Continuous MDP Homomorphisms and Homomorphic Policy Gradient

Sahand Rezaei-Shoshtari
McGill University and Mila

Rosie Zhao
McGill University and Mila

Prakash Panangaden
McGill University and Mila

David Meger
McGill University and Mila

Doina Precup
McGill University, Mila, and DeepMind

Abstract

Abstraction has been widely studied as a way to improve the efficiency and generalization of reinforcement learning algorithms. In this paper, we study abstraction in the continuous-control setting. We extend the definition of MDP homomorphisms to encompass continuous actions in continuous state spaces. We derive a policy gradient theorem on the abstract MDP, which allows us to leverage approximate symmetries of the environment for policy optimization. Based on this theorem, we propose an actor-critic algorithm that is able to learn the policy and the MDP homomorphism map simultaneously, using the lax bisimulation metric. We demonstrate the effectiveness of our method on benchmark tasks in the DeepMind Control Suite. Our method's ability to utilize MDP homomorphisms for representation learning leads to improved performance when learning from pixel observations.

1 Introduction

For reinforcement learning from high-dimensional observations, such as images, learning a simpler problem by abstraction from the original problem can be critical [2, 52]. The coupling between states, actions and rewards complicates learning RL abstractions. MDP homomorphisms [68, 69, 71, 62] define a concept that allows one to exploit symmetries, yielding behavioral equivalence and preserving values, while giving the potential to arrive at a substantially smaller MDP. Recent works [87, 89, 12] have shown that using MDP homomorphisms is effective to guide learning in discrete problems. This paper is one of the first to consider MDP homomorphisms in the continuous-control setting and to develop actor-critic algorithms with a tightly integrated state-action abstraction. To that end, we identify and answer a series of key challenges:

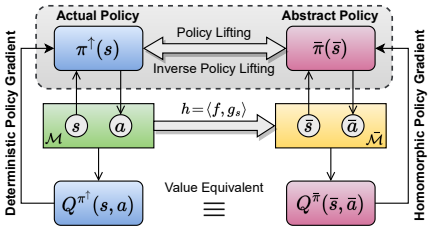


Figure 1: Schematics of our method. The actual MDP $\mathcal{M}$ is used to train $Q^{\pi^{\uparrow}}$ and update $\pi^{\uparrow}$ with DPG, while the abstract MDP $\overline{M}$ is used to train $Q^{\overline{\pi}}$ and update $\overline{\pi}$ with HPG. $\overline{\mathcal{M}}$ is the MDP homomorphic image of $\mathcal{M}$ obtained by learning the homomorphism map $h=(f, g_s)$. Policies $\pi^{\uparrow}$ and $\overline{\pi}$ can be derived from each other.

Can MDP homomorphisms be defined on continuous state and action spaces? Our first contribution is to define continuous MDP homomorphisms on continuous state and action spaces, which requires more intricate proofs that do not follow in any direct way from the finite case and requires tools from measure theory and differential geometry.

Can MDP homomorphisms be tightly integrated into the policy gradient? Our second contribution is the derivation of the *homomorphic policy gradient* (HPG) theorem to closely integrate the

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

abstract MDP into the policy gradient. Importantly, we rigorously prove that performing HPG on the abstract MDP is equivalent to performing the deterministic policy gradient (DPG) on the actual MDP. Therefore, HPG can act as an additional gradient estimator capable of utilizing approximate symmetries for improved sample efficiency. To derive these results, we prove that continuous MDP homomorphisms preserve value functions [35], which in turn enables their use for policy evaluation.

Can MDP homomorphisms be learned simultaneously with the optimal policy in a practical deep reinforcement learning algorithm? We propose a deep actor-critic algorithm, depicted in Figure 1, based on HPG, referred to as *Deep Homomorphic Policy Gradient* (DHPG), that unifies state and action abstractions. DHPG is able to simultaneously learn the policy and the homomorphism map using the lax bisimulation metric [83], a metric for measuring the equivalence of state-action pairs under an MDP homomorphism relation. We empirically show that state-action abstractions learned through MDP homomorphisms provide a natural inductive bias for representation learning.

Despite the existence of well-studied abstraction notions, learning state abstractions in a scalable fashion for continuous control remains a key challenge. In contrast to previous works on learning MDP homomorphisms [87, 89, 12], our algorithm is readily applicable to continuous actions, and compared to previous works guided by bisimulations [98, 31, 48], our algorithm leads to more robust solutions, as suggested by our empirical results. The bisimulation relation [59, 50, 13, 22, 32] and bisimulation metrics [23, 28, 27, 29] do not allow abstracting actions as they require exact matching of actions, whereas MDP homomorphisms and equivalently the lax bisimulation metric [83] remove this strong limitation, giving them greater modeling flexibility. Most importantly, the key difference between prior works [87, 89, 98, 31, 48] and our method is the homomorphic policy gradient theorem which allows for a tight integration of the abstraction notion into the policy gradient that theoretically motivates using the abstract MDP for policy optimization in the actual MDP. Our contributions are:

1. Defining continuous MDP homomorphisms on continuous state and action spaces, using tools from measure theory and differential geometry.
2. Proving that continuous MDP homomorphisms preserve value and optimal value functions.
3. Deriving the homomorphic policy gradient theorem.
4. Developing a deep actor-critic algorithm for learning the optimal policy simultaneously with the MDP homomorphism map in challenging continuous control problems.

DHPG improves upon strong baselines on pixel observations [95, 98] on DM Control, and our visualizations demonstrate the potential of MDP homomorphisms in learning structured representations that can preserve values and represent the minimal MDP image. To the best of our knowledge, this is the first homomorphic policy gradient derivation and the first work to define and scale up MDP homomorphisms to continuous visual control problems. Our code is publicly available at https://github.com/sahandrez/homomorphic_policy_gradient.

2 Background

2.1 Markov Decision Processes

We consider the standard MDP that is defined by a 5-tuple $(\mathcal{S}, \mathcal{A}, \tau_a, R, \gamma)$, with *state space* $\mathcal{S}$, *action space* $\mathcal{A}$, *transition dynamics* $\tau_a : \mathcal{S} \times \mathcal{A} \to \mathrm{Dist}(\mathcal{S})$, *reward function* $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and *discount factor* $\gamma \in (0, 1]$. The goal is to find a policy $\pi : \mathcal{S} \to \mathrm{Dist}(\mathcal{A})$ that maximizes the expected sum of discounted rewards, the *expected return*, defined as $\mathbb{E}_\pi[R_t] = \mathbb{E}_\pi[\sum_{k=0}^{T} \gamma^k r_{t+k+1}]$. *Value function* $V^\pi(s)$ denotes the expected return from s under policy π, and *action-value function* $Q^\pi(s, a)$ denotes the expected return from s after taking action a under π. Value functions are fixed points of the Bellman equation [9] and can be computed iteratively through a process referred to as *policy evaluation* [80]. Similarly, optimal value functions $V^*(s)$ and $Q^*(s, a)$ are fixed points of the Bellman optimality equation [9].

2.2 MDP Homomorphisms

MDP homomorphisms are formally defined for finite MDPs by Ravindran and Barto [69] as:

Definition 1 (MDP Homomorphism)**.** An *MDP homomorphism* $h = (f, g_s) : \mathcal{M} \to \overline{\mathcal{M}}$ is a surjective map from a finite MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, \tau_a, \gamma)$ onto an abstract finite MDP $\overline{\mathcal{M}} = (\overline{\mathcal{S}}, \overline{\mathcal{A}}, \overline{R}, \overline{\tau}_{\overline{a}}, \gamma)$ where

$f:\mathcal{S}\to\overline{\mathcal{S}}$ and $g_s:\mathcal{A}\to\overline{\mathcal{A}}$ are surjective maps onto the abstract state and action spaces:

$$\text{Invariance of reward: } \overline{R}(f(s), g_s(a)) = R(s,a) \quad \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{1}$$

$$\text{Equivariance of transitions: } \overline{\tau}_{g_s(a)}(f(s')|f(s)) = \sum_{s''\in[s']_{B_h|\mathcal{S}}} \tau_a(s''|s) \quad \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{2}$$

where B_h is the partition of $\mathcal{S}$ induced by the equivalence relation of homomorphism h, $B_h|\mathcal{S}$ is the projection of B_h onto $\mathcal{S}$, and $[s']_{B_h|\mathcal{S}}$ denotes the block of $B_h|\mathcal{S}$ to which s' belongs. Thus, when applying action a in state s, the right-hand side is the probability that the resulting state is in $[s']_{B_h|\mathcal{S}}$. The abstract MDP $\overline{\mathcal{M}}$ is in fact the quotient MDP $\mathcal{M}/B_h$ based on the homomorphism map $h:\mathcal{M}\to\mathcal{M}/B_h$. As MDP homomorphisms are sensitive with respect to changes in rewards or transitions, *approximate* MDP homomorphisms [71] allow equations (1-2) to hold approximately. The significance of MDP homomorphisms is the *optimal value equivalence* between $\mathcal{M}$ and $\overline{\mathcal{M}}$ [69]:

$$V^*(s) = \overline{V}^*(f(s)) \quad \forall s \in \mathcal{S}, \qquad Q^*(s,a) = \overline{Q}^*(f(s), g_s(a)) \quad \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{3}$$

which in turn allows for learning the optimal policy $\overline{\pi}^*$ in the abstract MDP and consequently *lifting* it to obtain the optimal policy in the actual MDP, using:

$$\pi^{\uparrow}(a|s) = \frac{\overline{\pi}(\overline{a}|f(s))}{|\{a \in g_s^{-1}(\overline{a})\}|}, \qquad \forall s \in \mathcal{S}, a \in g_s^{-1}(\overline{a})$$

where $g_s^{-1}(\overline{a})$ denotes the set of actions that have the same image $\overline{a}$ under g_s. Equivalently, the two policies must satisfy $\sum_{a\in g_s^{-1}(\overline{a})} \pi^{\uparrow}(a|s) = \overline{\pi}(\overline{a}|f(s))$ for all $s \in \mathcal{S}$ and $\overline{a} \in \overline{\mathcal{A}}$.

2.3 Bisimulation and Lax Bisimulation Metrics

Bisimulation for finite MDPs [22, 32] defines an equivalence relation on $\mathcal{S}$ where two states s_i and s_j are equivalent or *bisimilar* if $R(s_i,a) = R(s_j,a)$ and $\tau_a(C|s_i) = \tau_a(C|s_j)$ for all $a \in \mathcal{A}$ and every equivalence class C defined by the equivalence relation. The rigidity of bisimulation limits its applications. *Bisimulation metrics* [28, 27, 29] measure the equivalence as an approximation:

$$d_{\text{bisim}}\big(s_i, s_j\big) = \max_{a\in\mathcal{A}} c_r\big|R(s_i,a) - R(s_j,a)\big| + c_t K\big(\tau_a(\cdot|s_i), \tau_a(\cdot|s_j)\big), \tag{4}$$

where the first term measures reward similarity and K is the Kantorovich (Wasserstein) metric measuring the distance between the transition probabilities. However, bisimulation metrics can still be brittle as they require the behaviour to match for all actions. This may be problematic particularly in the case of continuous actions in which small changes to actions may not drastically change the outcome. Additionally, bisimulation metrics are not able to represent environment symmetries. Instead, *lax bisimulation* [83] waives the requirement on action matching in favor of extending the state equivalence relation to state-action equivalence. Taylor et al. [83] show that lax bisimulation is precisely the same relation as the MDP homomorphism and define the *lax bisimulation metric* as:

$$d_{\text{lax}}\big((s_i,a_i),(s_j,a_j)\big) = c_r\big|R(s_i,a_i) - R(s_j,a_j)\big| + c_t K\big(\tau_{a_i}(\cdot|s_i), \tau_{a_j}(\cdot|s_j)\big). \tag{5}$$

Furthermore, Taylor et al. [83] show that minimizing the lax bisimulation metric corresponds to finding approximate MDP homomorphisms and bound the value error.

3 Value Equivalence Property

To motivate the use of MDP homomorphisms for policy evaluation and consequently policy optimization, we first prove their *value equivalence property* in the finite case as the generalization of the prior result of the *optimal* value equivalence [69], stated in Equation (3). The proof is in Appendix C.1.

Theorem 1 (Value Equivalence)**.** *Let $\overline{\mathcal{M}}$ be the image of an MDP homomorphism h from a finite $\mathcal{M}$. Then any two corresponding policies $\pi^{\uparrow}$ = lift($\overline{\pi}$) have equivalent values:*

$$V^{\pi^{\uparrow}}(s) = V^{\overline{\pi}}(f(s)) \quad \forall s \in \mathcal{S}, \qquad Q^{\pi^{\uparrow}}(s,a) = Q^{\overline{\pi}}(f(s), g_s(a)) \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$$

4 Continuous MDP Homomorphisms

To concretely lay out the foundations of using MDP homomorphisms for continuous control, we extend their definition to continuous state and action spaces, and derive results analogous to the finite case. First, we define continuous MDPs and state our underlying assumptions. Importantly, the correct definitions of continuous MDPs and continuous MDP homomorphisms require care regarding measurability and differentiability of spaces, and our formulation is chosen to fit the HPG derivation; see Appendix A.2 for an overview of the tools we used from measure theory and differential geometry.

Definition 2 (Continuous MDP)**.** A *continuous Markov decision process (MDP)* is a 6-tuple:

$$\mathcal{M} = (\mathcal{S}, \Sigma, \mathcal{A}, \forall a \in \mathcal{A}\ \tau_a : \mathcal{S} \times \Sigma \to [0,1], R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}, \gamma)$$

where $\mathcal{S}$, the state space is assumed to be a Polish space, Σ is a σ-algebra on $\mathcal{S}$[1], $\mathcal{A}$, the space of *actions*, is a locally compact metric space, usually taken to be a subset of $\mathbb{R}^n$, τ_a is the transition probability kernel for each possible action a, for each fixed s, $\tau_a(\cdot|s)$ is a probability distribution on Σ while R is the reward function, and γ is the discount factor. Furthermore, for all $s \in \mathcal{S}$ and $B \in \Sigma$ the map $a \mapsto \tau_a(B|s)$ is smooth. The last assumption is required for differentiability with respect to actions a, which is needed in Section 5 for deriving the HPG theorem.

Given the continuous MDPs described above, we define continuous MDP homomorphisms. The equivariance condition on the transition dynamics, Equation (2), can no longer be expressed in terms of a discrete sum over partitions, and instead we use the σ-algebra structure on the different state spaces.

Definition 3 (Continuous MDP Homomorphism)**.** A *continuous MDP homomorphism* is a map $h = (f, g_s) : \mathcal{M} \to \overline{\mathcal{M}}$ where $f : \mathcal{S} \to \overline{\mathcal{S}}$ and for every s in $\mathcal{S}$, $g_s : \mathcal{A} \to \overline{\mathcal{A}}$ are measurable, surjective maps such that the following hold:

$$\text{Invariance of reward: } \overline{R}(f(s), g_s(a)) = R(s,a) \qquad \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{6}$$

$$\text{Equivariance of transitions: } \overline{\tau}_{g_s(a)}(\overline{B}|f(s)) = \tau_a(f^{-1}(\overline{B})|s) \qquad \forall\ s \in \mathcal{S}, a \in \mathcal{A}, \overline{B} \in \overline{\Sigma} \tag{7}$$

Note that if g_s is the identity map, the second condition reduces to $\overline{\tau}_a(\overline{B}|f(s)) = \tau_a(f^{-1}(\overline{B})|s)$ which is simply the condition for preservation of transition probabilities as used in bisimulation [22].

4.1 Optimal Value Equivalence

Assuming the conditions given in Definition 3, we prove that optimal value functions are preserved by the continuous MDP homomorphism as in the finite case.

Theorem 2 (Optimal Value Equivalence)**.** *Let $\overline{\mathcal{M}} = (\overline{\mathcal{S}}, \overline{\Sigma}, \overline{\mathcal{A}}, \overline{\tau}_{\overline{a}}, \overline{R})$ be the image of a continuous MDP homomorphism $h = (f, g_s)$ from $\mathcal{M} = (\mathcal{S}, \Sigma, \mathcal{A}, \tau_a, R)$. Then:*

$$V^*(s) = \overline{V}^*(f(s)) \quad \forall s \in \mathcal{S}, \qquad Q^*(s,a) = \overline{Q}^*(f(s), g_s(a)) \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A} \tag{8}$$

The proof, given in Appendix C.2, uses the change of variable formula of the pushforward measure of $\tau_a(\cdot|s)$ with respect to f to change the integration space from $\mathcal{S}$ to $\overline{\mathcal{S}}$.

