Plannable Approximations to MDP Homomorphisms: Equivariance under Actions

van der Pol, Elise; Kipf, Thomas; Oliehoek, Frans A.; Welling, Max

Publication date
2020

Document Version
Final published version

Published in
Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2020

Citation (APA)
van der Pol, E., Kipf, T., Oliehoek, F. A., & Welling, M. (2020). Plannable Approximations to MDP Homomorphisms: Equivariance under Actions. In B. An, A. El Fallah Seghrouchni, & G. Sukthankar (Eds.), *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2020* (pp. 1431–1439). (Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS; Vol. 2020-May). International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS).

Plannable Approximations to MDP Homomorphisms: Equivariance under Actions

Elise van der Pol
UvA-Bosch Deltalab, University of Amsterdam
e.e.vanderpol@uva.nl

Thomas Kipf
University of Amsterdam
t.n.kipf@uva.nl

Frans A. Oliehoek
Delft University of Technology
f.a.oliehoek@tudelft.nl

Max Welling
UvA-Bosch Deltalab, University of Amsterdam
m.welling@uva.nl

ABSTRACT

This work exploits action equivariance for representation learning in reinforcement learning. Equivariance under actions states that transitions in the input space are mirrored by equivalent transitions in latent space, while the map and transition functions should also commute. We introduce a contrastive loss function that enforces action equivariance on the learned representations. We prove that when our loss is zero, we have a homomorphism of a deterministic Markov Decision Process (MDP). Learning equivariant maps leads to structured latent spaces, allowing us to build a model on which we plan through value iteration. We show experimentally that for deterministic MDPs, the optimal policy in the abstract MDP can be successfully lifted to the original MDP. Moreover, the approach easily adapts to changes in the goal states. Empirically, we show that in such MDPs, we obtain better representations in fewer epochs compared to representation learning approaches using reconstructions, while generalizing better to new goals than model-free approaches.

KEYWORDS

MDPs; Equivariance; Planning; MDP Homomorphisms

ACM Reference Format:
Elise van der Pol, Thomas Kipf, Frans A. Oliehoek, and Max Welling. 2020. Plannable Approximations to MDP Homomorphisms: Equivariance under Actions. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

Dealing with high dimensional state spaces and unknown environmental dynamics presents an open problem in decision making [19]. Classical dynamic programming approaches require knowledge of environmental dynamics and low dimensional, tabular state spaces [41]. Recent deep reinforcement learning methods on the other hand offer good performance, but often at the cost of being unstable and sample-hungry [19, 22, 36, 37]. The deep model-based reinforcement learning literature aims to fill this gap, for example by finding policies after learning models based on input reconstruction [7, 16, 32, 54], by using environmental models in auxiliary losses [9, 22], or by forcing network architectures to resemble planning algorithms [39, 45]. While effective in learning end-to-end policies, these types of approaches are not forced to learn good representations and may thus not build proper environmental models. In this work, we focus on learning representations of the world that are suitable for exact planning methods. To combine dynamic programming with the representational power of deep networks, we factorize the online decision-making problem into a self-supervised model learning stage and a dynamic programming stage. We do this under the assumption that good representations minimize MDP metrics [10, 15, 35, 46]. While such metrics have desirable theoretical guarantees, they require an enumerable state space and knowledge of the environmental dynamics, and are thus not usable in many problems. To resolve this issue, we propose to learn representations using the more flexible notion of action equivariant mappings, where the effects of actions in input space are matched by equivalent action effects in the latent space. See Figure 1. We make the following contributions. First, we propose learning an equivariant map and corresponding action embeddings. This corresponds to using MDP homomorphism [43] metrics [46] of deterministic MDPs, enabling planning in the homomorphic image of the original MDP. Second, we prove that for deterministic MDPs, when our loss is zero, we have an MDP homomorphism. This means that the resulting policy can be lifted to the original MDP. Third, we provide experimental evaluation in a variety of settings to show 1) that we can recover the graph structure of the input MDP, 2) that planning in this abstract space results in good policies for the original space, 3) that we can change to arbitrary new goal states without further gradient descent updates and 4) that this works even when the input states are continuous, or when generalizing to new instances with the same dynamics.

