environment, which would likely propagate to the actions taken in the target environment.

To quantify this forgetting, we measure the expected Euclidean distance between the representations of a set of source observations before adaptation and the representations of the same observations at a later time in the adaptation process. We plot the evolution of these distances during adaptation in Figure 5 for three of the environments. We see that the Euclidean distance monotonically increases during adaptation, meaning that the original policy is gradually forgotten. Later, in Section IV, we will show how forgetting directly hurts performance in the source and target environments during adaptation.

We find that the cosine distance at the encoder-level, which is invariant to the magnitude of the convolutional features, shows the same behaviour as the Euclidean distance at the bottleneck layer. This empirical finding supports the metric choice of our analysis. Due to limited space, we illustrate this only on the reacher-easy environment in Figure 6.

### B. Bounding the Action-Mismatch

We visually summarize these findings in the diagrammatic illustration in Figure 1 for a pair of matching observations of the source and target environments. Starting from this model, in this section, we perform a theoretical analysis of self-supervised adaptation.

Let $g_e : \mathcal{O} \to \mathcal{E}$ be the state of the encoder $f_e$ before adaptation. Then, based on the previous results, we expect the embeddings of two matching observations $o_1$ and $o_2$ to be at some distance $\epsilon_e = d(b(g_e(o_1)), b(f_e(o_2)))$ away from each other. This distance would depend on how much forgetting has taken place and how close to each other the two environments have become. At the same time, we would expect the action mismatch between the two environments to increase with this distance. In what follows, we formalise these intuitions.

*Definition III.1 (Lipschitz continuous function [13]):* Given two metric spaces $(X, d_x)$ and $(Y, d_y)$, a function $f : X \to Y$ is K-Lipschitz continuous if there exists a constant $K \geq 0$ such that $d_y(f(x_1), f(x_2)) \leq K d_x(x_1, x_2)$ for all $x_1$ and $x_2$. We refer to the smallest such $K$ as the Lipschitz constant of the function $f$.

As before, we use the usual Euclidean distance as the metric associated with the domain and co-domain of the functions.

*Proposition III.1:* Let $h_a(e) = [\mu(e), \sigma^2(e)]$ be the components of $f_a$ that specify the mean and variance of the multivariate (normal) action distribution of the SAC actor. Additionally, let $\mu$ and $\sigma^2$ be $K$-Lipschitz continuous and $\sigma_i^2 \geq \sigma_{min}^2$ for all components $i$. Let $e_1$ and $e_2$ be the embeddings of two matching observations with $d(e_1, e_2) = \epsilon_e$. Then we have that $D_{\mathrm{KL}}(\mathcal{N}(\mu(e_1), \sigma(e_1))||\mathcal{N}(\mu(e_2), \sigma(e_2))) \in O((\epsilon_e K)^2)$.

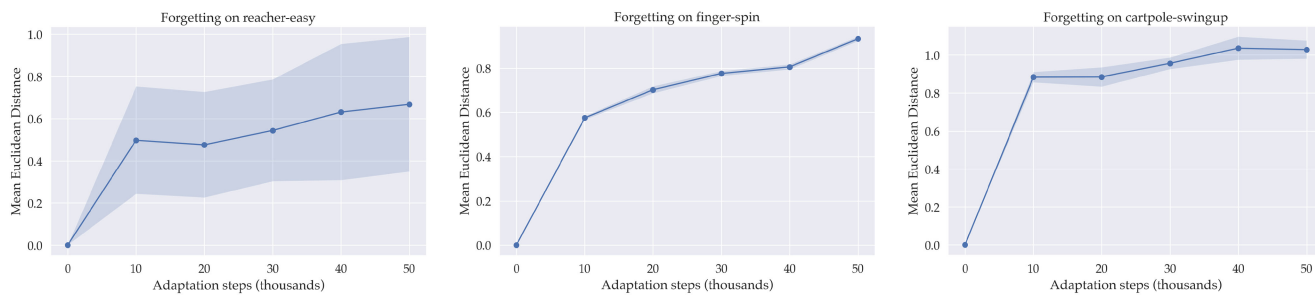


Fig. 5. The distance between the source environment embeddings at different adaptation steps and the same embeddings before adaptation in the reacher-easy *(left)*, finger-spin *(middle)* and cartpole-swingup *(right)* environments. We report the mean and standard error over five different episodes. The original representations are progressively forgotten during adaptation (i.e. the source environment representations are increasingly different from those seen in training).