4.2 Value Equivalence for Lifting Deterministic Policies

As in the finite case, we also require a lifting process to define $\pi^\uparrow = \mathit{lift}(\overline{\pi})$ given a policy $\overline{\pi}$ on the abstract MDP. In general, the lifted policy needs to satisfy the relation $\pi^\uparrow(g_s^{-1}(\beta)|s) = \overline{\pi}(\beta|f(s))$ for every Borel set $\beta \subseteq \overline{\mathcal{A}}$ and $s \in \mathcal{S}$. While our initial progress shows that lifted stochastic policies exist based on the disintegration theorem, the full proof and design of a computationally tractable algorithm for this process is left for future work. Therefore, here and in the subsequent sections we assume the policy is deterministic in which case the lifted policy can be simply obtained by choosing one representative for the preimage $g_s^{-1}\big(\overline{\pi}(f(s))\big)$. If we select g_s to be a bijection, the lifted policy can be uniquely defined as $\pi^\uparrow(s) = g_s^{-1}\big(\overline{\pi}(f(s))\big)$. The assumption on deterministic policies is not limiting, as in general the optimal policy of a given MDP is deterministic [10]. With this lifting definition, we state and prove the following value equivalence result:

[1] Usually the Borel algebra.

Theorem 3 (Value Equivalence for Deterministic Policies)**.** *Let $\overline{\mathcal{M}}$ be the image of a continuous MDP homomorphism $h = (f, g_s)$ from $\mathcal{M}$, then any two deterministic policies $\pi^{\uparrow} : \mathcal{S} \to \mathcal{A}$ and $\overline{\pi} : \overline{\mathcal{S}} \to \overline{\mathcal{A}}$ where $\pi^{\uparrow} = lift(\overline{\pi})$ have equivalent value functions on their domain:*

$$V^{\pi^{\uparrow}}(s) = V^{\overline{\pi}}(f(s)) \quad \forall s \in \mathcal{S}, \qquad Q^{\pi^{\uparrow}}(s,a) = Q^{\overline{\pi}}(f(s), g_s(a)) \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}$$

The proof, given in Appendix C.3, uses the change of variable formula of the pushforward measure of $\tau_a(\cdot|s)$ with respect to f to change the integration space from $\mathcal{S}$ to $\overline{\mathcal{S}}$ and assumes g_s to be bijective.

5 Homomorphic Policy Gradient

The next goal of this work is to derive a policy gradient estimator using samples obtained from the abstract MDP. Intuitively, this allows for direct incorporation of state-action abstraction as an inductive bias for policy optimization, thereby reducing the variance of actor updates and improving sample efficiency. Equipped with continuous MDP homomorphisms from Definition 3 and their value equivalence property, we now derive the *homomorphic policy gradient* (HPG) theorem.

In this section, we assume having access to an MDP homomorphism map $h = (f, g_s)$, parameterized by differentiable functions. The problem of learning such mapping from samples is addressed in Section 6. Additionally, we assume the MDP and the homomorphism map adhere to the conditions of Definition 2 and Appendix B. Similarly to prior works on policy gradients [81, 74], we define the performance measure as $J(\theta) = \mathbb{E}_{\pi}[V^{\pi}(s)]$ where the expectation is over the uncertainty in transitions, rewards, and initial states. Finally, as detailed in Section 4.2, our results are derived for deterministic policies and a bijective g_s. Notably, this choice allows us to parameterize one of the policies and to uniquely derive the other policy. In practice, we parameterize the actual policy as $\pi^{\uparrow}_{\theta}$ and obtain the abstract policy as $\overline{\pi}_{\theta} = g_s(\pi^{\uparrow}_{\theta}(s))$. First, we show the *equivalence of policy gradients*:

Theorem 4 (Equivalence of Deterministic Policy Gradients)**.** *Let $\overline{\mathcal{M}}$ be the image of a continuous MDP homomorphism h from $\mathcal{M}$, and let $\pi^{\uparrow}_{\theta} : \mathcal{S} \to \mathcal{A}$ be the lifted deterministic policy corresponding to the abstract deterministic policy $\overline{\pi}_{\theta} : \overline{\mathcal{S}} \to \overline{\mathcal{A}}$. Then for any $(s,a) \in \mathcal{S} \times \mathcal{A}$ we have:*

$$\nabla_a Q^{\pi^{\uparrow}_{\theta}}(s,a)\Big|_{a=\pi^{\uparrow}_{\theta}(s)} \nabla_{\theta} \pi^{\uparrow}_{\theta}(s) = \nabla_{\overline{a}} Q^{\overline{\pi}_{\theta}}(\overline{s}, \overline{a})\Big|_{\overline{a}=\overline{\pi}_{\theta}(\overline{s})} \nabla_{\theta} \overline{\pi}_{\theta}(\overline{s}).$$

The proof is given in Appendix C.4 and uses the chain rule and the inverse function theorem on manifolds, which in turn raises the need for g_s to be a bijection and local diffeomorphism. Theorem 4 highlights that the gradient of the abstract MDP is equivalent to that of the original, despite the underlying spaces being abstracted. This implies that performing HPG on the abstract MDP is equivalent to performing DPG on the actual MDP, allowing us to use them synergistically to update the same parameters θ, as shown in Figure 1.

While one can naively use Theorem 4 to substitute gradients of the standard DPG, theoretically this does not produce any useful result as the expectation remains estimated with respect to the stationary state distribution of the actual MDP $\mathcal{M}$ under $\pi^{\uparrow}_{\theta}(s)$. However, using properties of continuous MDP homomorphisms, we can change the integration space from $\mathcal{S}$ to $\overline{\mathcal{S}}$, and consequently estimate the policy gradient with respect to the stationary distribution of the abstract MDP $\overline{\mathcal{M}}$ under $\overline{\pi}_{\theta}(\overline{s})$:

Theorem 5 (Homomorphic Policy Gradient Theorem)**.** *Let $\overline{\mathcal{M}}$ be the image of a continuous MDP homomorphism h from $\mathcal{M}$, and let $\overline{\pi}_{\theta} : \overline{\mathcal{S}} \to \overline{\mathcal{A}}$ be a deterministic abstract policy defined on $\overline{\mathcal{M}}$. Then the gradient of the performance measure $J(\theta)$, defined on the actual MDP $\mathcal{M}$, w.r.t. θ is:*

$$\nabla_{\theta} J(\theta) = \int_{\overline{s} \in \overline{\mathcal{S}}} \rho^{\overline{\pi}_{\theta}}(\overline{s}) \nabla_{\overline{a}} Q^{\overline{\pi}_{\theta}}(\overline{s}, \overline{a})\Big|_{\overline{a}=\overline{\pi}_{\theta}(\overline{s})} \nabla_{\theta} \overline{\pi}_{\theta}(\overline{s}) d\overline{s}.$$

where $\rho^{\overline{\pi}_{\theta}}(\overline{s})$ is the discounted state distribution of $\overline{\mathcal{M}}$ following the deterministic policy $\overline{\pi}_{\theta}(\overline{s})$.

The proof is given in Appendix C.5 and applies the result of Theorem 4 and the change of variables formula of the pushforward measure on the state space. The significance of Theorem 5, which forms the basis of our proposed homomorphic actor-critic algorithm, is twofold. First, we can get another estimate for the policy gradient based on the approximate MDP homomorphic image in addition to DPG. Although the two policy gradient estimates are not statistically independent from one another

as they are tied through the homomorphism map, HPG will potentially have less variance at the expense of some bias due to the approximation of the MDP homomorphism.

Second, since the minimal image of an MDP is the MDP homomorphic image [69], the abstract critic $Q^{\overline{\pi}_\theta}$ is trained on a simplified problem. In other words, each abstract state-action pair $(\overline{s}, \overline{a})$ used to train $Q^{\overline{\pi}_\theta}$ represents all (s, a) pairs that are equivalent under the MDP homomorphism relation, thus improving sample efficiency. However, the amount of complexity reduction is dependent on the approximate symmetries of the environment, as also supported by our empirical results.

6 Homomorphic Actor-Critic Algorithms

We propose a deep actor-critic algorithm based on HPG by adapting DDPG [54]. We refer to our method as *Deep Homomorphic Policy Gradient* (DHPG). Although HPG is applicable to any other deterministic actor-critic, we chose DDPG as its simplicity compared to modern choices [8, 45] allows for a better study on the impact of MDP homomorphisms. Notably, a strong advantage of DHPG is that it is readily applicable to pixel observations without the need for extra mechanisms such as image reconstruction [31, 36, 97, 38], as the notion of MDP homomorphism provides a natural inductive bias for learning representations that preserve values and optimal values.

Since DHPG is learning the MDP homomorphism map h online and concurrently with the policy, using the actual MDP for training the critic, specifically at the early stages of training, is helpful. Therefore, we utilize a separate critic for each MDP $\mathcal{M}$ and $\overline{\mathcal{M}}$. Ultimately, critics are used to update a single set of parameters, as shown in Figure 1; see Appendix D.5 for the ablation study on this.

Thus, the components of DHPG are: actual critic $Q_\psi(s, a)$, abstract critic $\overline{Q}_{\overline{\psi}}(\overline{s}, \overline{a})$, deterministic actor $a{=}\pi_\theta(s)$, homomorphism map $h_{\phi,\eta}{=}(f_\phi(s), g_\eta(s, a))$, reward predictor $\overline{R}_\rho(\overline{s})$, and probabilistic transition model $\overline{\tau}_\nu(\overline{s}'|\overline{s}, \overline{a})$ which outputs a Gaussian distribution. We use target networks and a vanilla replay buffer [60, 54]. As discussed in Section 5, an abstract actor is obtained as $\overline{a}{=}g_s(\pi_\theta(s))$. In case of pixel observations, a single image encoder E_μ is shared among all components.

Training the Policy and Critic. Actual and abstract critics are trained using n-step TD error for a faster reward propagation [8]. The loss function for each critic is therefore defined as the expectation of the n-step Bellman error estimated over transitions samples from the replay buffer $\mathcal{B}$:

$$\mathcal{L}_{\text{actual critic}}(\psi) = \mathbb{E}_{(s,a,s',r)\sim\mathcal{B}}\Big[\Big(R_t^{(n)} + \gamma^n Q_{\psi'}(s_{t+n}, a_{t+n}) - Q_\psi(s_t, a_t)\Big)^2\Big] \tag{9}$$

$$\mathcal{L}_{\text{abstract critic}}(\overline{\psi}, \phi, \eta) = \mathbb{E}_{(s,a,s',r)\sim\mathcal{B}}\Big[\Big(R_t^{(n)} + \gamma^n \overline{Q}_{\overline{\psi}'}(\overline{s}_{t+n}, \overline{a}_{t+n}) - \overline{Q}_{\overline{\psi}}(\overline{s}_t, \overline{a}_t)\Big)^2\Big], \tag{10}$$

where $\overline{s}_t{=}f_\phi(s_t)$ and $\overline{a}_t{=}g_\eta(s_t, a_t)$ are computed using the learned MDP homomorphism, ψ' and $\overline{\psi}'$ denote parameters of target networks, and $R_t^{(n)}{=}\sum_{i=0}^{n-1}\gamma^i r_{t+i}$ is the n-step return. Consequently, we train the policy using DPG [74] and HPG from Theorem 5 by backpropagating the following loss:

$$\mathcal{L}_{\text{actor}}(\theta) \approx -\mathbb{E}_{s\sim\mathcal{B}}\Big[Q_\psi\big(s, \pi_\theta(s)\big) + \overline{Q}_{\overline{\psi}}\big(f_\phi(s), g_\eta\big(s, \pi_\theta(s)\big)\big)\Big]. \tag{11}$$

Here, the two gradients are added together and a single policy update is conducted; see Appendix D.5 for the ablation study on other combinations of HPG and DPG. Finally, we utilize target policy smoothing and delayed actor updates [30]. The pseudo-code of DHPG is presented in Appendix E.1.

(a) Sample efficiency. (b) Performance profiles. (c) Learning curves. (d) Learning curves.

Figure 2: Results of DM Control tasks with **state observations** obtained on 10 seeds. RLiable metrics are aggregated over 17 tasks. **(a)** RLiable IQM scores as a function of number of steps for comparing sample efficiency, **(b)** RLiable performance profiles at 500k steps, **(c)**-**(d)** examples of learning curves. Full results are in Appendix D.1. Shaded regions represent 95% confidence intervals.

Learning Continuous MDP Homomorphisms. We now address the problem of learning continuous MDP homomorphisms. While few methods have been proposed for learning finite MDP homomorphisms [87, 89], these are not readily extendable to continuous actions. In this work, we use the lax bisimulation metric [83], Equation (5), to propose a loss function that encodes lax bisimilar states closer together in the abstract space. The lax bisimulation metric is applicable to continuous actions and as a (pseudo-)metric, it can naturally represent approximate MDP homomorphisms.

Following the same intuition as prior works on bisimulations [98], we define our proposed lax bisimulation loss over pairs of transition tuples sampled from the replay buffer. We permute samples to compute their pairwise distance in the abstract space and their pairwise lax bisimilarity distance. Consequently, we minimize the distance between these two terms:

$$\mathcal{L}_{\text{lax}}(\phi,\eta) = \mathbb{E}_{\mathcal{B}}\Big[\|f_\phi(s_i) - f_\phi(s_j)\|_1 - \|r_i - r_j\|_1 - \alpha W_2\big(\overline{\tau}_\nu(\cdot|f_\phi(s_i), g_\eta(s_i,a_i)), \overline{\tau}_\nu(\cdot|f_\phi(s_j), g_\eta(s_j,a_j))\big)\Big] \tag{12}$$

Similarly to Zhang et al. [98], we replaced the Kantorovich (W_1) metric in Equation (5) with the W_2 metric as there is an explicit formula for it for Gaussian distributions. Finally, we apply the conditions of a continuous MDP homomorphism map from Definition 3 via the loss function of:

$$\mathcal{L}_{\text{h}}(\phi,\eta,\nu,\rho) = \mathbb{E}_{(s_i,a_i,s_i',r_i)\sim\mathcal{B}}\Big[\big(f_\phi(s_i') - \overline{s}_i'\big)^2 + \big(r_i - \overline{R}_\rho(f_\phi(s_i))\big)^2\Big], \tag{13}$$

where $\overline{s}_i' \sim \overline{\tau}_\nu(\cdot|f_\phi(s_i), g_\eta(s_i,a_i))$. The final loss function is obtained as $\mathcal{L}_{\text{lax}}(\phi,\eta) + \mathcal{L}_{\text{h}}(\phi,\eta,\rho,\nu)$.

7 Experiments

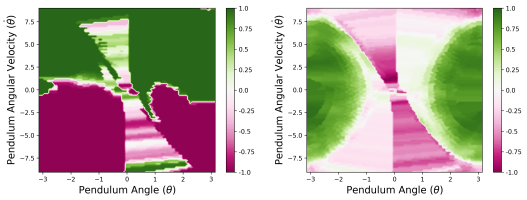
In our experiments, we aim to answer the following key questions:

1. Does the homomorphic policy gradient improve policy optimization?
2. What are the qualitative properties of the learned representations and the abstract MDP?
3. Can DHPG learn and recover the minimal MDP image from raw pixel observations?

We evaluate DHPG on continuous control tasks from DM Control on state and pixel observations. Importantly, to reliably evaluate our algorithm against the baselines and to correctly capture the distribution of results, we follow the best practices proposed by Agarwal et al. [5] and report the interquartile mean (IQM) and performance profiles aggregated on all tasks over 10 random seeds. While our baseline results are obtained using the official code, when possible [2], some of the results may differ from the originally reported ones due to the difference in the seed numbers and our goal to present a faithful representation of the true performance distribution [5].

7.1 State Observations

We compare DHPG on state observations against three commonly-used off-policy model-free algorithms: DDPG [54], TD3 [30], and SAC [37]. All methods use n-step returns, and share the same hyperparameters presented in Appendix E.2. For a fair comparison with DDPG and a better study of the impact of HPG, we have improved our DDPG by adding delayed policy updates and target policy [30]. Thus, the only difference between our DDPG and TD3 is the clipped double Q-learning present in TD3, which appears to be hurting the performance in some tasks of DMC as also observed in [63, 53].