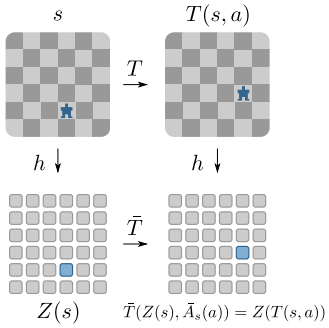


Figure 1: Visualization of the notion of equivariance under actions. We say Z is an action equivariant mapping if $Z(T(s,a)) = \bar{T}(Z(s), \bar{A}_s(a))$.

TK is now at Google Research, Brain Team.

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

2 BACKGROUND

Markov Decision Processes. An infinite horizon Markov Decision Process (MDP) is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, T, \gamma)$, where $s \in \mathcal{S}$ is a Markov state, $a \in \mathcal{A}$ is an action that an agent can take, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function that returns a scalar signal r defining the desirability of some observed transition, $0 \leq \gamma \leq 1$ is a discount factor that discounts future rewards exponentially and $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition function, that for a pair of states and an action assigns a probability of transitioning from the first to the second state. The goal of an agent in an MDP is to find a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, a function assigning probabilities to actions in states, that maximizes the *return* $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$. The expected return of a state, action pair under a policy π is given by a Q-value function $Q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ where $Q_\pi(s, a) = \mathbb{E}_\pi [G_t | s_t = s, a_t = a]$. The value of a state under an optimal policy π^* is given by the value function $V^* : \mathcal{S} \to \mathbb{R}$, defined as $V^* = \max_a Q^*(s, a)$ under the Bellman optimality equation.

Value Iteration. Value Iteration (VI) is a dynamic programming algorithm that finds Q-values in MDPs, by iteratively applying the Bellman optimality operator. This can be viewed as a graph diffusion where each state is a vertex and transition probabilities define weighted edges. VI is guaranteed to find the optimal policy in an MDP. For more details, see [41].

Bisimulation Metrics. To enable computing optimal policies in MDPs with very large or continuous state spaces, one approach is aggregating states based on their similarity in terms of environmental dynamics [8, 35]. A key concept is the notion of *stochastic bisimulations* for MDPs, which was first introduced by Dean and Givan [8]. Stochastic bisimulation defines an equivalence relation on MDP states based on matching reward and transition functions, allowing states to be compared to each other. Later work [10] observes that the notion of stochastic bisimulation is too stringent (everything must match exactly) and proposes using a more general *bisimulation metric* instead, with the general form

$$d(s, s') = \max_a \Big(c_R |R(s, a) - R(s', a)| + c_T d_P(T(s, a), T(s', a)) \Big) \tag{1}$$

where c_R and c_T are weighting constants, $T(\cdot, a)$ is a distribution over next states and d_P is some probability metric, such as the Kantorovich (Wasserstein) metric. Such probability metrics are recursively computed. For more details, see [10]. The bisimulation metric provides a distance between states that is not based on input features but on environmental dynamics.

MDP Homomorphism. A generalization of the mapping induced by bisimulations is the notion of MDP homomorphisms [43]. MDP homomorphisms were introduced by [42] as an extension of [8]. An MDP homomorphism h is a tuple of functions $\left\langle Z, \left\{ \bar{A}_s \right\} \right\rangle$ with $Z : \mathcal{S} \to \mathcal{Z}$ a function that maps states to abstract states, and each $\bar{A}_s : \mathcal{A} \to \bar{\mathcal{A}}$ a state-dependent function that maps actions to abstract actions, that preserves the structure of the input MDP. We use the definition given by Ravindran and Barto [43]:

Definition 2.1 (Stochastic MDP Homomorphism). A *Stochastic MDP homomorphism* from a stochastic MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R \rangle$ to an MDP $\bar{\mathcal{M}} = \left\langle \mathcal{Z}, \bar{\mathcal{A}}, \bar{T}, \bar{R} \right\rangle$ is a tuple $h = \left\langle Z, \left\{ \bar{A}_s \right\} \right\rangle$, with