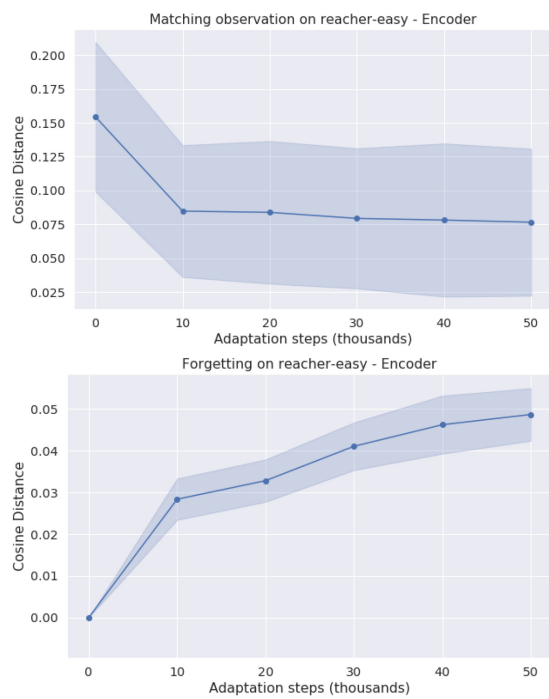


Fig. 6. The cosine distance at the encoder level has a similar qualitative behaviour during adaptation as that of the Euclidean distance.

*Proof:* The KL divergence between two multivariate normal distributions is given by $D_{\mathrm{KL}} = \frac{1}{2}\left[\ln\frac{|\Sigma_2|}{|\Sigma_1|} - D + \mathrm{Tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2-\mu_1)^T\Sigma_2^{-1}(\mu_2-\mu_1)\right]$, where $D$ is the dimension of the random vector. Let $\delta = \max\{\|\mu_1-\mu_2\|, \|\sigma_1^2-\sigma_2^2\|\}$. Throughout the proof, we repeatedly use the fact that $\max(\sigma_{1,i}^2,\sigma_{2,i}^2) \le \min(\sigma_{1,i}^2,\sigma_{2,i}^2)+\delta$ since the Euclidean distance upper-bounds the difference between the individual components of the vectors. We begin by bounding each of the terms in this expression by a function of $K$ and $\epsilon_e$. We start by bounding the logarithm.

$$\ln\frac{|\Sigma_2|}{|\Sigma_1|} = \ln\prod_i\frac{\sigma_{2,i}^2}{\sigma_{1,i}^2} \le \ln\prod_i\frac{\min(\sigma_{1,i}^2,\sigma_{2,i}^2)+\delta}{\min(\sigma_{1,i}^2,\sigma_{2,i}^2)} \tag{1}$$

$$= \sum_i \ln\left(1+\frac{\delta}{\min(\sigma_{1,i}^2,\sigma_{2,i}^2)}\right) \tag{2}$$

$$\le \sum_i \ln\left(1+\frac{\delta}{\sigma_{\min}^2}\right) \tag{3}$$

$$\le \sum_i \frac{\delta}{\sigma_{min}^2} \quad (\text{since } \ln(1+x)\le x, x>-1) \tag{4}$$

$$= \frac{D}{\sigma_{min}^2}\delta \le \frac{D}{\sigma_{min}^2}K\epsilon_e \in O(K\epsilon_e) \tag{5}$$

We can obtain a similar bound for the trace term.

$$\mathrm{Tr}(\Sigma_2^{-1}\Sigma_1) = \sum_i\frac{\sigma_{1,i}^2}{\sigma_{2,i}^2} \le \sum_i\frac{\min(\sigma_{1,i}^2,\sigma_{2,i}^2)+\delta}{\min(\sigma_{1,i}^2,\sigma_{2,i}^2)} \tag{6}$$

$$= D + \sum_i\frac{\delta}{\min(\sigma_{1,i}^2,\sigma_{2,i}^2)} \tag{7}$$

$$\le D + D\frac{\delta}{\sigma_{min}^2} \le D + D\frac{K\epsilon_e}{\sigma_{min}^2} \in O(K\epsilon_e) \tag{8}$$

Finally, we bound the last term of the KL divergence:

$$(\mu_2-\mu_1)^T\Sigma_2^{-1}(\mu_2-\mu_1) \tag{9}$$

$$\le \frac{1}{\sigma_{min}^2}(\mu_2-\mu_1)^T(\mu_2-\mu_1) \le \frac{1}{\sigma_{min}^2}\delta^2 \tag{10}$$

$$\le \frac{1}{\sigma_{min}^2}(K\epsilon_e)^2 \in O((K\epsilon_e)^2) \tag{11}$$

Putting it all together, we have $D_{\mathrm{KL}} \in O((K\epsilon_e)^2)$. ■

This proposition formalises the intuition that the closer the two states are and the smoother the actor function is, the more similar the two action distributions are going to be.