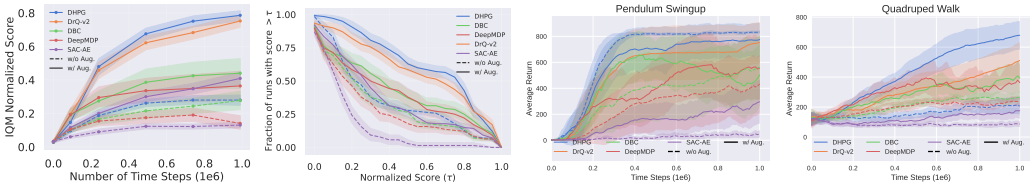


(a) Actual optimal policy. (b) Abstract optimal policy.

Figure 3: Contours of actual and abstract optimal actions over the state space of the pendulum-swingup task. Colors represent action values, and states are $s = (\theta, \dot{\theta})$. **(a)** Actual optimal policy; contours of optimal actions $a^* = \pi^{\uparrow *}(s)$. **(b)** Abstract optimal policy; contours of abstract optimal actions $\overline{a}^* = g_s(a^*) = \overline{\pi}^*(\overline{s})$. The relation $g_{s_1}(a_1) = g_{s_2}(a_2)$ holds for equivalent state-action pairs, and the abstract optimal policy is symmetric.

DHPG outperforms or matches other algorithms on state observations and has a better sample efficiency. Results are presented in Figure 2, and *full results are in Appendix D.1*. Expectedly, performance gains are larger on tasks with symmetries, as DHPG is able to learn a compressed abstract MDP.

[2] We use the official implementations of DrQv2, DBC, and SAC-AE, while we re-implement DeepMDP due to the unavailability of the official code. See Appendix E.3 for full details.



(a) Sample efficiency. (b) Performance profiles. (c) Learning curves. (d) Learning curves.

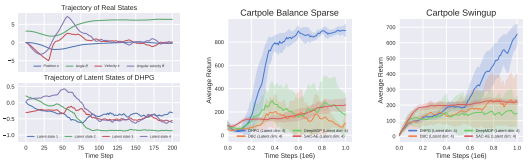
Figure 4: Results of DM Control tasks with **pixel observations** obtained on 10 seeds. RLiable metrics are aggregated over 14 tasks. **(a)** RLiable IQM scores as a function of number of steps for comparing sample efficiency, **(b)** RLiable performance profiles at 500k steps, **(c)-(d)** examples of learning curves. Full results are in Appendix D.2. Shaded regions represent 95% confidence intervals.

The learned mapping h=(f, g_s) demonstrates properties of an MDP homomorphism. We use the pendulum swingup task to visualize its learned MDP homomorphism, as its symmetries are perfectly intelligible. Two state-action pairs $(s_1 = (\theta_1, \dot{\theta}_1), a_1)$ and $(s_2 = (\theta_2, \dot{\theta}_2), a_2)$ are equivalent if $a_1 = -a_2$, $\theta_1 = -\theta_2$, and $\dot{\theta}_1 = -\dot{\theta}_2$. Therefore, the learned action representations are expected to reflect this by setting $g_{s_1}(a_1) = g_{s_2}(a_2)$. Figure 3a shows contours of optimal actions over $\mathcal{S}$, while Figure 3b shows action representations $\overline{a} = g_s(a)$ of optimal actions over $\mathcal{S}$. Clearly, abstract actions adhere to the aforementioned relation for equivalent state-action pairs, indicating $g_s(a)$ is in fact representing the action encoder of an MDP homomorphism mapping.

7.2 Pixel Observations

We compare the effectiveness of DHPG on pixel observations against DBC [98], DeepMDP [31], SAC-AE [97], and state-of-the-art performing DrQ-v2 [95]. All methods use n-step returns, share the same hyperparameters in Appendix E.2 and all hyperparameters are adapted from DrQ-v2 *without any further tuning*. Importantly, for a fair comparison with DrQ-v2 which uses image augmentation, we present two variations of DHPG and other baselines, *with and without image augmentation*.

DHPG outperforms or matches other algorithms on pixel observations, demonstrating its effectiveness in representation learning. Results are presented in Figure 4 and *full results are in Appendix D.2*. Interestingly, DHPG without image augmentation outperforms DrQ-v2 on domains with easily learnable MDP homomorphism maps, such as cartpole and pendulum, showing its power of representation learning.



(a) Trajectories. (b) Curves. (c) Curves.

Figure 5: Effectiveness of DHPG in recovering the minimal MDP from **pixels**. All methods are limited to a 4-dimensional latent space which is equal to the dimensions of the real state space of cartpole. **(a)** Trajectories of real states obtained from Mujoco and trajectories of latent states of DHPG. **(b, c)** Learning curves averaged on 10 seeds.

DHPG can learn and recover a low-dimensional MDP image. A key strength of MDP homomorphisms is their ability to represent the minimal MDP image [69], which is particularly important when learning from pixel observations. To demonstrate this ability, we have limited the latent space dimensions to the dimension of the real system and compared DHPG (without image augmentation) with baselines in Figure 5. While other methods are not able to learn the tasks, DHPG can successfully learn the policy and the minimal low-dimensional latent space. Surprisingly, trajectories of the latent states resemble that of the real states as shown in Figure 5a.

The abstract MDP demonstrates properties of an MDP homomorphic image. To qualitatively demonstrate the significance of learning joint state-action representations, Figure 6 shows visualizations of latent states for quadruped-walk, a task with symmetries around movements of its four legs. Interestingly, while the latent space of DHPG (Figure 6a) shows distinct states for each leg, abstract state encoder f_ϕ has mapped corresponding legs (e.g., left forward leg and right backward leg) to the same abstract latent state (Figure 6b) as they are some homomorphic image of one another. Clearly, DBC and DrQ-v2 are not able to achieve this.

The learned representations and the MDP homomorphism map transfer to new tasks within the same domain. Importantly, one consideration with representation learning methods relying on rewards is the transferability of the learned representations to a new reward setting within the same domain. To ensure that our method does not hinder such transfer, we have carried out experiments in which the actor, critics, and the learned MDP homomorphism map are transferred to another task from the same domain. Results, given in Appendix D.3 show that our method has not compromised transfer abilities.

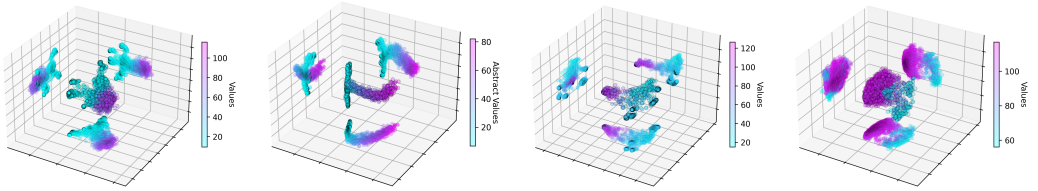
Accounting for the larger network capacity of DHPG compared to the baselines. Since our DHPG algorithm contains additional networks, such as the parameterized MDP homomorphism map and the abstract critic, it may have a higher network capacity compared to the baselines. To control for the effect of the network capacity and for a fair evaluation, we compare DHPG with higher-capacity variants of DBC and DrQ-v2 that have a larger critic networks, selected such that the total number of parameters are considerably more than that of DHPG. Results are presented in Figure 7, while *full results and a detailed description of the total number of parameters are in Appendix D.7*. As suggested by the results, DHPG outperforms or matches the performance of the higher-capacity baselines, demonstrating the improved performance is rather due to the use of the abstract MDP homomorphic image for representation learning and performing HPG updates.

Additional Experiments. We study the value equivalence property as a measure for the quality of the learned MDP homomorphisms in Appendix D.4, and we present ablation studies on DHPG variants, and the impact of n-step return on our method in Appendices D.5 and D.6, respectively.

8 Related Work

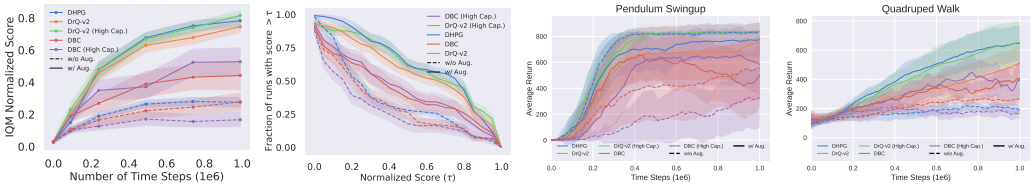
State Abstraction. Bisimulation [59, 50] is a notion of behavioral equivalence between systems. It was extended to continuous state spaces by Blute et al. [13, 22] and extended to MDPs by Givan et al. [32]. Bisimulation metrics [23, 28, 27, 29] define a pseudometric to quantify the degree of behavioural similarity. Recently, Zhang et al. [98] defined a loss function for learning representations via bisimilarity of latent states, and Kemertas et al. [48] have further improved its robustness. Castro [16] has proposed a method to approximate the bisimulation metric for deterministic MDPs with continuous states but discrete actions. van der Pol et al. [87] have defined a contrastive loss based on MDP homomorphisms for learning an abstract MDP for planning, however, their method is only applicable to finite MDPs. Another approach is to directly embed the MDP homomorphic relation in the network architecture [89, 88]. Other recently proposed methods seek to learn representations that preserve values [35, 34] or policies [4], or via a sampling-based similarity metric [17]. Finally, state abstractions can in principle help improve transferring of policies [1, 18, 76, 77, 67], or learning temporally extended actions [19, 94, 93, 82].

Action Abstraction. Action representations are often studied in the context of large discrete action spaces [72] as a form of a look-up embedding that is known *a-priori* [24], factored representations [73], or policy decomposition [20]. Action representations can also be learned from expert demonstrations [84]. More related to our work is dynamics-aware embeddings [91] where a combined state-action



(a) Latent states, DHPG. (b) Abstract states, DHPG. (c) Latent states, DBC. (d) Latent States, DrQ-v2.

Figure 6: PCA projection of learned representations for quadruped-walk with **pixel observations**. **(a)** Latent states $s = E_\mu(o)$, **(b)** abstract latent states $\overline{s} = f_\phi(E_\mu(o))$ for DHPG, **(c)** latent states $s = E_\mu(o)$ for DBC, and **(d)** DrQ-v2. Color of each point denotes its value learned by $Q(s,a)$ or $\overline{Q}(\overline{s},\overline{a})$. Points are also projected onto each main plane. The homomorphism map of DHPG has mapped the latent states of corresponding legs (e.g., left forward leg and right backward leg) **(a)** on to the same abstract latent states **(b)**, indicating a clear structure in $\overline{\mathcal{S}}$.



(a) Sample efficiency. (b) Performance profiles. (c) Learning curves. (d) Learning curves.

Figure 7: Results of DM Control tasks with **pixel observations** for **higher-capacity variants** of DBC and DrQ-v2 obtained on 10 seeds. RLiable metrics are aggregated over 14 tasks. **(a)** RLiable IQM scores as a function of number of steps for comparing sample efficiency, **(b)** RLiable performance profiles at 500k steps, **(c)**-**(d)** examples of learning curves. Full results are in Appendix D.7. Shaded regions represent 95% confidence intervals.

embedding for continuous control is learned. In contrast, we use the notion of homomorphisms to learn the state-dependent action representations, while preserving values. Action representations can be combined with temporal abstraction [82] for discovering extended actions [70, 3, 18, 19].

State Representation Learning. Extant methods for learning the underlying state space from raw observations often use latent models [31, 38, 39, 36, 11], auxiliary prediction tasks [46, 56, 57], physics-inspired inductive biases [47, 21, 33], unsupervised learning [44, 55], or self-supervised learning [6, 75, 40, 41, 26]. From another point of view, representation learning can be effectively decoupled from the RL problem [25, 79]. Symmetries of the environment can also be used for representation learning [61, 58, 64, 90, 42, 43, 66, 15]. In fact, MDP homomorphisms are specializations of such approaches for RL. A key distinguishing factor of MDP homomorphisms is their ability to take actions into account for representation learning in the same premises as Thomas et al. [85]. Recently, simple image augmentation methods have shown significant improvements in RL performance [96, 51]. Since these approaches are in general orthogonal to state abstractions, they can be combined together.

9 Conclusion

In this paper, we developed the novel theory of continuous MDP homomorphisms using measure theory, and we rigorously proved their value and optimal value equivalence properties. We derived the homomorphic PG in order to directly use a joint state-action abstraction for policy optimization. Importantly, we rigorously proved that applying our homomorphic PG on the abstract MDP is equivalent to applying the standard DPG on the actual MDP. Based on our novel theoretical results, we developed a deep actor-critic algorithm that can simultaneously learn the policy and the MDP homomorphism map using the lax bisimulation metric. Our algorithm improves upon strong baselines in both learning from state and pixel observations. The visualization of the latent space demonstrates the strong potential of MDP homomorphisms in learning structured representations that can preserve value functions. We believe that our work will open-up future possibilities for the application of MDP homomorphisms in challenging continuous control problems.

Acknowledgements

SRS is supported by an NSERC CGS-D scholarship. RZ was supported by an NSERC CGS-M scholarship at the time this work was completed. We would like to thank Juan Camilo Gamboa Higuera, Harley Wiltzer, and Scott Fujimoto for insightful discussions. The computing resources for this research were provided by Calcul Quebec and the Digital Research Alliance of Canada.

References

[1] David Abel, Dilip Arumugam, Kavosh Asadi, Yuu Jinnai, Michael L Littman, and Lawson LS Wong. State abstraction as compression in apprenticeship learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3134–3142, 2019.

[2] David Abel, David Hershkowitz, and Michael Littman. Near optimal behavior via approximate state abstraction. In *International Conference on Machine Learning*, pages 2915–2923. PMLR, 2016.

[3] David Abel, Nate Umbanhowar, Khimya Khetarpal, Dilip Arumugam, Doina Precup, and Michael Littman. Value preserving state-action abstractions. In *International Conference on Artificial Intelligence and Statistics*, pages 1639–1650. PMLR, 2020.

[4] 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*, 2020.

[5] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34, 2021.

[6] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised state representation learning in Atari. *Advances in Neural Information Processing Systems*, 32:8769–8782, 2019.

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[8] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

[9] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[10] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 1. Athena scientific, 2012.

[11] O Biza, R Platt, JW van de Meent, and L Wong. Learning discrete state abstractions with deep variational inference. *Advances in Approximate Bayesian Inference*, 2021.

[12] Ondrej Biza and Robert Platt. Online abstraction with mdp homomorphisms for deep learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019.

[13] Richard Blute, Josée Desharnais, Abbas Edalat, and Prakash Panangaden. Bisimulation for labelled markov processes. In *Proceedings of Twelfth Annual IEEE Symposium on Logic in Computer Science*, pages 149–158. IEEE, 1997.

[14] Vladimir Igorevich Bogachev and Maria Aparecida Soares Ruas. *Measure theory*, volume 1. Springer, 2007.

[15] Hugo Caselles-Dupré, Michael Garcia Ortiz, and David Filliat. Symmetry-based disentangled representation learning requires interaction with environments. *Advances in Neural Information Processing Systems*, 32:4606–4615, 2019.

[16] Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10069–10076, 2020.

[17] Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. Mico: Improved representations via sampling-based state similarity for Markov decision processes. *Advances in Neural Information Processing Systems*, 34, 2021.

[18] Pablo Samuel Castro and Doina Precup. Using bisimulation for policy transfer in MDPs. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[19] Pablo Samuel Castro and Doina Precup. Automatic construction of temporally extended actions for MDPs using bisimulation metrics. In *European Workshop on Reinforcement Learning*, pages 140–152. Springer, 2011.

[20] Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip Thomas. Learning action representations for reinforcement learning. In *International Conference on Machine Learning*, pages 941–950. PMLR, 2019.

[21] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

[22] J. Desharnais, A. Edalat, and P. Panangaden. Bisimulation for labeled Markov processes. *Information and Computation*, 179(2):163–193, Dec 2002.

[23] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labeled Markov systems. In *Proceedings of CONCUR99*, number 1664 in Lecture Notes in Computer Science. Springer-Verlag, 1999.

[24] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

[25] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

[26] Linxi Fan, Guanzhi Wang, De-An Huang, Zhiding Yu, Li Fei-Fei, Yuke Zhu, and Animashree Anandkumar. Secant: Self-expert cloning for zero-shot generalization of visual policies. In *International Conference on Machine Learning*, pages 3088–3099. PMLR, 2021.