- $Z : \mathcal{S} \to \mathcal{Z}$ the state embedding function, and
- $\bar{A}_s : \mathcal{A} \to \bar{\mathcal{A}}$ the action embedding functions,

such that the following identities hold:

$$\forall_{s, s' \in \mathcal{S}, a \in \mathcal{A}} \quad \bar{T}(Z(s') | Z(s), \bar{A}_s(a)) = \sum_{s'' \in [s']_Z} T(s'' | s, a) \tag{2}$$

$$\forall_{s \in \mathcal{S}, a \in \mathcal{A}} \quad \bar{R}(Z(s), \bar{A}_s(a)) = R(s, a) \tag{3}$$

Here, $[s']_Z = Z^{-1}(Z(s'))$ is the equivalence class of s' under Z.

We specifically consider deterministic MDPs. In that case:

Definition 2.2 (Deterministic MDP Homomorphism). A *Deterministic MDP homomorphism* from a deterministic MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R \rangle$ to an MDP $\bar{\mathcal{M}} = \left\langle \mathcal{Z}, \bar{\mathcal{A}}, \bar{T}, \bar{R} \right\rangle$ is a tuple $h = \left\langle Z, \left\{ \bar{A}_s \right\} \right\rangle$, with

- $Z : \mathcal{S} \to \mathcal{Z}$ the state embedding function, and
- $\bar{A}_s : \mathcal{A} \to \bar{\mathcal{A}}$ the action embedding functions,

such that the following identities hold:

$$\forall_{s, s' \in \mathcal{S}, a \in \mathcal{A}} \quad T(s, a) = s' \implies \bar{T}(Z(s), \bar{A}_s(a)) = Z(s') \tag{4}$$

$$\forall_{s \in \mathcal{S}, a \in \mathcal{A}} \quad \bar{R}(Z(s), \bar{A}_s(a)) = R(s, a) \tag{5}$$

The states s are organized into equivalence classes under Z if they follow the same dynamics in z-space. The MDP $\bar{\mathcal{M}}$ is referred to as the *homomorphic image* of $\mathcal{M}$ under h [43]. An important property of MDP homomorphisms is that a policy optimal in homomorphic image $\bar{\mathcal{M}}$ can be *lifted* to an optimal policy in $\mathcal{M}$ [18, 43]. Looking at these definitions, it may be clear that MDP homomorphisms and bisimulation metrics are closely related. The difference is that the latter measures distances between two MDP states, while the former is a map from one MDP to another. However, the idea of forming a distance metric by taking a sum of the distances can be extended to homomorphisms, as proposed by Taylor et al. [46]:

$$\begin{aligned} d((s, a), (Z(s), \bar{A}_s(a))) = c_R |R(s, a) - \bar{R}(Z(s), \bar{A}_s(a))| \\ + c_T d_P(ZT(s, a), \bar{T}(Z(s), \bar{A}_s(a))), \end{aligned} \tag{6}$$

with d_P a suitable measure of the difference between distributions (e.g., Kantorovich metric), and $ZT(s, a)$ shorthand for projecting the distribution over next states into the space of $\mathcal{Z}$ (see [13] for details). We refer to this as the *MDP homomorphism metric*.

Action-Equivariance. We define a mapping $Z : \mathcal{S} \to \mathcal{Z}$ to be action-equivariant if $Z(T(s, a)) = \bar{T}(Z(s), \bar{A}_s(a))$ and $R(s, a) = \bar{R}(Z(s), \bar{A}_s(a))$, i.e. when the constraints in Eq. 4 and Eq. 5 hold.

3 LEARNING MDP HOMOMORPHISMS

We are interested in learning compact, plannable representations of MDPs. We call MDP representations *plannable* if the optimal policy found by planning algorithms such as VI can be lifted to the original MDP and still be close to optimal. This is the case when the representation respects the original MDP's dynamics, such as when the equivariance constraints in Eq. 4 and Eq. 5 hold. In this paper we leverage MDP homomorphism metrics to find such representations. In particular, we introduce a loss function that enforces these equivariance constraints, then construct an abstract MDP in the learned representation space. We compute a policy in the abstract MDP $\bar{\mathcal{M}}$ using VI, and *lift* the abstract policy to the

original space. To keep things simple, we focus on deterministic MDPs, but in preliminary experiments our method performed well out of the box on stochastic MDPs. Additionally, the framework we outline here can be extended to the stochastic case, as Gelada et al. [13] does for bisimulation metrics.