Obtaining a similar bound for QT-Opt is more challenging. Because the actions are selected through a maximisation operation $\arg\max_a Q(s,a)$, any potential bound on the action mismatch would depend on the landscape of $Q(s,\cdot)$. This is stated formally in the following proposition.

*Proposition III.2:* Let $|h_c(e_1,a_1)-h_c(e_1,a_2)| = \Delta_1$ be the predicted Q-value difference for actions $a_1$ and $a_2$ at observation embedding $e_1$ with $h_c(e_1,a_1) > h_c(e_1,a_2)$. Let $e_2$ be the embedding of another observation. Assume we have a

metric over $\mathcal{E} \times \mathcal{A}$ with the property that $d([e_1, a_1], [e_2, a_1]) = d([e_1, a_2], [e_2, a_2]) = d(e_1, e_2) = \epsilon_e$. Additionally, let $f_c$ be $K$-Lipschitz continuous. Then if $\epsilon_e < \Delta_1/(2K)$, the order between predicted $Q$ values at $e_2$ is preserved and we have $h_c(e_2, a_1) > h_c(e_2, a_2)$.

*Proof:* Let $|h_c(e_1, a_1) - h_c(e_2, a_1)| = \Delta_2$ and $|h_c(e_1, a_2) - h_c(e_2, a_2)| = \Delta_3$. Then if $\Delta_1 > \Delta_2 + \Delta_3$, the order is preserved, since the summed variation in the two predicted $Q$ values at the state $s_2$ compared to $s_1$ is insufficient to change the order between the two. Using the Lipschitz property of the $Q$ function $f_c$, we can require a stronger inequality to be satisfied $\Delta_2 + \Delta_3 \leq K[d([e_1, a_1], [e_2, a_1]) + d([e_1, a_2], [e_2, a_2])] = 2K\epsilon_e < \Delta_1$. It follows that the order between $Q$ values is preserved if $\epsilon_e < \frac{\Delta_1}{2K}$ ■

This result says the order between any two predicted $Q$ values can be preserved for states in a certain neighbourhood given by an open ball $B_{(\epsilon_e, e_1)}$ centred at $e_1$. To increase the size of this ball, we would like to maximise $\Delta_1/(2K)$. A first idea would be to increase the value of $f_c(e_1, a_1)$ as much as possible and decrease the other $Q$ values in order to increase $\Delta_1$. However, $\Delta_1$ also depends on $K$ as shown in the next result.

*Proposition III.3:* Let $d$ be a metric with $d([e_1, a_1], [e_1, a_2]) = d([e_2, a_1], [e_2, a_2]) = d(a_1, a_2) = \epsilon_a$. Then $\Delta_1/(2K)$ can be at most $\frac{\epsilon_a}{2}$.

*Proof:* This follows directly from the Lipschitz continuity of $f_c$ and we have $\epsilon_e < \frac{\Delta_1}{2K} \leq \frac{Kd([e_1,a_1],[e_1,a_2])}{2K} = \frac{1}{2}\epsilon_a$ ■

This shows that the best we could do for a $K$-Lipschitz critic function is to have a unimodal landscape, where the $Q$ values of other actions strictly decrease with the distance from the optimal action.

From the perspective of Lipschitz continuity, these results describe how one can manipulate the policy's behaviour in the target environment by exploiting the "stiffness" of the manifold produced by the actor or critic functions, where the "stiffness" is given by the Lipschitz constant. While a low Lipschitz constant gives more power to control the behaviour in the target environment, it can affect the performance in the source environment if the actor and critic functions are not flexible enough. Therefore, these trade-offs must be carefully considered.

From a distance minimization perspective, it is clear that one should try to reduce $d(b(g_e(o_1)), b(f_e(o_2)))$ as much as possible to reduce the action mismatch between the two embeddings. To that end, we can use the following remark.