[27] Norm Ferns, Pablo Samuel Castro, Doina Precup, and Prakash Panangaden. Methods for computing state similarity in Markov decision processes. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pages 174–181, 2006.

[28] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for Markov decision processes with infinite state spaces. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 201–208, 2005.

[29] Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous Markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714, 2011.

[30] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[31] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pages 2170–2179. PMLR, 2019.

[32] Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.

[33] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in neural information processing systems*, 32, 2019.

[34] Christopher Grimm, André Barreto, Gregory Farquhar, David Silver, and Satinder Singh. Proper value equivalence. *arXiv preprint arXiv:2106.10316*, 2021.

[35] Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. *arXiv preprint arXiv:2011.03506*, 2020.

[36] David Ha and Jürgen Schmidhuber. World models. *arXiv e-prints*, pages arXiv–1803, 2018.

[37] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[38] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2019.

[39] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[40] 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*, 2020.

[41] Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13611–13617. IEEE, 2021.

[42] Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.

[43] Irina Higgins, Peter Wirnsberger, Andrew Jaegle, and Aleksandar Botev. Symetric: Measuring the quality of learnt hamiltonian dynamics inferred from vision. *Advances in Neural Information Processing Systems*, 34, 2021.

[44] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2018.

[45] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. In *International Conference on Learning Representations*, 2018.

[46] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[47] Rico Jonschkowski and Oliver Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39(3):407–428, 2015.

[48] Mete Kemertas and Tristan Aumentado-Armstrong. Towards robust bisimulation metric learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[49] Serge Lang. *Differential and Riemannian manifolds*, volume 160. Springer Science & Business Media, 2012.

[50] Kim G Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and computation*, 94(1):1–28, 1991.

[51] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. *arXiv preprint arXiv:1910.05396*, 2019.

[52] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for MDPs. *ISAIM*, 4:5, 2006.

[53] Qing Li. Continuous control benchmark of DeepMind control suite and MuJoCo. `https://github.com/LQNew/Continuous_Control_Benchmark`, 2021.

[54] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[55] Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pages 6736–6747. PMLR, 2021.

[56] Shikun Liu, Andrew Davison, and Edward Johns. Self-supervised generalisation with meta auxiliary learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[57] Clare Lyle, Mark Rowland, Georg Ostrovski, and Will Dabney. On the effect of auxiliary tasks on representation dynamics. In *International Conference on Artificial Intelligence and Statistics*, pages 1–9. PMLR, 2021.

[58] Anuj Mahajan and Theja Tulabandhula. Symmetry learning for function approximation in reinforcement learning. *arXiv preprint arXiv:1706.02999*, 2017.

[59] Robin Milner. *Communication and concurrency*, volume 84. Prentice hall Englewood Cliffs, 1989.

[60] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[61] Arnab Kumar Mondal, Vineet Jain, Kaleem Siddiqi, and Siamak Ravanbakhsh. Eqr: Equivariant representations for data-efficient reinforcement learning. In *International Conference on Machine Learning*, pages 15908–15926. PMLR, 2022.

[62] Shravan Matthur Narayanamurthy and Balaraman Ravindran. On the hardness of finding symmetries in markov decision processes. In *Proceedings of the 25th international conference on Machine learning*, pages 688–695, 2008.

[63] Fabio Pardo. Tonic: A deep reinforcement learning library for fast prototyping and benchmarking. *arXiv preprint arXiv:2011.07537*, 2020.

[64] Jung Yeon Park, Ondrej Biza, Linfeng Zhao, Jan Willem van de Meent, and Robin Walters. Learning symmetric embeddings for equivariant world models. *arXiv preprint arXiv:2204.11371*, 2022.

[65] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[66] Robin Quessard, Thomas D Barrett, and William R Clements. Learning group structure and disentangled representations of dynamical environments. *arXiv preprint arXiv:2002.06991*, 2020.

[67] Srividhya Rajendran and Manfred Huber. Learning to generalize and reuse skills using approximate partial policy homomorphisms. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 2239–2244. IEEE, 2009.

[68] Balaraman Ravindran. *An algebraic approach to abstraction in reinforcement learning*. University of Massachusetts Amherst, 2004.

[69] Balaraman Ravindran and Andrew G Barto. Symmetries and model minimization in markov decision processes, 2001.

[70] Balaraman Ravindran and Andrew G Barto. Relativized options: Choosing the right transformation. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 608–615, 2003.

[71] Balaraman Ravindran and Andrew G Barto. Approximate homomorphisms: A framework for non-exact minimization in Markov Decision Processes, 2004.

[72] Brian Sallans and Geoffrey E Hinton. Reinforcement learning with factored states and actions. *The Journal of Machine Learning Research*, 5:1063–1088, 2004.

[73] Sahil Sharma, Aravind Suresh, Rahul Ramesh, and Balaraman Ravindran. Learning to factor policies and action-value functions: Factored action space representations for deep reinforcement learning. *arXiv preprint arXiv:1705.07269*, 2017.

[74] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[75] Samarth Sinha, Ajay Mandlekar, and Animesh Garg. S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *5th Annual Conference on Robot Learning*, 2021.

[76] Vishal Soni and Satinder Singh. Using homomorphisms to transfer options across continuous reinforcement learning domains. In *AAAI*, volume 6, pages 494–499, 2006.

[77] Jonathan Sorg and Satinder Singh. Transfer via soft homomorphisms. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 741–748, 2009.

[78] Michael Spivak. *Calculus on manifolds: a modern approach to classical theorems of advanced calculus*. CRC press, 2018.

[79] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pages 9870–9879. PMLR, 2021.

[80] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[81] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[82] Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[83] Jonathan Taylor, Doina Precup, and Prakash Panagaden. Bounding performance loss in approximate mdp homomorphisms. *Advances in Neural Information Processing Systems*, 21:1649–1656, 2008.

[84] Guy Tennenholtz and Shie Mannor. The natural language of actions. In *International Conference on Machine Learning*, pages 6196–6205. PMLR, 2019.

[85] Valentin Thomas, Jules Pondard, Emmanuel Bengio, Marc Sarfati, Philippe Beaudoin, Marie-Jean Meurs, Joelle Pineau, Doina Precup, and Yoshua Bengio. Independently controllable factors. *arXiv preprint arXiv:1708.01289*, 2017.

[86] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[87] Elise van der Pol, Thomas Kipf, Frans A Oliehoek, and Max Welling. Plannable approximations to mdp homomorphisms: Equivariance under actions. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1431–1439, 2020.

[88] Elise van der Pol, Herke van Hoof, Frans A Oliehoek, and Max Welling. Multi-agent MDP homomorphic networks. *arXiv preprint arXiv:2110.04495*, 2021.

[89] Elise van der Pol, Daniel Worrall, Herke van Hoof, Frans Oliehoek, and Max Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[90] Dian Wang, Robin Walters, and Robert Platt. So(2)-equivariant reinforcement learning. In *International Conference on Learning Representations*, 2021.

[91] William Whitney, Rajat Agarwal, Kyunghyun Cho, and Abhinav Gupta. Dynamics-aware embeddings. In *International Conference on Learning Representations*, 2019.

[92] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[93] Alicia P Wolfe and Andrew G Barto. Decision tree methods for finding reusable mdp homomorphisms. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 21, page 530. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[94] Alicia Peregrin Wolfe and Andrew G Barto. Defining object types and options using mdp homomorphisms. In *Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*. Citeseer, 2006.

[95] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[96] 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*, 2020.

[97] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10674–10681, 2021.

[98] 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*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] All the theoretical claims are substantiated in Sections 3, 4, and 5. All the empirical claims are substantiated in Section 7 and Appendix D.
   (b) Did you describe the limitations of your work? [Yes] Theoretical limitations and assumptions are discussed in Section 4, and empirical limitations are discussed in Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work is foundational research on reinforcement learning and state abstraction, hence we do not foresee any substantive societal and ethical implications.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] The full set of assumptions are detailed in Section 4 and Appendix B.
   (b) Did you include complete proofs of all theoretical results? [Yes] The proofs of all theoretical results are given in Appendix C.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Our code, including the instructions, are submitted in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Hyperparameters, implementation, and training details are given in Appendix E
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All experiments are obtained on 10 seeds, and we report confidence intervals, interquartile mean, median, mean, and performance profiles as suggested by Agarwal et al. [5].
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We describe our hardware setup in Appendix E, but we do not include the amount of compute used.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] The creators are cited appropriately and information of each asset is included in Appendix E.3.
   (b) Did you mention the license of the assets? [N/A] All assets are open-source and under permissive open-source licenses.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include the code of our algorithm in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We are using open-source simulators and reinforcement learning environments.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We are using open-source simulators and reinforcement learning environments.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Additional Background

A.1 Background on the Policy Gradient Theorem

RL algorithms can be broadly divided into *value-based* and *policy gradient* (PG) methods. While value-based methods select actions via a greedy maximization step based on the learned action-values, PG methods directly optimize a parameterized policy π_θ based on the performance gradient $\nabla_\theta J(\theta)$. Thus, unlike value-based methods, PG algorithms inherit the strong, albeit local, convergence guarantees of the gradient descent and are naturally extendable to continuous actions. The fundamental theorem underlying PG methods is the *policy gradient theorem* [81]:

$$\nabla_\theta J(\pi_\theta) = \int_{s\in\mathcal{S}} \rho^{\pi_\theta}(s) \int_{a\in\mathcal{A}} \nabla_\theta \pi_\theta(a|s) Q^{\pi_\theta}(s,a) \tag{14}$$

where $\rho^{\pi_\theta}(s) = \lim_{t\to\infty} \gamma^t P(s_t = s|s_0, a_{0:t} \sim \pi_\theta)$ is the discounted stationary distribution of states under π_θ which is assumed to exist and to be independent of the initial state distribution (ergodicity assumption). The significance of the PG theorem is that the effect of policy changes on the state distribution does not appear in its expression, allowing for a sample-based estimate of the gradient [92].

The deterministic policy gradient (DPG) is derived for deterministic policies by Silver et al. [74] as:

$$\nabla_\theta J(\pi_\theta) = \int_{s\in\mathcal{S}} \rho^{\pi_\theta}(s) \nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s,a)\big|_{a=\pi_\theta(s)} \tag{15}$$

Since DPG does not need to integrate over the action space, it is often more sample-efficient than the stochastic policy gradient [74]. However, a noise needs to be manually injected during exploration as the deterministic policy does not have any inherent means of exploration. Finally, it is worth noting that due to the differentiation of the value function with respect to a, DPG is only applicable to continuous actions.

A.2 Mathematical Tools

Various mathematical concepts from measure theory and differential geometry are presented in this section. We only explicitly introduce concepts which are directly mentioned or relevant to the proofs presented in section C; for a more comprehensive overview, we direct the reader to textbooks such as [14, 49, 78].

Definition 4 (σ-algebra)**.** Given a set X, a σ-algebra on X is a family Σ of subsets of X such that 1) $X \in \Sigma$, 2) $A \in \Sigma$ implies $A^c \in \Sigma$ (closure under complements), and 3) if $(A_i)_{i\in\mathbb{N}}$ satisfies $A_i \in \Sigma$ for all $i \in \mathbb{N}$, then $\cup_{i\in\mathbb{N}} A_i \in \Sigma$ (closure under countable union). The tuple (X, Σ) is a measurable space.

The σ-algebra of a space specifies the sets in which a measure is defined; in probability theory— and in our use case— a σ-algebra represents a collection of events which can be assigned probabilities.

Definition 5 (Pushforward measure)**.** Let (X_1, Σ_1) and (X_2, Σ_2) be two measurable spaces, $f : X_1 \to X_2$ a measurable map and $\mu : \Sigma_1 \to [0, \infty]$ a measure on X_1. Then the pushforward measure of μ with respect to f, denoted $f_*(\mu) : \Sigma_2 \to [0, \infty]$ is defined as:

$$(f_*(\mu))(B) = \mu(f^{-1}(B)) \ \forall\ B \in \Sigma_2.$$

Theorem 6 (Change of variables)**.** *A measurable function g on X_2 is integrable with respect to $f_*(\mu)$ if and only if the function $g \circ f$ is integrable with respect to μ, in which case the integrals are equal:*

$$\int_{X_2} g d(f_*(\mu)) = \int_{X_1} g \circ f d\mu.$$

Definition 6 (Local diffeomorphism)**.** Let M and N be differentiable manifolds. A function $f : M \to N$ is a *local diffeomorphism*, if for each point $x \in M$ there exists an open set U containing x such that $f(U)$ is open in N and $f|_U : U \to f(U)$ is a diffeomorphism.

Theorem 7 (Inverse function theorem for manifolds)**.** *If $f : M \to N$ is a smooth map whose differential $df_x : T_x M \to T_{f(x)} N$ is an isomorphism at a point $x \in M$. Then f is a local diffeomorphism at x.*

Theorem 8 (Chain rule for manifolds)**.** *If $f : M \to N$ and $g : N \to O$ are smooth maps of manifolds, then:*

$$d(g \circ f)_x = dg_{f(x)} \circ df_x.$$

B Assumptions and Conditions

The derivation of our homomorphic policy gradient theorem is for continuous state and action spaces. Therefore, we have assumed the following regularity conditions on the actual MDP $\mathcal{M}$ and its MDP homomorphic image $\overline{\mathcal{M}}$ under the MDP homomorphism map h. The conditions are largely based on the regularity conditions of the deterministic policy gradient theorem [74]:

Regularity conditions 1: $\tau_a(s'|s)$, $\nabla_a \tau_a(s'|s)$, $\overline{\tau}_{\overline{a}}(\overline{s}'|\overline{s})$, $\nabla_{\overline{a}} \overline{\tau}_{\overline{a}}(\overline{s}'|\overline{s})$, $R(s,a), \nabla_a R(s,a)$, $\overline{R}(\overline{s},\overline{a}), \nabla_{\overline{a}} \overline{R}(\overline{s},\overline{a})$, $\pi_\theta^{\uparrow}(s), \nabla_\theta \pi_\theta^{\uparrow}(s), \overline{\pi}_\theta(\overline{s}), \nabla_\theta \overline{\pi}_\theta(\overline{s}), p_1(s)$, and $\overline{p}_1(\overline{s})$ are continuous with respect to all parameters and variables $s, \overline{s}, a, \overline{a}, s'$, and $\overline{s}'$.

Regularity conditions 2: There exists a b and L such that $\sup_s p_1(s) < b$, $\sup_{\overline{s}} \overline{p}_1(\overline{s}) < b$, $\sup_{a,s,s'} \tau_a(s'|s) < b$, $\sup_{\overline{a},\overline{s},\overline{s}'} \overline{\tau}_{\overline{a}}(\overline{s}'|\overline{s}) < b$, $\sup_{a,s} R(s,a) < b$, $\sup_{\overline{a},\overline{s}} \overline{R}(\overline{s},\overline{a}) < b$, $\sup_{a,s,s'} \|\nabla_a \tau_a(s'|s)\| < L, \sup_{\overline{a},\overline{s},\overline{s}'} \|\nabla_{\overline{a}} \overline{\tau}_{\overline{a}}(\overline{s}'|\overline{s})\| < L$, $\sup_{s,a} \|\nabla_a R(s,a)\| < L, \sup_{\overline{s},\overline{a}} \|\nabla_{\overline{a}} \overline{R}(\overline{s},\overline{a})\| < L$.

We also assume the following conditions on the continuous MDP homomorphism map $h = (f, g_s)$, as discussed in Definition 3:

Regularity conditions 3: The action mapping $g_s(a)$ is a local diffeomorphism (Definition 6). Hence it is continuous with respect to a and locally bijective with respect to a. Additionally, $\nabla_a g_s(a)$ is continuous with respect to the parameter a, and there exists a L such that $\sup_{s,a} \|\nabla_a g_s(a)\| < L$.

C Proofs

Below are the proofs accompanying Sections 3, 4 and 5.