3.1 Learning State Representations

Here we show how to learn state representations that respect action-equivariance. We embed the states in $\mathcal{S}$ into Euclidean space using a contrastive loss based on MDP homomorphism metrics. Similar losses have often been used in related work [2, 11, 13, 30, 40], which we compare in Section 5. We represent the mapping Z using a neural network parameterized by θ, whose output will be denoted Z_θ. This function maps a state $s \in \mathcal{S}$ to a latent representation $z \in \mathcal{Z} \subseteq \mathbb{R}^D$. We additionally approximate the abstract transition $\bar{T}$ by a function $\bar{T}_\phi : \mathcal{Z} \times \bar{\mathcal{A}} \to \mathcal{Z}$ parameterized by ϕ, and the abstract rewards $\bar{R}$ by a neural network $\bar{R}_\zeta : \mathcal{Z} \to \mathbb{R}$, parameterized by ζ, that predicts the reward for an abstract state. From Eq. 5 we simplify to a state-dependent reward using $R(s) = \bar{R}(Z(s))$ where $R(s)$ is the reward function that outputs a scalar value for an $s \in \mathcal{S}$, and $\bar{R}$ is its equivalent in $\bar{\mathcal{M}}$. During training, we first sample a set of *experience tuples* $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}_{n=1}^N$ by rolling out an exploration policy π_e for K trajectories. To learn representations that respect Eq. 4 and 5, we minimize the distance between the result of transitioning in observation space, and then mapping to $\mathcal{Z}$, or first mapping to $\mathcal{Z}$ and then transitioning in latent space (see Figure 1). Additionally, the distance between the observed reward $R(s)$ and the predicted reward $\bar{R}_\zeta(Z_\theta(s))$ is minimized. We thus include a general reward loss term. We write $s'_n = T(s_n, a_n)$, $z_n = Z_\theta(s_n)$, and minimize

$$\mathcal{L}(\theta, \phi, \zeta) = \frac{1}{N} \sum_{n=1}^{N} \Big[d\left(Z_\theta(s'_n), \bar{T}_\phi(z_n, \bar{A}_\phi(z_n, a_n))\right) + d\left(R(s_n), \bar{R}_\zeta(z_n)\right) \Big] \tag{7}$$

by randomly sampling batches of experience tuples from $\mathcal{D}$. In this paper, we use $d(z, z') = \frac{1}{2}(z - z')^2$ to model distances in $\mathcal{Z} \subseteq \mathbb{R}^D$. Here, $\bar{T}_\phi$ is a function that maps a point in latent space $z \in \mathcal{Z}$ to a new state $z' \in \mathcal{Z}$ by predicting an *action-effect* that acts on z. We adopt earlier approaches of letting $\bar{T}_\phi$ be of the form $\bar{T}_\phi(z, \bar{a}) = z + \bar{A}_\phi(z, a)$, where $\bar{A}_\phi(z, a)$ is a simple feedforward network [11, 30]. Thus $\bar{A}_\phi : \mathcal{Z} \times \mathcal{A} \to \bar{\mathcal{A}}$ is a function mapping from the original action space to an abstract action space, and $\bar{A}_\phi(z, a)$ approximates $\bar{A}_s(a)$ (Eq. 4). The resulting transition loss is a variant of the loss proposed in [30]. The function $\bar{R}_\zeta : \mathcal{Z} \to \mathbb{R}$ predicts the reward from z. Since Z, $\bar{T}$ and $\bar{R}$ are neural networks optimized with SGD, Eq. 7 has a trivial solution where all states are mapped to the same point, especially in the sparse reward case. When the reward function is informative, minimizing Eq. 7 can suffice, as is empirically demonstrated in [13]. However, when rewards are sparse, the representations may collapse to the trivial embedding, and for more complex tasks [13] requires a pixel reconstruction term. In practice, earlier works use a variety of solutions to prevent the trivial map. Approaches based on pixel reconstructions are common [7, 13, 16, 17, 25, 33, 49, 50, 54], but there are also approaches based on self-supervision that use alternatives to reconstruction of input states [1, 2, 4, 11, 30, 40, 53].