*Remark:* From the triangle inequality we have that

$$d(b(g_e(o_1)), b(f_e(o_2))) \leq d(b(g_e(o_1)), b(f_e(o_1))) + d(b(f_e(o_1)), b(f_e(o_2))) \quad (12)$$

This explicitly upper bounds $d(b(g_e(o_1)), b(f_e(o_2)))$ on the amount of forgetting that has taken place (the first term) and how well the auxiliary objective has brought the two environments closer to each other (the second term). As shown in Section III-A, the inverse dynamics objective of [1] implicitly minimizes the second term and undesirably increases the first. Therefore, one would like to keep $d(b(g_e(o_1)), b(f_e(o_1)))$ as close to zero as possible. However, this could interfere with the self-supervised objective. In the next section, we propose a better alternative that allows us to consider only $d(b(f_e(o_1)), b(f_e(o_2)))$.

### C. Behaviour Cloning-Based Adaptation

To address the catastrophic forgetting problem, we consider a parallel data collection strategy together with a loss split across the two environments. The loss combines the self-supervised objective in the target environment with a behaviour cloning loss in the source environment, which ensures that $h_a(b_a(g_e(o))) \approx h_a(b_a(f_e(o)))$ for SAC and $h_c(b_c((g_e(o)))) \approx h_c(b_c(f_e(o)))$ for QT-Opt, even though $b(g_e(o)) \neq b(f_e(o))$. Therefore, the action mismatch would depend approximately only on $d(b(f_e(o_1)), b(f_e(o_2)))$.

For SAC, we clone the weights of the encoder and the actor into networks $g_e = f_e$ and $g_a = f_a$ before adaptation. Then, at adaption time, we use $g_a(g_e)$ as a target action to approximate on states coming from the source environment. The gradient of this loss is propagated only through the actor network. Concurrently, we continue minimizing the inverse dynamics loss as before, with gradients propagated through the inverse dynamics network and the encoder. Ultimately, this results in the following loss

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_1 + \mathcal{L}_2 \\ &= \mathbb{E}_{o\sim\mathcal{D}_1}\left[D_{\mathrm{KL}}[g_a(g_e(o))||f_a(\bar{f}_e(o)]\right] \\ &\quad + \mathbb{E}_{(o_t,a_t,o_{t+1})\sim\mathcal{D}_2}\left[(f_i(f_e(o_t), f_e(o_{t+1})) - a)^2)\right],\end{aligned}$$

where $\mathcal{D}_1$ and $\mathcal{D}_2$ represent the replay buffer for environments $\mathcal{M}_1$ and $\mathcal{M}_2$ and $\bar{f}_e$ denotes the gradients are stopped from propagating through the encoder. The loss makes the actor adjust to the changes in the original representation to preserve its original behaviour.

Similarly, for QT-Opt, we use the target encoder and target critic network from the training stage $f_e^T$, $f_c^T$ and use it as a target for the $Q$ values of the state-action pairs on the source environment. As with SAC, we backpropagate this additional loss only through $f_c$, but not through $f_e$. We obtain a similar loss function:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_1 + \mathcal{L}_2 \\ &= \mathbb{E}_{o,a\sim\mathcal{D}_1}\left[(f_c(\bar{f}_e(o), a) - f_c^T(f_e^T(o), a))^2\right] \\ &\quad + \mathbb{E}_{(o_t,a,o_{t+1})\sim\mathcal{D}_2}\left[(f_i(f_e(o_t), f_e(o_{t+1})) - a)^2)\right],\end{aligned}$$

This loss makes the critic $f_c$ adjust its weights to compensate for the adjustment in the representations and predict the same Q-values in the source environment.

## IV. Results

### A. Behaviour Cloning-Based Adaptation

We compare the proposed method with an online adaptation process [1], which adapts only on the latest collected transition from the target environment with multiple crop augmentations. Another baseline is the replay buffer-based adaptation previously described in the experimental section. Additionally, we include for reference the original performance of the agent

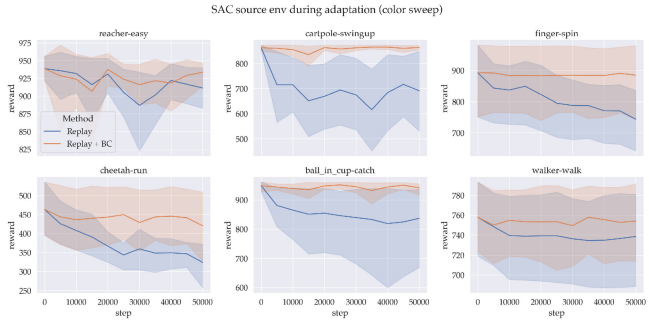


Fig. 7. SAC performance in the source environments during adaptation in the color sweep environments. We report the mean and standard error across five independently trained and adapted agents. Behaviour cloning stops the catastrophic forgetting of the policy obtained from the training phase and performance stays constant.