C.1 Proof of Theorem 1: Value Equivalence

Proof. The proof is along the lines of the *optimal value equivalence* theorem of Ravindran and Barto [69]. We define the m-step discounted action value function $Q_m^{\pi^\uparrow}(s,a)$ recursively for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ and for all integers $m \geq 1$ as:

$$Q_m^{\pi^\uparrow}(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} \tau_a(s'|s) \sum_{a' \in \mathcal{A}} \pi^\uparrow(a'|s') Q_{m-1}^{\pi^\uparrow}(s',a'),$$

with $Q_0^{\pi^\uparrow}(s,a) = R(s,a)$. The proof is by induction on m; the base case of $m = 0$ is true because:

$$Q_0^{\pi^\uparrow}(s,a) = R(s,a) = \overline{R}(f(s), g_s(a)) = Q_0^{\overline{\pi}}(f(s), g_s(a)).$$

Now suppose towards induction that $Q_k^{\pi^\uparrow}(s,a) = Q_k^{\overline{\pi}}(f(s), g_s(a))$ for all values of k less than m and all state action pairs $(s,a) \in \mathcal{S} \times \mathcal{A}$. Using the fact that $h = (f, g_s)$ is an MDP homomorphism, we have:

$$\begin{aligned}
Q_m^{\pi^\uparrow}(s,a) &= R(s,a) + \gamma \sum_{s' \in \mathcal{S}} \tau_a(s'|s) \sum_{a' \in \mathcal{A}} \pi^\uparrow(a'|s') Q_{m-1}^{\pi^\uparrow}(s',a') \\
&= R(s,a) + \gamma \sum_{[s']_{B_h|\mathcal{S}} \in B_h|\mathcal{S}} \sum_{s'' \in [s']_{B_h|\mathcal{S}}} \tau_a(s''|s) \sum_{a' \in \mathcal{A}} \pi^\uparrow(a'|s') Q_{m-1}^{\overline{\pi}}(f(s'), g_{s'}(a')) \qquad (16) \\
&= R(s,a) + \gamma \sum_{[s']_{B_h|\mathcal{S}} \in B_h|\mathcal{S}} \sum_{s'' \in [s']_{B_h|\mathcal{S}}} \tau_a(s''|s) \sum_{\overline{a}' \in \overline{\mathcal{A}}} \sum_{a'' \in g_{s'}^{-1}(\overline{a}')} \pi^\uparrow(a''|s') Q_{m-1}^{\overline{\pi}}(f(s'), \overline{a}') \\
&= \overline{R}(f(s), g_s(a)) + \gamma \sum_{[s']_{B_h|\mathcal{S}} \in B_h|\mathcal{S}} \overline{\tau}_{g_s(a)}(f(s')|f(s)) \sum_{\overline{a}' \in \overline{\mathcal{A}}} \overline{\pi}(\overline{a}'|f(s')) Q_{m-1}^{\overline{\pi}}(f(s'), \overline{a}') \qquad (17) \\
&= \overline{R}(f(s), g_s(a)) + \gamma \sum_{\overline{s}' \in \overline{\mathcal{S}}} \overline{\tau}_{g_s(a)}(\overline{s}'|f(s)) \sum_{\overline{a}' \in \overline{\mathcal{A}}} \overline{\pi}(\overline{a}'|\overline{s}') Q_{m-1}^{\overline{\pi}}(\overline{s}', \overline{a}') \\
&= Q_m^{\overline{\pi}}(f(s), g_s(a)).
\end{aligned}$$

Where in equation (16) we used the fact that $Q_{m-1}^{\pi^\uparrow}(s,a) = Q_{m-1}^{\overline{\pi}}(f(s), g_s(a))$ from the induction assumption. In equation (17) we used $\sum_{s'' \in [s']_{B_h|\mathcal{S}}} \tau_a(s''|s) = \overline{\tau}_{g_s}(f(s')|f(s))$ and $\sum_{a'' \in g_{s'}^{-1}(\overline{a}')} \pi^\uparrow(a''|s') = \overline{\pi}(\overline{a}'|f(s'))$ from the definition of MDP homomorphism [69]. Since R and $\overline{R}$ are bounded, it follows by induction that $Q^{\pi^\uparrow}(s,a) = Q^{\overline{\pi}}(f(s), g_s(a))$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$.

The proof for $V^{\pi^\uparrow}(s) = V^{\overline{\pi}}(f(s))$ follows directly from the equivalence of action value functions and the fact that the two policies are tied together through the lifting process because in general we have: $V^\pi(s) = \sum_{a \in A} \pi(a|s) Q^\pi(s,a)$. □

C.2 Proof of Theorem 2: Optimal Value Equivalence for Continuous MDP Homomorphisms

Proof. The proof follows along the same lines as Ravindran and Barto [69]. We will first prove the following claim:

Claim. For $m \geq 1$, define the sequence $Q_m : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as

$$Q_m(s,a) = R(s,a) + \gamma \int_{s' \in \mathcal{S}} \tau_a(ds'|s) \sup_{a' \in \mathcal{A}} Q_{m-1}(s',a')$$

and $Q_0(s,a) = R(s,a)$. Define the sequence $\overline{Q}_m : \overline{\mathcal{S}} \times \overline{\mathcal{A}} \to \mathbb{R}$ analogously. Then for any $(s,a) \in \mathcal{S} \times \mathcal{A}$ we have

$$Q_m(s,a) = \overline{Q}_m(f(s), g_s(a)).$$

We will prove this claim by induction on m. The base case $m = 0$ follows from the reward invariance property of continuous MDP homomorphisms:

$$Q_0(s,a) = R(s,a) = \overline{R}(f(s), g_s(a)) = \overline{Q}_0(f(s), g_s(a)).$$

For the inductive case, note that

$$\begin{aligned}
Q_m(s,a) &= R(s,a) + \gamma \int_{s' \in \mathcal{S}} \tau_a(ds'|s) \sup_{a' \in \mathcal{A}} Q_{m-1}(s', a') \\
&= \overline{R}(f(s), g_s(a)) + \gamma \int_{s' \in \mathcal{S}} \tau_a(ds'|s) \sup_{a' \in \mathcal{A}} \overline{Q}_{m-1}(f(s'), g_{s'}(a')) && (18) \\
&= \overline{R}(f(s), g_s(a)) + \gamma \int_{s' \in \mathcal{S}} \tau_a(ds'|s) \sup_{\overline{a}' \in \overline{\mathcal{A}}} \overline{Q}_{m-1}(f(s'), \overline{a}') && (19) \\
&= \overline{R}(f(s), g_s(a)) + \gamma \int_{\overline{s}' \in \overline{\mathcal{S}}} \overline{\tau}_{g_s(a)}(d\overline{s}'|f(s)) \sup_{\overline{a}' \in \overline{\mathcal{A}}} \overline{Q}_{m-1}(\overline{s}', \overline{a}') && (20) \\
&= Q_{m-1}(f(s), g_s(a)), && (21)
\end{aligned}$$

where Equation 18 follows from the inductive hypothesis, Equation 19 follows from g_s being surjective, and Equation 20 follows from the change of variables formula (Theorem 6); indeed, from Definition 3 we have the pushforward measure of $\tau_a(\cdot|s)$ with respect to f equals $\tau_{g_s(a)}(\cdot|f(s))$ and here $g : \overline{\mathcal{S}} \to \mathbb{R}$ is defined as $g(\overline{s}) = \sup_{\overline{a}' \in \overline{\mathcal{A}}} \overline{Q}_{m-1}(\overline{s}, \overline{a}')$. This concludes the induction proof. Since $\lim_{m\to\infty} Q_m(s,a) = Q^*(s,a)$, it follows that $Q^*(s,a) = \overline{Q}^*(f(s), g_s(a))$.

The proof for $V^*(s) = \overline{V}^*(f(s))$ follows directly from the equivalence of optimal action value functions as $V^*(s) = \max_a Q^*(s,a)$ in general. □

C.3 Proof of Theorem 3: Value Equivalence for Deterministic Policies and Continuous MDP Homomorphisms

Proof. Unlike the proofs of Theorems 1 and 2, here we assume the policy is deterministic due to the complications of lifting stochastic policies discussed in Section 4.2. Therefore, the lifting process can be simply obtained as $\pi^\uparrow(s) = g_s^{-1}(\overline{\pi}(f(s))$ and the inverse of the lifting process is $\overline{\pi}(f(s)) = g_s(\pi^\uparrow(s))$, as the mapping g_s is assumed to be an invertible continuous map (Appendix B).

Similarly to Ravindran and Barto [69], the proof is by induction. We define the m-step discounted action value function $Q_m^{\pi^\uparrow}(s,a)$ for the domain $\mathcal{S} \times \mathcal{A}$ and for all integers $m \geq$ as:

$$Q_m^{\pi^\uparrow}(s,a) = R(s,a) + \gamma \int_{s' \in \mathcal{S}} \tau_a(ds'|s) Q_{m-1}^{\pi^\uparrow}(s', \pi^\uparrow(s')),$$

with $Q_0^{\pi^\uparrow}(s,a) = R(s,a)$ for all pairs $(s,a) \in \mathcal{S} \times \mathcal{A}$. The proof is by induction on m, the base case of $m = 0$ is true because:

$$Q_0^{\pi^\uparrow}(s,a) = R(s,a) = \overline{R}(f(s), g_s(a)) = Q_0^{\overline{\pi}}(f(s), g_s(a)).$$

Now suppose towards induction that $Q_k^{\pi^\uparrow}(s,a) = Q_k^{\overline{\pi}}(f(s), g_s(a))$ for all values of k less than m on the domain $\mathcal{S} \times \mathcal{A}$. Using the fact that $h = (f, g_s)$ is a continuous MDP homomorphism, we have:

$$\begin{aligned}
Q_m^{\pi^\uparrow}(s,a) &= R(s,a) + \gamma \int_{s' \in \mathcal{S}} \tau(ds'|s) Q_{m-1}^{\pi^\uparrow}(s', \pi^\uparrow(s')) \\
&= R(s,a) + \gamma \int_{s' \in \mathcal{S}} \tau_a(ds'|s) Q_{m-1}^{\overline{\pi}}(f(s'), g_{s'}(\pi^\uparrow(s'))) && (22) \\
&= \overline{R}(f(s), g_s(a)) + \gamma \int_{s' \in \mathcal{S}} \tau_a(ds'|s) Q_{m-1}^{\overline{\pi}}(f(s'), \overline{\pi}(f(s'))) && (23) \\
&= \overline{R}(f(s), g_s(a)) + \gamma \int_{\overline{s} \in \overline{\mathcal{S}}} \overline{\tau}_{g_s(a)}(d\overline{s}|f(s)) Q_{m-1}^{\overline{\pi}}(\overline{s}', \overline{\pi}(\overline{s}')) && (24) \\
&= Q_m^{\overline{\pi}}(f(s), g_s(a)). && (25)
\end{aligned}$$

Where in equation (22), we used the induction assumption,in equation (23) we used the definition the inverse of policy lifting as defined above, and in equation (24) we applied the change of variables

formula (Theorem 6) using the fact that $\overline{\tau}_{g_s(a)}(\cdot|f(s))$ is the pushforward measure of $\tau_a(\cdot|s)$ under f by definition. Since R and $\overline{R}$ are bounded, it follows by induction that $Q^{\pi^\uparrow}(s,a) = Q^{\overline{\pi}}(f(s), g_s(a))$.

The proof for $V^{\pi^\uparrow}(s) = V^{\overline{\pi}}(f(s))$ follows directly from the equivalence of action value functions and the fact that the two policies are tied together through the lifting process because $V^{\pi}(s) = Q^{\pi}(s, \pi(s))$ for deterministic policies.

□

C.4 Proof of Theorem 4: Equivalence of Deterministic Policy Gradients

Proof. Assuming the conditions described in Appendix B, we first take the derivative of the deterministic policy lifting relation w.r.t. the policy parameters θ using the chain rule:

$$\begin{aligned}
(g_s \circ \pi^\uparrow)(s) &= (\overline{\pi} \circ f)(s) \\
d(g_s \circ \pi^\uparrow)_\theta(s) &= d(\overline{\pi} \circ f)_\theta(s) \\
d(g_s)_{\pi^\uparrow(s)} \circ d(\pi^\uparrow)_\theta(s) &= d(\overline{\pi} \circ f)_\theta(s) \\
\underbrace{\nabla_a g_s(a)\big|_{a=\pi^\uparrow(s)}}_{P} \nabla_\theta \pi^\uparrow(s) &= \nabla_\theta \overline{\pi}(f(s)),
\end{aligned} \tag{26}$$

where $\circ$ is the composition operator and the dimensions of the matrices are $P \in \mathbb{R}^{|\overline{\mathcal{A}}| \times |A|}$, $\nabla_\theta \pi^\uparrow(s) \in \mathbb{R}^{|A| \times |\theta|}$, and $\nabla_\theta \overline{\pi}(\overline{s}) \in \mathbb{R}^{|\overline{\mathcal{A}}| \times |\theta|}$. Second, we take the derivative of the value equivalence theorem w.r.t. the actions a using the chain rule:

$$\begin{aligned}
Q^{\pi^\uparrow}(s,a) &= Q^{\overline{\pi}}(f(s), g_s(a)) \\
dQ^{\pi^\uparrow}(s,a)_a &= dQ^{\overline{\pi}}(f(s), g_s(a))_a \\
\nabla_a Q^{\pi^\uparrow}(s,a)\big|_{a=\pi^\uparrow(s)} &= \nabla_{\overline{a}} Q^{\overline{\pi}}(f(s), \overline{a})\big|_{\overline{a}=\overline{\pi}(f(s))} \underbrace{\nabla_a g_s(a)\big|_{a=g_s^{-1}(\overline{\pi}(f(s)))}}_{P},
\end{aligned} \tag{27}$$

where the dimensions of the matrices are $\nabla_a Q^{\pi^\uparrow}(s,a) \in \mathbb{R}^{|A|}$, $\nabla_{\overline{a}} Q^{\overline{\pi}}(\overline{s}, \overline{a}) \in \mathbb{R}^{|\overline{\mathcal{A}}|}$, and similarly as before $P \in \mathbb{R}^{|\overline{\mathcal{A}}| \times |A|}$. As we assumed the g_s to be a local diffeomorphism, the inverse function theorem (Theorem 7) states that the matrix P is invertible, thus we right-multiply both sides of equation (27) by P^{-1} and left-multiply the resulting equation by equation (26) to obtain the desired result:

$$\begin{aligned}
\nabla_a Q^{\pi^\uparrow}(s,a)\big|_{a=\pi^\uparrow(s)} P^{-1} P \nabla_\theta \pi^\uparrow(s) &= \nabla_{\overline{a}} Q^{\overline{\pi}}(f(s), \overline{a})\big|_{\overline{a}=\overline{\pi}(f(s))} \nabla_\theta \overline{\pi}(f(s)) \\
\nabla_a Q^{\pi^\uparrow}(s,a)\big|_{a=\pi^\uparrow(s)} \nabla_\theta \pi^\uparrow(s) &= \nabla_{\overline{a}} Q^{\overline{\pi}}(f(s), \overline{a})\big|_{\overline{a}=\overline{\pi}(f(s))} \nabla_\theta \overline{\pi}(f(s)).
\end{aligned} \tag{28}$$