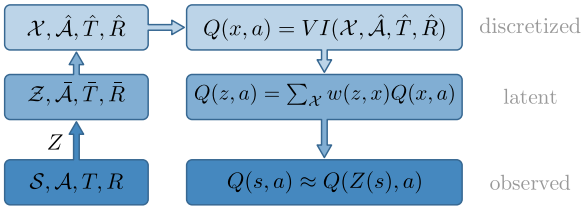


Figure 2: Schematic overview of our method. We learn the map Z from $\mathcal{S}$ to $\mathcal{Z}$ and discretize $\mathcal{Z}$ to obtain $\mathcal{X}$. We plan in $\mathcal{X}$ and use interpolated Q-values to obtain a policy in $\mathcal{S}$.

To prevent trivial solutions, we use a contrastive loss, maximizing the distance between the latent next state and the embeddings of a set of random other states, $\mathcal{S}_\neg = \{s_j\}_{j=1}^J$ sampled from the same trajectory on every epoch. Thus, the complete loss is

$$\mathcal{L}(\theta, \phi, \zeta) = \frac{1}{N} \sum_{n=1}^{N} \Big[d\left(Z_\theta(s'_n), \bar{T}_\phi(z_n, \bar{A}_\phi(z_n, a_n))\right) + d\left(R(s_n), \bar{R}_\zeta(z_n)\right) + \sum_{s_\neg \in \mathcal{S}_\neg} d_\neg\left(Z_\theta(s_\neg), \bar{T}_\phi(z_n, \bar{A}_\phi(z_n, a_n))\right) \Big] \tag{8}$$

where $d_\neg$ is a negative distance function. Similar to [30], we use the hinge loss $d_\neg(z, z') = \max(0, \epsilon - d(z, z'))$ to prevent the negative distance from growing indefinitely. Here, ϵ is a parameter that controls the scale of the embeddings. To limit the scope of this paper, we consider domains where we can find a reasonable data set of transitions without considering exploration. Changing the sampling policy will introduce bias in the data set, influencing the representations. Here we evaluate if we can find plannable MDP homomorphisms and leave the exploration problem to future work.

3.2 Constructing the Abstract MDP

After learning a structured latent space, we find abstract MDP $\bar{\mathcal{M}}$ by constructing reward and transition functions from Z_θ, $\bar{T}_\phi$, $\bar{R}_\zeta$.

3.2.1 Abstract States. Core to our approach is the idea that exploiting action-equivariance constraints leads to nicely structured abstract spaces that can be planned in. Of course the space $\mathcal{Z}$ is still continuous, which requires either more complex planning methods, or state discretization. In this paper we aim for the latter, simpler, option, by constructing a discrete set $\mathcal{X}$ of ('prototype') latent states in $\mathcal{Z}$ over which we can perform standard dynamic programming techniques. We will denote such prototype states as $x \in \mathcal{X}$, cf. Figure 2. Of course, we then also need to construct discrete transition $\hat{T}_\phi$ and reward $\hat{R}_\zeta$ functions. The next sub-sections will outline methods to obtain these from $\mathcal{Z}$, $\bar{T}_\phi$ and $\bar{R}_\zeta$. To find a 'plannable' set of states, the abstract state space should be sufficiently covered. To construct the set, we sample L states from the replay memory and encode them, i.e. $\mathcal{X} = \{Z_\theta(s_l) | s_l \sim \mathcal{D}\}_{l=1}^L$, pruning duplicates.

3.2.2 Reward Function. In Eq. 8 we use a reward prediction loss to encourage the latent states to contain information about

the rewards. This helps separate distinct states with comparable transition functions. During planning, we can use this predicted reward $\bar{R}_\zeta$. When the reward depends on a changing goal state, such as in the goal-conditioned tasks in Section 4, $\hat{R}_\zeta(Z_\theta(s)) = 1$ if $Z_\theta(s) = Z_\theta(s_g)$ and 0 otherwise. We use this reward function in planning, i.e. $\hat{R}_\zeta(x) = 1$ if $x = Z_\theta(s_g)$ and 0 otherwise.