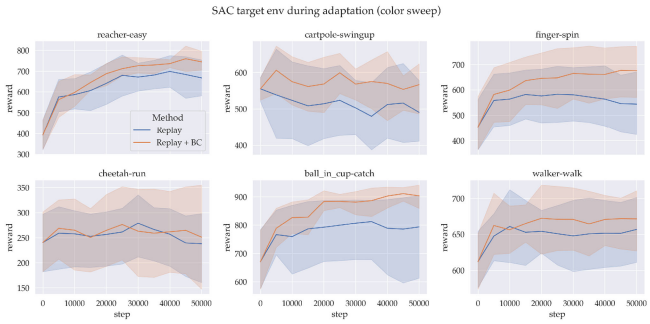


Fig. 8. SAC adaptation results for the color sweep environment. We report the mean and standard error across five independently trained and adapted agents. Behaviour cloning improves the adaptation performance across all environments.

before adaptation and an agent trained normally using rewards in the target environment.

In Section III-A we quantitatively showed how the original representations in the source environment are progressively forgotten. In Figure 7, we now measure the effects this has on the source environment performance for SAC. When using the vanilla adaptation, the total reward of the agent in the source environment degrades significantly with the adaptation step. In contrast, the agent adapting with the behaviour cloning loss completely avoids the degradation in performance observed in the vanilla model and, in average, maintains its original performance. The source environment results are similar for QT-Opt (see Appendix).

The consequences of catastrophic forgetting are not limited only to the source environment. In the target environment, using behaviour cloning to avoid catastrophic forgetting results in an improvement in the vast majority of environments. The evolution of the rewards during adaptation in the colour distraction environments is shown in Figure 8. For QT-Opt, the catastrophic forgetting is significantly attenuated in most environments, but not completely reduced because even tiny differences in the $Q$ values can make the maximisation step select another action (see Appendix). Table I summarises the final adaptation results across all models and environments. More figures describing the reward evolution during adaptation can be found in Appendix.

At the same time, we notice that the online adaptation intensifies the forgetting process when adapting for multiple episodes

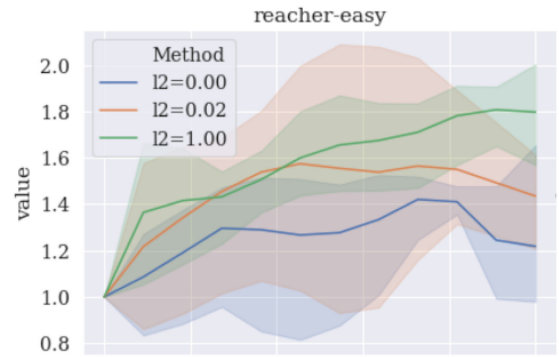


Fig. 9. Normalised adaptation performance with Lipschitz constraint. Increased function smoothness corresponds to increased generalisation.

and performance consequently degrades. While the replay-buffer based adaptation works better, forgetting still happens.

### B. Lipschitz Continuity

In this section, we are interested in exploiting the relationship between the smoothness of the actor function and the action-mismatch between the two environments in order to improve the performance in the adaptation environment. We enforce a Lipschitz constraint on the dense layers of the SAC actor that follow the bottleneck layer. While more sophisticated methods exist to do so [14], we simply reduce the magnitude of the weights with an $L_2$ regularization loss. We train the agent with this additional regularisation loss weighted by a coefficient $l_2$ and then we adapt it in the target environment as before. We show adaption results in Figure 9 for the reacher-easy environment. In accordance with our hypothesis, we find that higher $l_2$ coefficients, corresponding to a smaller $K$, make the agent increase its adaptation performance from a factor of 1.2 to 1.8.

## V. Related Work

A parallel stream of work has focused on adapting to distractions in the presence of rewards. In this setting, states and observations can be aggregated if they cannot be distinguished with respect to the reward sequences they produce under any action sequences. More generally, *bisimulation* metrics [15] can be used to quantitatively measure this behavioural similarity. However, they are difficult to compute [16], [17]. Recently, Zhang *et al.* [18] have proposed learning distraction invariant representations by learning an embedding space that respects the bisimulation metric between the observations. Similarly, Gelada *et al.* [19] learn in an unsupervised manner a latent MDP whose norm they theoretically connect to bisimulation metrics.