□

C.5 Proof of Theorem 5: Homomorphic Policy Gradient

Proof. The proof follows along the same lines of the deterministic policy gradient theorem [74], but with additional steps for changing the integration space from $\mathcal{S}$ to $\overline{\mathcal{S}}$. First, we derive a recursive expression for $\nabla_\theta V^{\pi_\theta^\uparrow}(s)$ as:

$$\begin{aligned}
\nabla_\theta V^{\pi_\theta^\uparrow}(s) &= \nabla_\theta Q^{\pi_\theta^\uparrow}\Big(s, \pi_\theta^\uparrow(s)\Big) \\
&= \nabla_\theta \Big[R(s, \pi_\theta^\uparrow(s)) + \gamma \int_{\mathcal{S}} \tau_{\pi_\theta^\uparrow(s)}(s, ds') V^{\pi_\theta^\uparrow}(s') \Big] \\
&= \nabla_\theta \pi_\theta^\uparrow(s) \nabla_a R(s,a)\Big|_{a=\pi_\theta^\uparrow(s)} \\
&\quad + \gamma \int_{\mathcal{S}} \Big[\tau_{\pi_\theta^\uparrow(s)}(s, ds') \nabla_\theta V^{\pi_\theta^\uparrow}(s') + \nabla_\theta \pi_\theta^\uparrow(s) \nabla_a \tau_a(s, ds')\Big|_{a=\pi_\theta^\uparrow(s)} V^{\pi_\theta^\uparrow}(s') \Big] \\
&= \nabla_\theta \pi_\theta^\uparrow(s) \nabla_a \Big[R\big(s,a\big) + \gamma \int_{\mathcal{S}} \tau_a(s, ds') V^{\pi_\theta^\uparrow}(s') \Big]\Big|_{a=\pi_\theta^\uparrow(s)} + \gamma \int_{\mathcal{S}} \tau_{\pi_\theta^\uparrow(s)}(s, ds') \nabla_\theta V^{\pi_\theta^\uparrow}(s')
\end{aligned} \tag{29}$$

$$= \nabla_\theta \pi_\theta^\uparrow(s) \nabla_a Q^{\pi_\theta^\uparrow}(s,a)\Big|_{a=\pi_\theta^\uparrow(s)} + \gamma \int_{\mathcal{S}} \tau_{\pi_\theta^\uparrow(s)}(s, ds') \nabla_\theta V^{\overline{\pi}_\theta}(f(s')) \tag{30}$$

$$= \nabla_\theta \pi_\theta^\uparrow(s) \nabla_a Q^{\pi_\theta^\uparrow}(s,a)\Big|_{a=\pi_\theta^\uparrow(s)} + \gamma \int_{\overline{\mathcal{S}}} \overline{\tau}_{g_s(\pi_\theta^\uparrow(s))}(f(s), d\overline{s}') \nabla_\theta V^{\overline{\pi}_\theta}(f(s')) \tag{31}$$

$$= \nabla_\theta \overline{\pi}_\theta(f(s)) \nabla_{\overline{a}} Q^{\overline{\pi}_\theta}(f(s),\overline{a})\Big|_{\overline{a}=\overline{\pi}_\theta(f(s))} + \gamma \int_{\overline{\mathcal{S}}} \overline{\tau}_{\overline{\pi}_\theta(\overline{s})}(\overline{s}, d\overline{s}') \nabla_\theta V^{\overline{\pi}_\theta}(\overline{s}') \tag{32}$$

$$= \nabla_\theta \overline{\pi}_\theta(f(s)) \nabla_{\overline{a}} Q^{\overline{\pi}_\theta}(f(s),\overline{a})\Big|_{\overline{a}=\overline{\pi}_\theta(f(s))} + \gamma \int_{\overline{\mathcal{S}}} p(\overline{s} \to \overline{s}', 1, \overline{\pi}_\theta) \nabla_\theta V^{\overline{\pi}_\theta}(\overline{s}') d\overline{s}'.$$

Where $p(\overline{s} \to \overline{s}', t, \overline{\pi}_\theta)$ is the probability of going from $\overline{s}$ to $\overline{s}'$ under the policy $\overline{\pi}_\theta(\overline{s})$ in t time steps. In equation (29) we were able to apply the Leibniz integral rule to exchange the order of derivative and integration because of the regularity conditions on the continuity of the functions. In equation (30) we used the value equivalence property, and in equation (31) we used the change of variables formula based on the pushforward measure (6) of $\tau_a(s,.)$ with respect to f. Finally, in equation (32) we used the equivalence of policy gradients from Theorem 4. By recursively rolling out the formula above, we obtain:

$$\begin{aligned}
\nabla_\theta V^{\pi_\theta^\uparrow}(s) &= \nabla_\theta \overline{\pi}_\theta(f(s)) \nabla_{\overline{a}} Q^{\overline{\pi}_\theta}(f(s),\overline{a})\Big|_{\overline{a}=\overline{\pi}_\theta(f(s))} \\
&\quad + \gamma \int_{\overline{\mathcal{S}}} p(\overline{s} \to \overline{s}', 1, \overline{\pi}_\theta) \nabla_\theta \overline{\pi}_\theta(f(s')) \nabla_{\overline{a}} Q^{\overline{\pi}_\theta}(f(s'),\overline{a})\Big|_{\overline{a}=\overline{\pi}_\theta(f(s'))} d\overline{s}' \\
&\quad + \gamma^2 \int_{\overline{\mathcal{S}}} p(\overline{s} \to \overline{s}', 1, \overline{\pi}_\theta) \int_{\overline{\mathcal{S}}} p(\overline{s}' \to \overline{s}'', 1, \overline{\pi}_\theta) \nabla_\theta V^{\pi_\theta^\uparrow}(f(s'')) d\overline{s}'' d\overline{s}' \\
&= \nabla_\theta \overline{\pi}_\theta(f(s)) \nabla_{\overline{a}} Q^{\overline{\pi}_\theta}(f(s),\overline{a})\Big|_{\overline{a}=\overline{\pi}_\theta(f(s))} \\
&\quad + \gamma \int_{\overline{\mathcal{S}}} p(\overline{s} \to \overline{s}', 1, \overline{\pi}_\theta) \nabla_\theta \overline{\pi}_\theta(f(s')) \nabla_{\overline{a}} Q^{\overline{\pi}_\theta}(f(s'),\overline{a})\Big|_{\overline{a}=\overline{\pi}_\theta(f(s'))} d\overline{s}' \\
&\quad + \gamma^2 \int_{\overline{\mathcal{S}}} p(\overline{s} \to \overline{s}'', 2, \overline{\pi}_\theta) \nabla_\theta V^{\overline{\pi}_\theta}(f(s'')) d\overline{s}'' \qquad (33) \\
&\vdots \\
&= \int_{\overline{\mathcal{S}}} \sum_{t=0}^{\infty} \gamma^t p(\overline{s} \to \overline{s}', t, \overline{\pi}_\theta) \nabla_\theta \overline{\pi}_\theta(f(s)) \nabla_{\overline{a}} Q^{\overline{\pi}_\theta}(f(s),\overline{a})\Big|_{\overline{a}=\overline{\pi}_\theta(f(s))} d\overline{s}'. \qquad (34)
\end{aligned}$$

Where in equation (33) we exchanged the order of integration using the Fubini's theorem that requires the boundedness of $\|\nabla_\theta V^{\overline{\pi}_\theta}(s)\|$ as described in the regularity conditions. Finally, we take the expectation of $\nabla_\theta V^{\pi_\theta^\uparrow}(s)$ over the initial state distribution:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \int_{\mathcal{S}} p_1(s) V^{\pi_\theta^\uparrow}(s) ds \\
&= \int_{\mathcal{S}} p_1(s) \nabla_\theta V^{\pi_\theta^\uparrow}(s) ds \\
&= \int_{\mathcal{S}} p_1(s) \int_{\overline{\mathcal{S}}} \sum_{t=0}^{\infty} \gamma^t p(\overline{s} \to \overline{s}', t, \overline{\pi}_\theta) \nabla_\theta \overline{\pi}_\theta(f(s)) \nabla_{\overline{a}} Q^{\overline{\pi}_\theta}(f(s),\overline{a})\Big|_{\overline{a}=\overline{\pi}_\theta(f(s))} d\overline{s}' ds \\
&= \int_{\overline{\mathcal{S}}} \overline{p}_1(\overline{s}) \int_{\overline{\mathcal{S}}} \sum_{t=0}^{\infty} \gamma^t p(\overline{s} \to \overline{s}', t, \overline{\pi}_\theta) \nabla_\theta \overline{\pi}_\theta(f(s)) \nabla_{\overline{a}} Q^{\overline{\pi}_\theta}(f(s),\overline{a})\Big|_{\overline{a}=\overline{\pi}_\theta(f(s))} d\overline{s}' d\overline{s} \qquad (35) \\
&= \int_{\mathcal{S}} \rho^{\overline{\pi}_\theta}(\overline{s}) \nabla_\theta \overline{\pi}_\theta(\overline{s}) \nabla_{\overline{a}} Q^{\overline{\pi}_\theta}(\overline{s},\overline{a})\Big|_{\overline{a}=\overline{\pi}_\theta(\overline{s})} d\overline{s}. \qquad (36)
\end{aligned}$$

Where $\rho^{\overline{\pi}_\theta}(\overline{s})$ is the discounted stationary distribution induced by the policy $\overline{\pi}_\theta$. In equation (35) we used the change of variable formula. Similar to the steps before, we have used the Leibniz integral rule to exchange the order of integration and derivative, used Fubini's theorem to exchange the order of integration. □

D Full Results

As discussed in Section 7, we evaluate DHPG on continuous control tasks from DM Control on state and pixel observations. Importantly, to reliably evaluate our algorithm against the baselines and to correctly capture the distribution of results, we follow the best practices proposed by Agarwal et al. [5] and report the interquartile mean (IQM) and performance profiles aggregated on all tasks over 10 random seeds. While our baseline results are obtained using the official code, when possible, some of the results may differ from the originally reported ones due to the difference in the seed numbers and our goal to present a faithful representation of the true performance distribution [5].

We use the official implementations of DrQv2, DBC, and SAC-AE, while we re-implement DeepMDP due to the unavailability of the official code; See Appendix E.3 for full details on the baselines.

D.1 State Observations

Figure 8 shows full results obtained on 18 DeepMind Control Suite tasks with state observations to supplement results of Section 7.1. Domains that require excessive exploration and large number of time steps (e.g., acrobot, swimmer, and humanoid) are not included in this benchmark.

Figures 9 and 10 respectively show performance profiles and aggregate metrics [5] on 17 tasks; hopper hop is removed from RLiable evaluation as none of the algorithms have acquired reasonable performance in 1 million steps.

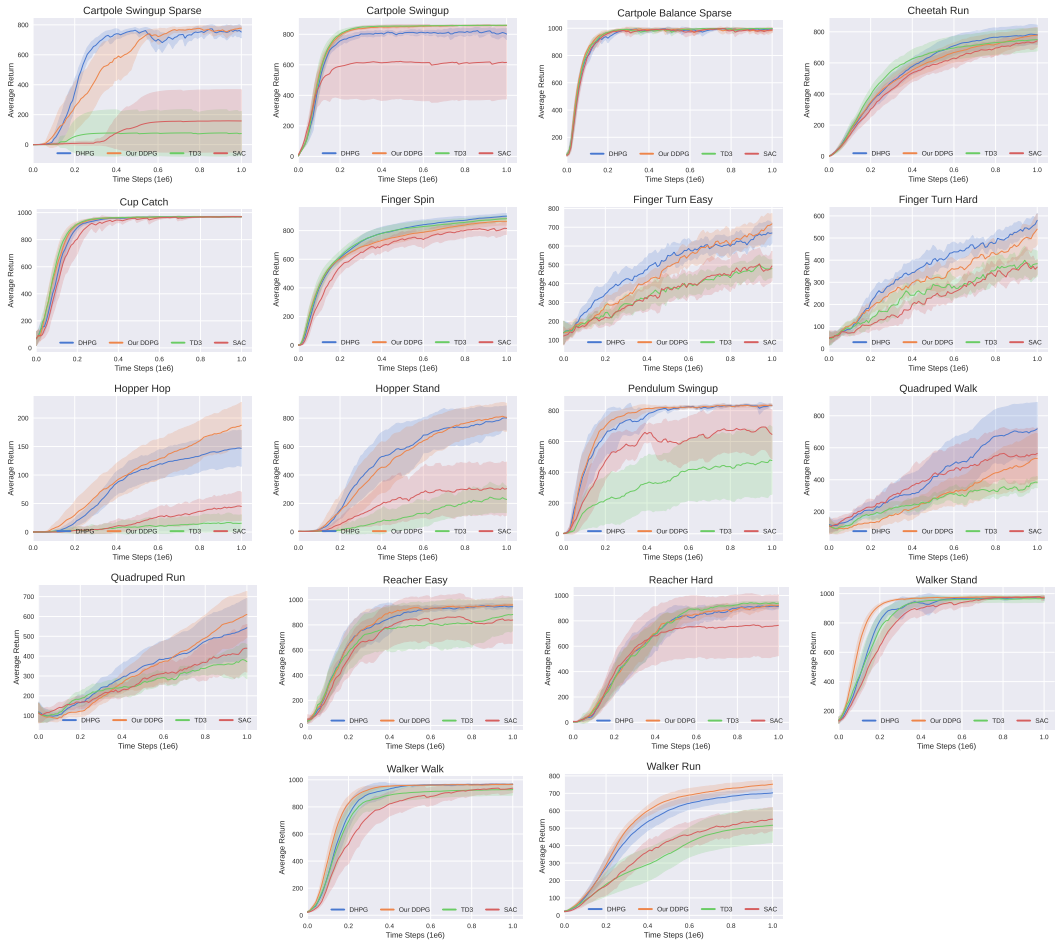
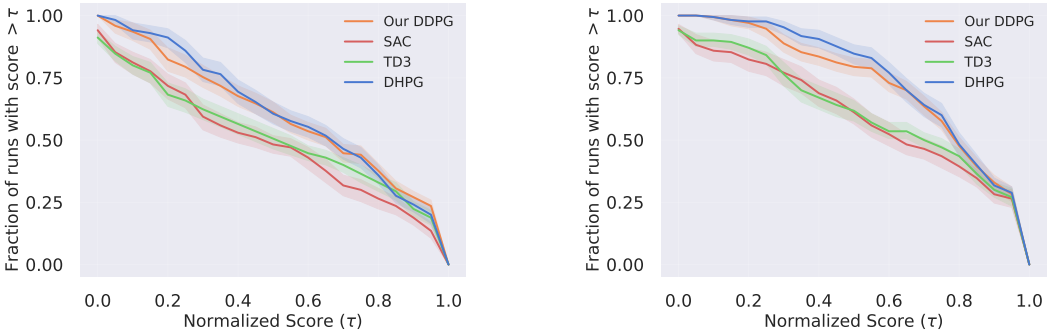


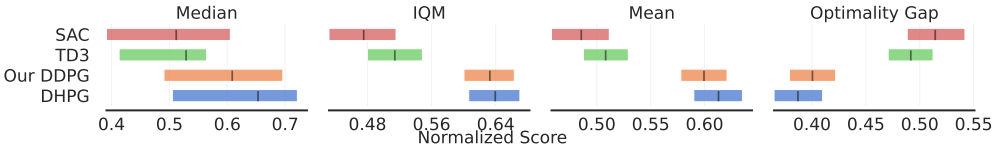
Figure 8: Learning curves for 18 DM control tasks with **state observations**. Mean performance is obtained over 10 seeds and shaded regions represent 95% confidence intervals. Plots are smoothed uniformly for visual clarity.



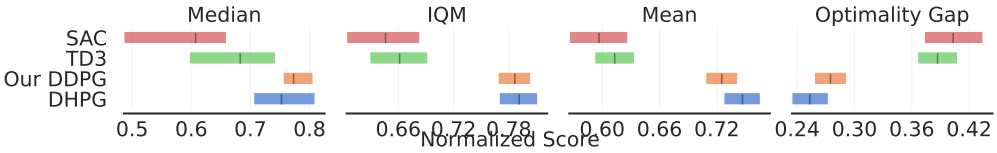
(a) 250k step benchmark.

(b) 500k step benchmark.

Figure 9: Performance profiles for **state observations** based on 17 tasks over 10 seeds, at 250k steps **(a)**, and at 500k steps **(b)**. Shaded regions represent 95% confidence intervals.



(a) 250k step benchmark.



(b) 500k step benchmark.

Figure 10: Aggregate metrics for **state observations** with 95% confidence intervals based on 17 tasks over 10 seeds, at 250k steps **(a)**, and at 500k steps **(b)**.

D.2 Pixel Observations

Figure 11 shows full results obtained on 16 DeepMind Control Suite tasks with pixel observations to supplement results of Section 7.2. Domains that require excessive exploration and large number of time steps (e.g., acrobot, swimmer, and humanoid) and domains with visually small targets (e.g., reacher hard and finger turn hard) are not included in this benchmark. In each plot, the solid lines present algorithms with image augmentation and dashed lines present algorithms without image augmentation.

Figures 12 and 13 respectively show performance profiles and aggregate metrics [5] on 14 tasks; hopper hop and walker run are removed from RLiable evaluation as none of the algorithms have acquired reasonable performance in 1 million steps.

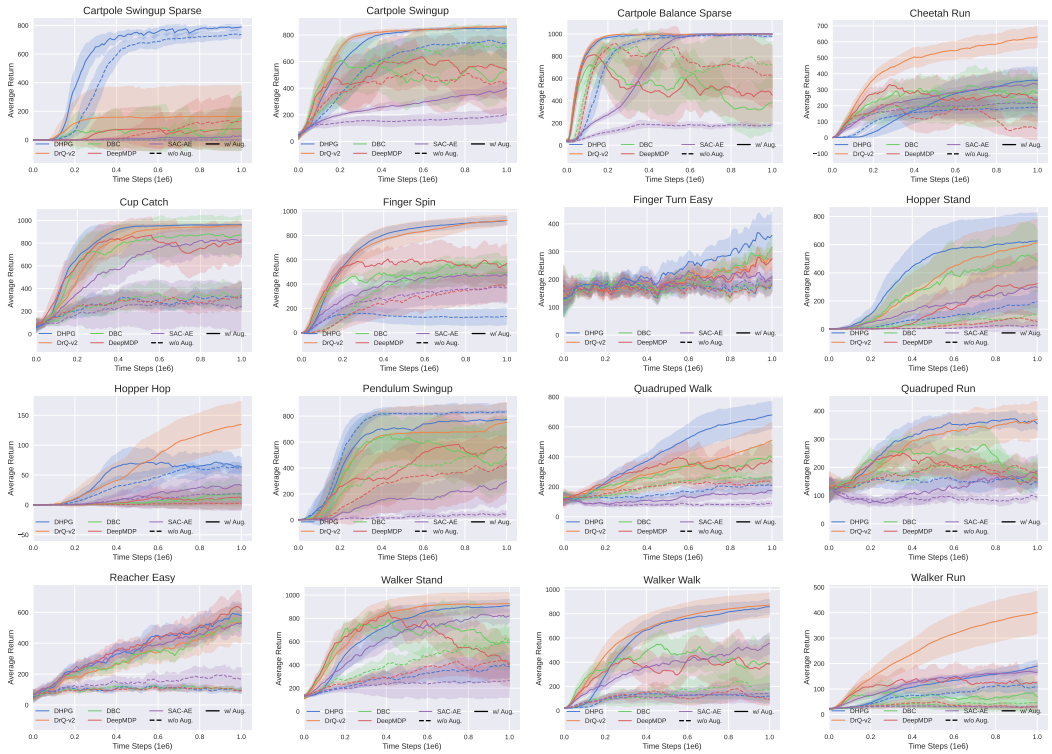
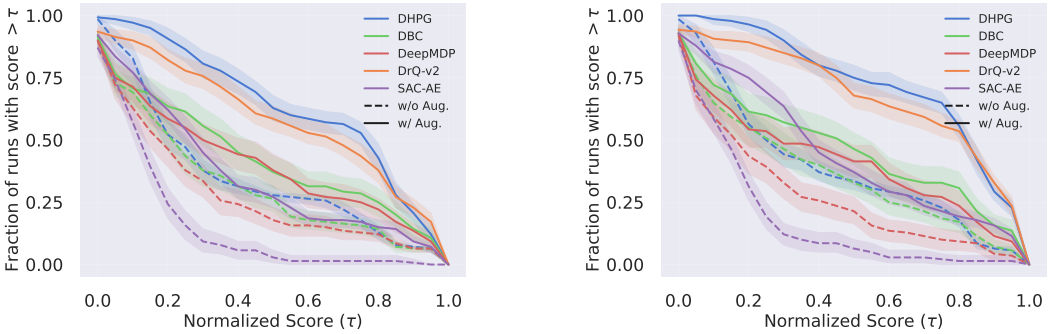


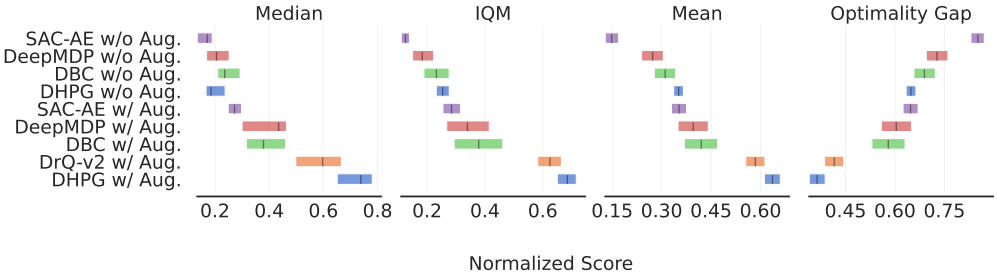
Figure 11: Learning curves for 16 DM control tasks with **pixel observations**. Mean performance is obtained over 10 seeds and shaded regions represent 95% confidence intervals. Plots are smoothed uniformly for visual clarity.



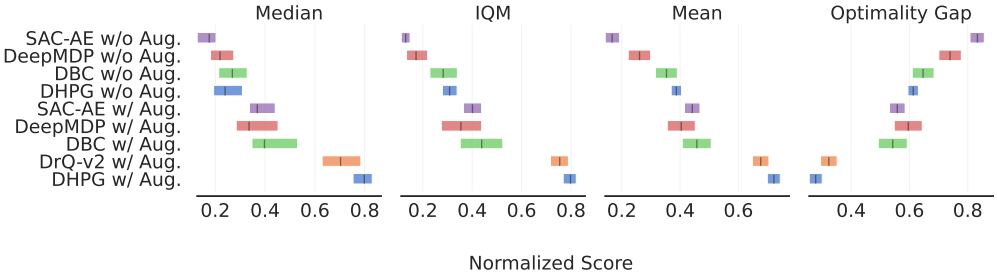
(a) 500k step benchmark.

(b) 1m step benchmark.

Figure 12: Performance profiles for **pixel observations** based on 14 tasks over 10 seeds, at 500k steps **(a)**, and at 1m steps **(b)**. Shaded regions represent 95% confidence intervals.



(a) 500k step benchmark.



(b) 1m step benchmark.

Figure 13: Aggregate metrics for **pixel observations** with 95% confidence intervals based on 14 tasks over 10 seeds, at 500k steps **(a)**, and at 1m steps **(b)**.

D.3 Transfer Learning Experiments

As discussed in Section 7.2, the purpose of transfer experiments is to ensure that using MDP homomorphisms does not compromise transfer abilities. Figure 14 shows learning curves for a series of transfer scenarios in which the critic, actor, and representations are transferred to a new task within the same domain. DHPG matches the same transfer abilities of other methods.

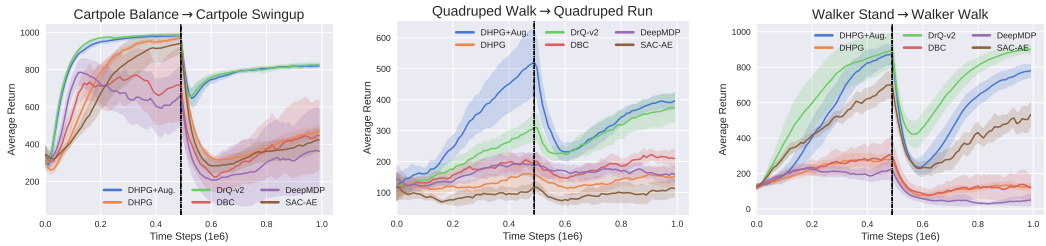


Figure 14: Learning curves for transfer experiments with **pixel observations**. At 500k time step mark, all components are transferred to a new task on the same domain. Mean performance is obtained over 10 seeds and shaded regions represent 95% confidence intervals. Plots are smoothed uniformly for visual clarity.

D.4 Value Equivalence Property in Practice

We can use the value equivalence between the critics of the actual and abstract MDPs as a measure for the quality of learned MDP homomorphismsm, since the two critics are not directly trained to minimize this distance, instead they have equivalent values through the learned MDP homomorphism map. Figure 15 shows the normalized mean absolute error of $|Q(s,a) - \overline{Q}(\overline{s},\overline{a})|$ during training, indicating the property is holding in practice. Expectedly, for lower-dimensional tasks with easily learnable homomorphism maps (e.g., cartpole) the error is reduced earlier than more complicated tasks (e.g., quadruped and walker). But importantly, in all cases the error decreases over time and is at a reasonable range towards the end of the training, meaning the continuous MDP homomorphisms is adhering to conditions of Definition 3.

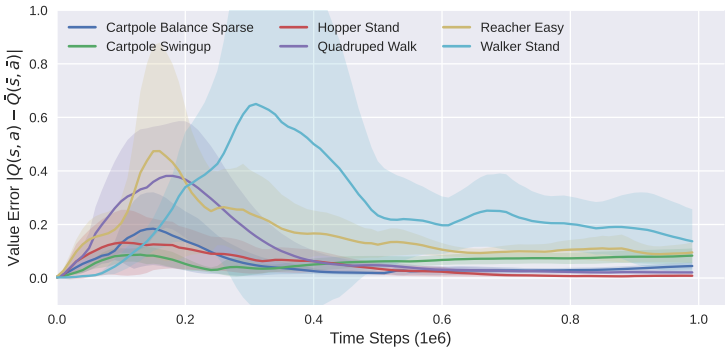


Figure 15: Normalized mean absolute error $|Q(s,a) - \overline{Q}(\overline{s},\overline{a})|$ as a measure for the value equivalence property during training of different tasks from **pixel observations**. The error is measured on samples from the replay buffer and is normalzied by the range of the value function. The error is averaged over 10 seeds and shaded regions represent 95% confidence intervals.

D.5 Ablation Study on the Combination of HPG with DPG

We carry out an ablation study on the combination of HPG with DPG for actor updates as indicated discussed in Section 6. To that end, we evaluate the performance of four variants of DHPG (all using image augmentation) on pixel observations:

1. **DHPG:** Gradients of HPG and DPG are added together and a single actor update is done based on the sum of gradients. This is the standard DHPG algorithm that is used throughout the paper.
2. **DHPG with independent DPG update:** Gradients of HPG and DPG are independently used to update the actor.
3. **DHPG without DPG update:** Only HPG is used to update the actor.
4. **DHPG with single critic:** A single critic network is trained for learning values of both the actual and abstract MDP. Consequently, HPG and DPG are used to update the actor.

Figure 16 shows learning curves obtained on 16 DeepMind Control Suite tasks with pixel observations, and Figure 17 shows RLiable [5] evaluation metrics. In general, summing the gradients of HPG and DPG (variant 1) results in lower variance of gradient estimates compared to independent HPG and DPG updates (variant 2). Interestingly, the variant of DHPG without DPG (variant 3) performs reasonably well or even outperforms other variants in simple tasks where learning MDP homomorphisms is easy (e.g., cartpole and pendulum), indicating the effectiveness of our method in using **only** the abstract MDP to update the policy of the actual MDP. However, in the case of more complicated tasks (e.g., walker), DPG is required to additionally use the actual MDP for policy optimization. Finally, using a single critic for both the actual and abstract MDPs (variant 4) can improve sample efficiency in symmetrical MDPs, but may result in performance drops in non-symmetrical MDPs due to the large error bound between the two MDPs, $\|Q^{\pi^{\uparrow}}(s,a) - Q^{\overline{\pi}}(\overline{s},\overline{a})\|$ [83].

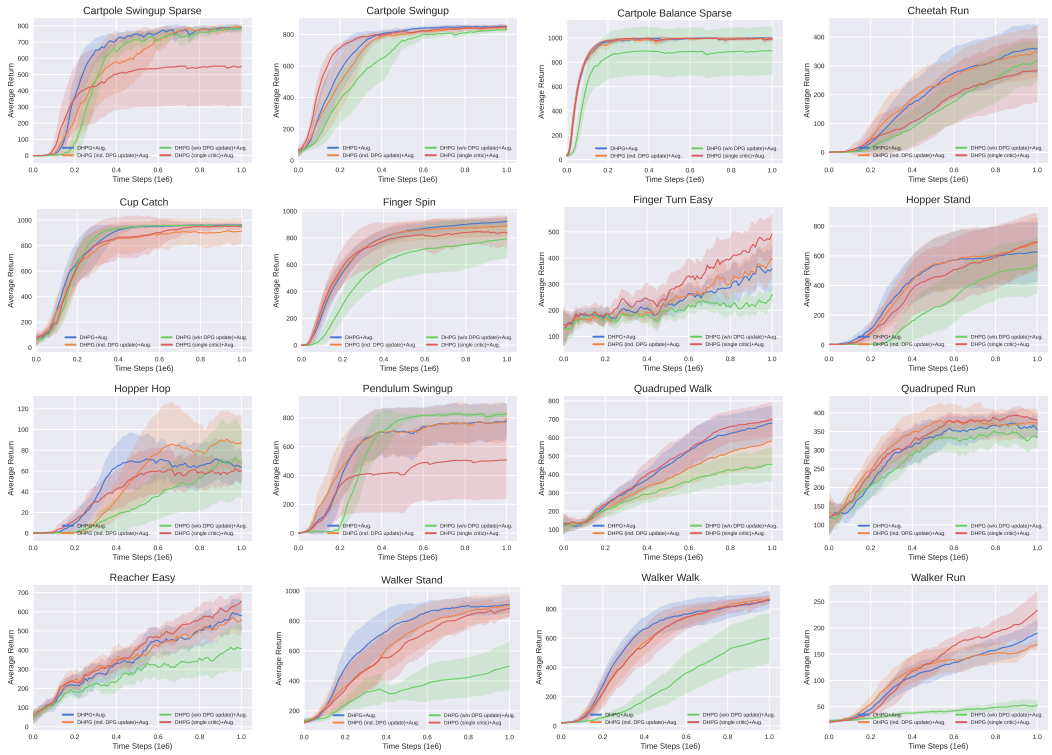
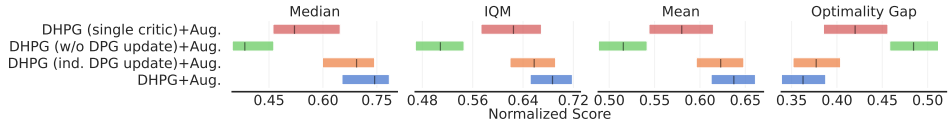
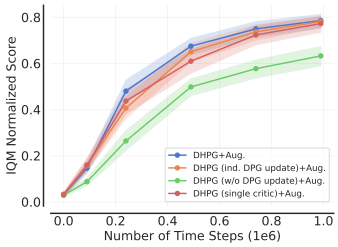


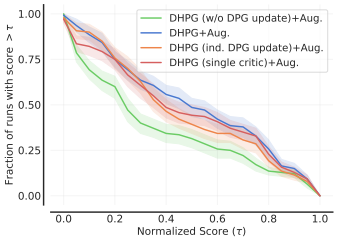
Figure 16: Ablation study on the combination of HPG and DPG. Learning curves for 16 DM control tasks with **pixel observations**. Mean performance is obtained over 10 seeds and shaded regions represent 95% confidence intervals. Plots are smoothed uniformly for visual clarity.



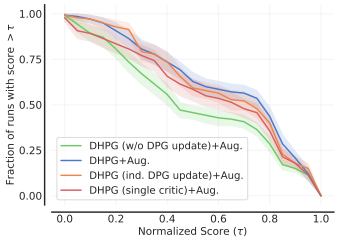
(a) Aggregate metrics at 500k steps.



(b) Sample efficiency.



(c) Performance profiles at 250k steps.

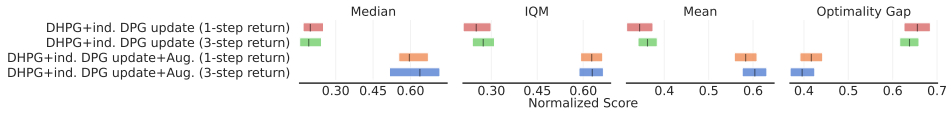


(d) Performance profiles at 500k steps.

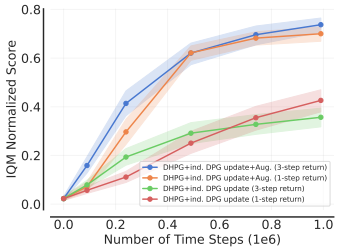
Figure 17: Ablation study on the combination of HPG and DPG. RLiable evaluation metrics for **pixel observations** averaged on 14 tasks over 10 seeds. Aggregate metrics at 500k steps **(a)**, IQM scores as a function of number of steps for comparing sample efficiency **(b)**, performance profiles at 250k steps **(c)**, performance profiles at 500k steps **(d)**. Shaded regions represent 95% confidence intervals.