In contrast, our work is part of a recent line of research on reward-free adaptation. Closer to the approach we analyse in our letter, Tzeng *et al.* [20] use an adversarial procedure to achieve a similar outcome of aligning the features of the two environments by fooling a discriminator that is trained to distinguish between the two. Another class of methods tries to train robust policies by applying various types of domain randomizations [21]–[23]. While these methods have been successful in making the representations more robust, they cannot possibly anticipate the full set of distractions from a real-world setting.

TABLE I
SAC AND QT-OPT MEAN TOTAL REWARD AND STANDARD ERROR AFTER ADAPTATION IN THE COLOR AND BACKGROUND DISTRACTION ENVIRONMENTS. THE STATISTICS ARE COMPUTED OVER FIVE INDEPENDENTLY TRAINED AND ADAPTED AGENTS. BEHAVIOUR CLONING (BC) EITHER PERFORMS SIMILARLY OR BETTER THAN THE VANILLA ADAPTATION (REPLAY OR ONLINE)

| Benchmark | Method | Reacher-easy | Cartpole-swingup | Finger-spin | Cheetah-run | Ball-in-cup catch | Walker-Walk |
|---|---|---|---|---|---|---|---|
| SAC: Color | Online | 182.06 ± 65.56 | 93.52 ± 18.30 | 136.75 ± 49.43 | 87.73 ± 35.87 | 271.63 ± 150.27 | 331.90 ± 83.07 |
| | Replay | 667.94 ± 52.65 | 490.51 ± 50.30 | 544.24 ± 73.62 | **237.92 ± 40.43** | 793.65 ± 88.79 | **656.78 ± 26.73** |
| | Replay + BC | **744.74 ± 26.50** | **566.80 ± 42.09** | **676.01 ± 59.60** | **251.19 ± 59.48** | **903.19 ± 23.40** | **671.48 ± 24.47** |
| | RL on Target | 210.04 ± 27.57 | 605.23 ± 12.49 | 719.59 ± 57.72 | 310.79 ± 40.69 | 777.85 ± 36.27 | 616.82 ± 48.30 |
| | Original | 392.64 ± 42.37 | 555.23 ± 18.23 | 453.86 ± 58.35 | 240.46 ± 33.46 | 670.99 ± 61.49 | 612.17 ± 21.96 |
| SAC: Background | Online | 122.12 ± 24.43 | 97.60 ± 3.74 | 21.24 ± 16.94 | 28.87 ± 13.74 | 121.56 ± 28.70 | 42.79 ± 16.75 |
| | Replay | 527.40 ± 51.11 | **260.28 ± 38.94** | 265.82 ± 41.94 | **173.55 ± 9.96** | 310.83 ± 67.42 | **355.68 ± 46.89** |
| | Replay + BC | **624.60 ± 85.84** | **253.76 ± 47.59** | **471.85 ± 47.20** | 118.00 ± 20.81 | **463.19 ± 69.65** | **355.45 ± 44.38** |
| | RL on Target | 93.00 ± 10.54 | 192.13 ± 19.07 | 184.36 ± 51.53 | 192.13 ± 19.07 | 78.64 ± 6.74 | 250.10 ± 41.53 |
| | Original | 166.26 ± 8.91 | 189.80 ± 25.50 | 108.00 ± 8.18 | 54.20 ± 6.01 | 129.92 ± 23.58 | 179.17 ± 29.81 |
| QT-Opt: Color | Online | 468.90 ± 138.63 | 116.10 ± 15.88 | 240.03 ± 39.18 | 66.06 ± 21.67 | 270.39 ± 95.18 | 87.04 ± 29.31 |
| | Replay | **855.81 ± 31.08** | **594.91 ± 32.96** | 570.85 ± 72.38 | **184.30 ± 64.93** | **830.78 ± 56.30** | 358.25 ± 11.61 |
| | Replay + BC | **842.74 ± 7.45** | **609.19 ± 24.79** | **746.45 ± 49.33** | **227.09 ± 51.75** | **862.53 ± 41.25** | **397.48 ± 11.41** |
| | RL on Target | 873.50 ± 17.48 | 556.47 ± 44.09 | 591.53 ± 66.04 | 197.33 ± 63.56 | 825.32 ± 57.54 | 362.01 ± 12.16 |
| | Original | 481.16 ± 52.45 | 541.13 ± 13.16 | 385.83 ± 35.32 | 150.11 ± 42.54 | 490.43 ± 116.99 | 296.68 ± 7.56 |
| QT-Opt: Background | Online | 197.57 ± 55.10 | 89.51 ± 5.48 | 174.29 ± 77.61 | 109.47 ± 23.58 | 109.55 ± 14.82 | 75.83 ± 23.74 |
| | Replay | **657.22 ± 104.57** | **249.04 ± 24.88** | 523.18 ± 27.06 | **89.57 ± 26.60** | **357.04 ± 109.15** | 210.67 ± 12.58 |
| | Replay + BC | **693.68 ± 90.79** | **274.48 ± 12.11** | **584.23 ± 29.78** | **81.21 ± 16.54** | **437.40 ± 84.99** | **301.87 ± 15.15** |
| | RL on Target | 695.34 ± 119.33 | 261.46 ± 17.55 | 542.61 ± 33.31 | 84.90 ± 24.07 | 424.07 ± 119.80 | 215.25 ± 16.45 |
| | Original | 143.20 ± 12.80 | 190.50 ± 7.18 | 146.25 ± 16.46 | 29.77 ± 4.61 | 92.41 ± 20.14 | 133.23 ± 11.07 |