D.6 Ablation Study on n-step Return

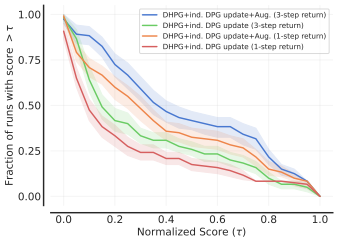
We carry out an ablation study on the choice of n-step return for DHPG. Figure 18 shows RLiable [5] evaluation metrics for DHPG with 1-step and 3-step returns for pixel observations. We show the impact of n-step return on DHPG with and without image augmentation. Overall, n-step return appears to improve the early stages of training. In the case of DHPG without image augmentation, the final performance of 1-step return is better than 3-step return, perhaps indicating that using n-step return can render learning MDP homomorphisms more difficult.



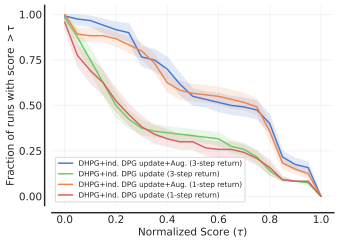
(a) Aggregate metrics at 500k steps.



(b) Sample efficiency.



(c) Performance profiles at 250k steps.



(d) Performance profiles at 500k steps.

Figure 18: Ablation study on n-step return. RLiable evaluation metrics for **pixel observations** averaged on 12 tasks over 10 seeds. Aggregate metrics at 1m steps **(a)**, IQM scores as a function of number of steps for comparing sample efficiency **(b)**, performance profiles at 250k steps **(c)**, and performance profiles at 500k steps **(d)**. Shaded regions represent 95% confidence intervals.

D.7 Comparison Against Higher-Capacity Baselines

The DHPG algorithm contains additional networks, such as the parameterized MDP homomorphism map and the abstract critic, thus it may have a higher network capacity compared to the baselines. To control for the effect of the network capacity and for a fair evaluation, we compare DHPG with higher-capacity variants of DBC and DrQ-v2 that have a larger critic networks. First, we provide a detailed list of network parameters based on the architecture described in Appendix E.2:

1. DHPG: image encoder (1,990,518) + actor (79,105) + critic (79,361) + dynamics model (117,348) + reward model (79,105) + abstract critic (91,905) + f (91,698) + g (91,954) = 2,620,994
2. DBC: image encoder (1,990,518) + actor (79,362) + critic (158,722) + dynamics model (104,804) + reward model (79,105) = 2,412,511
3. DrQ-v2: image encoder (1,990,518) + actor (79,105) + critic (158,722) = 2,228,602

To account for the parameter increase, we present variations of DBC and DrQ with a larger critic (512 hidden dim compared to the initial 256). Consequently, the new total number of parameters for DBC and DrQ are respectively 2,833,375 and 2,649,466. Figure 19 shows full results obtained on 16 DeepMind Control Suite tasks with pixel observations for higher-capacity variants of DBC and DrQ-v2 to supplement results of Section 7.2. Domains that require excessive exploration and large number of time steps (e.g., acrobot, swimmer, and humanoid) and domains with visually small targets (e.g., reacher hard and finger turn hard) are not included in this benchmark. In each plot, the solid lines present algorithms with image augmentation and dashed lines present algorithms without image augmentation.

Figures 20 and 21 respectively show performance profiles and aggregate metrics [5] on 14 tasks; hopper hop and walker run are removed from RLiable evaluation as none of the algorithms have acquired reasonable performance in 1 million steps.

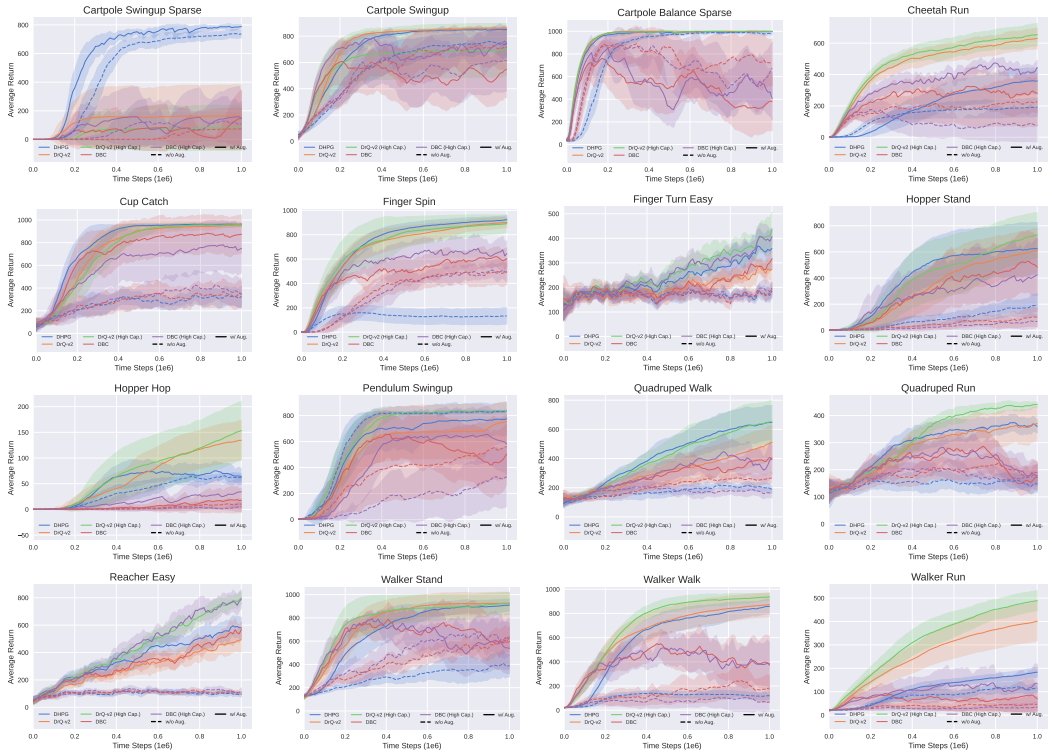
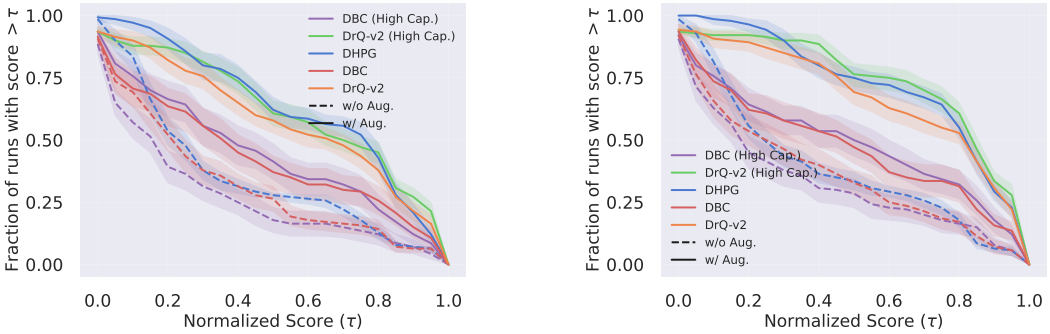


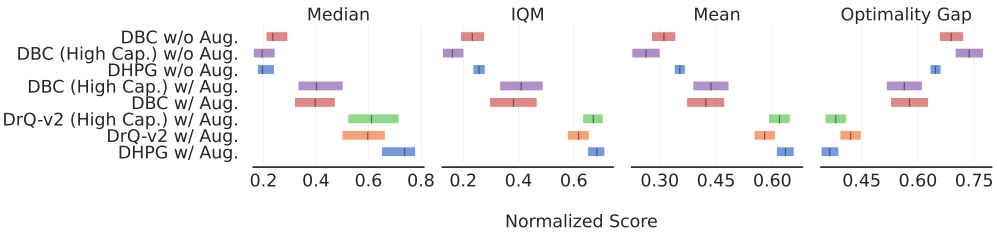
Figure 19: Learning curves for 16 DM control tasks with **pixel observations** for **higher-capacity variants** of DBC and DrQ-v2. Mean performance is obtained over 10 seeds and shaded regions represent 95% confidence intervals. Plots are smoothed uniformly for visual clarity.



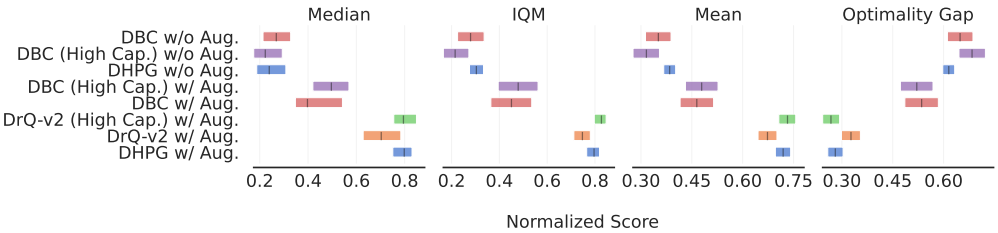
(a) 500k step benchmark.

(b) 1m step benchmark.

Figure 20: Performance profiles for **pixel observations** for **higher-capacity variants** of DBC and DrQ-v2 based on 14 tasks over 10 seeds, at 500k steps **(a)**, and at 1m steps **(b)**. Shaded regions represent 95% confidence intervals.



(a) 500k step benchmark.



(b) 1m step benchmark.

Figure 21: Aggregate metrics for **pixel observations** for **higher-capacity variants** of DBC and DrQ-v2 with 95% confidence intervals based on 14 tasks over 10 seeds, at 500k steps **(a)**, and at 1m steps **(b)**.

E Implementation Details

E.1 Pseudo-code

Algorithm 1 presents the details of the Deep Homomorphic Policy Gradient (DHPG) for pixel observations. This is the main variant used throughout the paper, in which policy gradients obtained from DPG and HPG are added together before updating the actor. For clarity, here the TD error is estimated with 1-step returns.

In the image augmentation version of DHPG, as well as all the baselines, we use image augmentation of DrQ [96] that simply applies random shifts to pixel observations. First, 84×84 images are padded by 4 pixels (by repeating boundary pixels), and then a random 84×84 crop is selected, rendering the original image shifted by ± 4 pixels. Similarly to Yarats et al. [95], we also apply bilinear interpolation on top of the shifted image by replacing each pixel value with the average of four nearest pixel values.

In order to use DHPG for state observations, Lines 8-11 should be simply removed.

Algorithm 1 Deep Homomorphic Policy Gradient (DHPG) for Pixel Observations

1: **Hyperparameters:**
Target network update weight α, actor update delay d, clipped noise parameters c and σ.
2: **Inputs:**
Policy $\pi_\theta(s, a)$, actual critic $Q_\psi(s, a)$, abstract critic $\overline{Q}_{\overline{\psi}}(\overline{s}, \overline{a})$, MDP homomorphism map $h_{\phi,\eta} = (f_\phi(s), g_\eta(s, a))$, reward predictor $\overline{R}_\rho(\overline{s})$, transition model $\tau_\nu(\overline{s}'|\overline{s}, \overline{a})$, CNN image encoder E_μ, and replay buffer $\mathcal{B}$.
3: Initialize target networks $\psi' \leftarrow \psi$, $\overline{\psi'} \leftarrow \overline{\psi}$, $\theta' \leftarrow \theta$.
4: **for** $t = 1$ **to** T **do**
5: Select action with exploration noise $a \sim \pi_\theta(E_\mu(s)) + \epsilon$, where $\epsilon \sim N(0, \sigma)$
6: Store transition (s, a, r, s') in $\mathcal{B}$
7: Sample mini-batch $B_i \sim \mathcal{B}$

8: **if** using image augmentation **then**
9: $s \leftarrow \text{aug}(s),\ s' \leftarrow \text{aug}(s')$
10: **end if**
11: Encode pixel observations: $s \leftarrow E_\mu(s),\ s' \leftarrow E_\mu(s')$

12: **Critic and MDP Homomorphism Update:**
13: Compute MDP homomorphism loss: $\mathcal{L}_{\text{lax}}(\phi, \eta, \mu) + \mathcal{L}_{\text{h}}(\phi, \eta, \rho, \nu, \mu)$ ▷ Equations (12-13)
14: Add clipped noise: $a' \leftarrow \pi_{\theta'}(s') + \epsilon$, where $\epsilon \sim \text{clip}(N(0, \sigma), -c, c)$ ▷ TD3 [30]
15: Compute critic loss: $\mathcal{L}_{\text{actual critic}}(\psi) + \mathcal{L}_{\text{abstract critic}}(\overline{\psi}, \phi, \eta)$ ▷ Equations (9-10)
16: Update: $\psi, \overline{\psi}, \phi, \eta, \rho, \nu, \mu \leftarrow \arg\min_{\psi,\overline{\psi},\phi,\eta,\rho,\nu,\mu} \mathcal{L}_{\text{lax}} + \mathcal{L}_{\text{h}} + \mathcal{L}_{\text{actual critic}} + \mathcal{L}_{\text{abstract critic}}$
17:
18: **Actor update:**
19: **if** $t \mod d$ **then**
20: Freeze $Q_\psi, \overline{Q}_{\overline{\psi}}, f_\phi, g_\eta$, and E_μ
21: Compute policy loss using DPG and HPG: $\mathcal{L}_{\text{actor}}(\theta)$ ▷ Equation (11)
22: Update policy: $\theta \leftarrow \arg\min \mathcal{L}_{\text{actor}}(\theta)$
23: Update target networks $\psi' \leftarrow \alpha\psi + (1-\alpha)\psi',\ \overline{\psi}' \leftarrow \alpha\overline{\psi} + (1-\alpha)\overline{\psi}',\ \theta' \leftarrow \alpha\theta + (1-\alpha)\theta'$
24: **end if**
25: **end for**

E.2 Hyperparameters

Our code is submitted in the suplemental material.

We implemented our method in PyTorch [65] and results were obtained using Python v3.8.10, PyTorch v1.10.0, CUDA 11.4, and Mujoco 2.1.1 [86] on A100 GPUs on a cloud computing service. Tables 1-3 present the hyperparameters used in our experiments. The hyperparameters are all adapted from DrQ-v2 [95] *without any further hyperparameter tuning*. We have kept the same set of hyperparameters

across all algorithms and tasks, except for the walker domain which similarly to DrQ-v2 [95], we used n-step return of $n = 1$ and mini-batch size of 512.

The core RL components (actor and critic networks), as well as the components of DHPG (state and action encoders, transition and reward models) are all MLP networks with the ReLU activation function and one hidden layer with dimension of 256.

In the case of state observations, the abstract MDP has the same state and action dimensions as the actual MDP. In the case of pixel observations, the image encoder is based on the architecture of DrQ-v2 which is itself based on SAC-AE [97] and consists of four convolutional layers of $32 \times 3 \times 3$ with ReLU as their activation functions, followed by a one-layer fully-connected neural network with layer normalization [7] and tanh activation function. The stride of the convolutional layers are 1, except for the first layer which has stride 2. The image decoder of the baseline models with image reconstruction is based on SAC-AE [97] and has a single-layer fully connected neural network followed by four transpose convolutional layers of $32 \times 32 \times 3$ with ReLU activation function. The stride of the transpose convolutional layers are 1, except for the last layer which has stride 2.

Table 1: Hyperparameters used in our experiments.

Hyperparameter	**Setting**
Learning rate	$1\text{e}{-4}$
Optimizer	Adam
n-step return	3
Mini-batch size	256
Actor update frequency d	2
Target networks update frequency	2
Target networks soft-update τ	0.01
Target policy smoothing stddev. clip c	0.3
Hidden dim.	256
Replay buffer capacity	10^6
Discount γ	0.99
Seed frames	4000
Exploration steps	2000
Exploration stddev. schedule	$\text{linear}(1.0, 0.1, 1\text{e}6)$

Table 2: Hyperparameters specific to state observations.

Hyperparameter	**Setting**
Feature dim.	Same as the state dim. of the task
Action repeat	1
Frame stack	N/A

Table 3: Hyperparameters specific to pixel observations.

Hyperparameter	**Setting**
Feature dim.	50
Action repeat	2
Frame stack	3

E.3 Baseline Implementations

All of the baselines are submitted in the supplemental material. We use the official implementations of DBC, SAC-AE, and TD3. DeepMDP does not have a publicly available code, and we use the implementation available in the official DBC code-base. The official DDPG implementation is in TensorFlow, thus we used the implementation available in the official TD3 code-base with additional improvements detailed in Section 7.1. Similarly, the official SAC implementation is in TensorFlow, thus we used the SAC implementation available in the official SAC-AE code-base. As discussed in Section 7.2, we have run two versions of the baselines, with and without image augmentation. The image augmented variants, use the same image augmentation method of DrQ-v2 described in Appendix E.1.