## VI. CONCLUSION

In this work, we take a closer look at the class of self-supervised adaptation methods introduced by [1] in a long-term adaptation setting. We propose a metric-space picture of the internal process that takes place in the embedding space during adaptation and discover an undesirable aspect of this process: the progressive forgetting of the original representations. We propose a method based on behaviour cloning to fix this problem. Additionally, we quantify the mismatch between actions taken in corresponding states of the two environments. As a next step, we aim to apply these techniques to real-world robotic applications and distractions specific to these environments [24].

## APPENDIX
## ADDITIONAL PLOTS

We include the source environment performance for SAC adapting to background distractions (Figure 10) and QT-Opt adapting to color distractions (Figure 11). Additionally, we include the target environment performance for SAC adapting to background distractions (Figure 12) and QT-Opt adapting to color distractions (Figure 13).

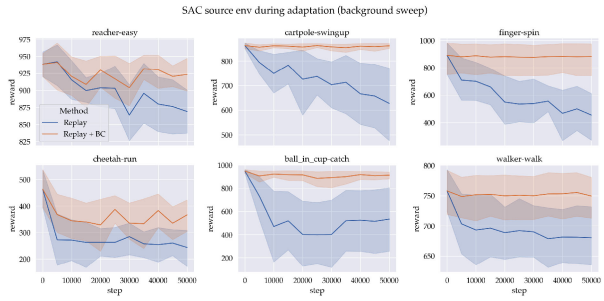


Fig. 10. SAC performance in the source environments during background adaptation. Behaviour cloning generally attenuates the catastrophic forgetting of the policy obtained from the training phase and performance stays constant.

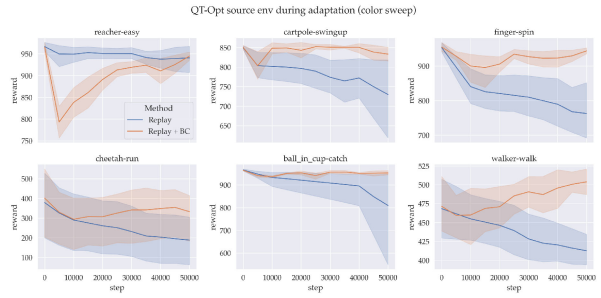


Fig. 11. QT-Opt performance in the source environments during color adaptation. Behaviour cloning generally attenuates the catastrophic forgetting of the policy obtained from the training phase and performance stays constant.

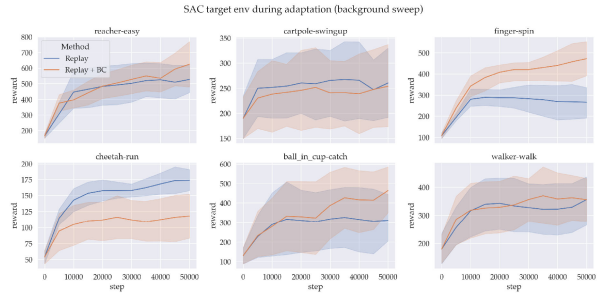


Fig. 12. SAC adaptation results for the background sweep environment. Behaviour cloning either improves or displays the same performance with the exception of the cheetah environment.

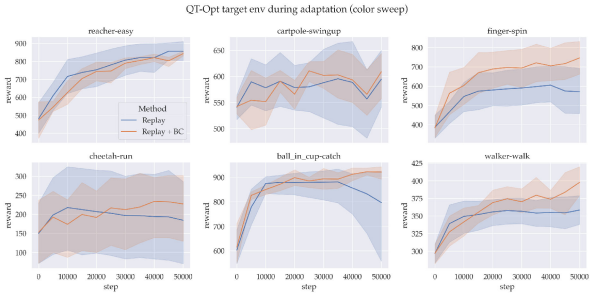


Fig. 13. QT-Opt adaptation results for the color sweep environment. Behaviour cloning either improves or displays the same performance with the exception of the cheetah environment.

## ACKNOWLEDGMENT

The authors would like to thank Oscar Ramirez for helping to set up reinforcement learning experiments.

## REFERENCES

[1] N. Hansen *et al.*, "Self-supervised policy adaptation during deployment," in *Proc. Int. Conf. Learn. Representations*, 2021.

[2] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. 35th Int. Conf. Mach. Learn.* ser. in Proc. Mach. Learn. Res., J. Dy and A. Krause, Eds., vol. 80, Jul. 2018, pp. 1861–1870.

[3] K. J. Astrom, "Optimal control of markov processes with incomplete state information," *J. Math. Anal. Appl.*, vol. 10, no. 1, pp. 174–205, 1965.

[4] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, no. 1, pp. 99–134, 1998.

[5] A. Stone, O. Ramirez, K. Konolige, and R. Jonschkowski, "The distracting control suite - a challenging benchmark for reinforcement learning from pixels," 2021, *arXiv:2101.02722*.

[6] Y. Tassa *et al.*, "Deepmind control suite," 2018, *arXiv:1801.00690*.

[7] D. Kalashnikov *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proc. 2nd Conf. Robot Learn. Ser. Proc. Machine Learn. Res.*, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87, Oct. 2018, pp. 651–673,.

[8] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3/4, pp. 279–292, 1992.

[9] C. Bodnar, A. Li, K. Hausman, P. Pastor, and M. Kalakrishnan, "Quantile qt-opt for risk-aware vision-based robotic grasping," in *Proc. Robot.: Sci. Syst.*, Corvalis, Oregon, USA, Jul. 2020.

[10] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning from images," 2019, *arXiv:1910.01741*.

[11] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

[12] D. Yarats, I. Kostrikov, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," in *Proc. Int. Conf. Learn. Representations*, 2021.

[13] M. O'Searcoid, *Metric Spaces, Ser. Springer Undergraduate Math. Ser*. London: Springer, 2006.

[14] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5767–5777.

[15] N. Ferns and D. Precup, "Bisimulation metrics are optimal value functions," in *Proc. 13th Conf. Uncertainty Artif. Intell., Ser. UAI'14*. Arlington, Virginia, USA: AUAI Press, 2014, pp. 210–219.

[16] N. Ferns, P. Panangaden, and D. Precup, "Bisimulation metrics for continuous markov decision processes," *SIAM J. Comput.*, vol. 40, no. 6, p. 1662–1714, Dec. 2011.

[17] J. Taylor, D. Precup, and P. Panagaden, "Bounding performance loss in approximate MDP homomorphisms," in *Proc. Adv. Neural Inf. Process. Syst.* 21, D. D. Koller Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2009, pp. 1649–1656. [Online]. Available: http://letters.nips.cc/letter/3423-bounding-performance-loss-in-approximate-mdp-homomorphisms.pdf

[18] A. Zhang, R. T. McAllister, R. Calandra, Y. Gal, and S. Levine, "Learning invariant representations for reinforcement learning without reconstruction," in *Proc. Int. Conf. Learn. Representations*, 2021.

[19] C. Gelada, S. Kumar, J. Buckman, O. Nachum, and M. G. Bellemare, "DeepMDP: Learning continuous latent space models for representation learning," *Ser. Proc. Mach. Learn.* Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09-15 Jun. 2019, pp. 2170–2179. [Online]. Available: http://proceedings.mlr.press/v97/gelada19a.html

[20] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7167–7176.

[21] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 1–8.

[22] F. Ramos, R. Possas, and D. Fox, "Bayessim: Adaptive domain randomization via probabilistic inference for robotics simulators," in *Robot.: Sci. Syst. XV*, Univ. Freiburg, Freiburg im Breisgau, Germany, Jun. 22-26, 2019, A. Bicchi, H. Kress-Gazit, and S. Hutchin-son, Eds., 2019.

[23] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 23–30.

[24] R. Julian, B. Swanson, G. S. Sukhatme, S. Levine, C. Finn, and K. Hausman, "Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning," in *Proc. Conf. Robot Learn.*, 2020.