3.2.3 Transition Function. We model the transitions on the basis of similarity in the abstract space. We follow earlier work [12, 29] and assume that if two states are connected by an action in the state space, they should be close after applying the latent action transition. The transition function is a distribution over next latent states. Therefore, we use a temperature softmax to model transition probabilities between representations of abstract states in $\mathcal{X}$:

$$\hat{T}'_\phi(z_j|z_i,\alpha) = \frac{e^{-d(z_j, z_i+\bar{A}_\phi(z_i,\alpha))/\tau}}{\sum_{k\in\mathcal{X}} e^{-d(z_k, z_i+\bar{A}_\phi(z_i,\alpha))/\tau}} \tag{9}$$

Thus, for the transitions between abstract states:

$$\hat{T}_\phi(x = j|x' = i, \hat{a} = \alpha) = \hat{T}'_\phi(z_j|z_i,\alpha) \tag{10}$$

where τ is a temperature parameter that determines how 'soft' the edges are, and z_j is the representation of abstract state j. Intuitively, this means that if an action moves two states closer together, the weight of their connection increases, and if it moves two states away from each other, the weight of their connection decreases. For very small τ, the transitions are deterministic.

3.2.4 Convergence to an MDP homomorphism. We now show that when combining optimization of our proposed loss fuction (8) with the construction of an abstract MDP as detailed in this subsection, we can approximate an MDP homomorphism. Specifically, for deterministic MDPs, we show that when the loss function in Eq. 8 reaches zero, we have an MDP homomorphism of $\mathcal{M}$.

Theorem 3.1. *In a deterministic MDP $\mathcal{M}$, assuming a training set that contains all state, action pairs, and an exhaustively sampled set of abstract states $\mathcal{X}$ we consider a sequence of losses in a successful training run, i.e. the losses converge to 0. In the limit of the loss $\mathcal{L}$ in Eq. 8 approaching 0, i.e. $\mathcal{L} \to 0$ and $0 < \tau \ll 1$, $\tau \ll \epsilon$, $h = (Z_\theta, \bar{A}_\phi)$ is an MDP homomorphism of $\mathcal{M}$.*

Proof. Fix $0 < \tau \ll 1$ and write $z = Z_\theta(s)$ and $\bar{a} = \bar{A}_\phi(z, a)$. Consider that learning converges, i.e. $\mathcal{L} \to 0$. This implies that the individual loss terms $d(\bar{T}_\phi(z,\bar{a}), z')$, $d_\neg(\bar{T}_\phi(z,\bar{a}), z_\neg)$ and $d(R(s), \bar{R}_\zeta(z))$ also go to zero for all $(s, a, r, s', s_\neg) \sim \mathcal{D}$.

Positive samples: As the distance for positive samples $d_+ = d(\bar{T}_\phi(z,\bar{a}), z') \to 0$, then $d_+ \ll \tau$. Since $d_+ \ll \tau$, then $e^{-d_+/\tau} \approx 1$.

Negative samples: Because the negative distance $d_\neg(\bar{T}_\phi(z,\bar{a}), z_\neg) \to 0$, $d_\neg \leq \epsilon$. This, in turn, implies that the distance to all negative samples $d_- = d(\bar{T}_\phi(z,\bar{a}), z_\neg) \geq \epsilon$ and thus $\tau \ll \epsilon \leq d_-$, meaning that $1 \ll \frac{d_-}{\tau}$ and thus $e^{-d_-/\tau} \approx 0$.

This means that when the loss approaches 0, $\hat{T}'_\phi(z'|z,\bar{a}) = 1$ where $T(s'|s,a) = 1$ and $\hat{T}'_\phi(z_\neg|z,\bar{a}) = 0$ when $T(s_\neg|s,a) = 0$. Since $\mathcal{M}$ is deterministic, $T(s'|s,a)$ transitions to one state with probability 1, and probability 0 for the others. Therefore, $\hat{T}'_\phi(Z_\theta(s')|Z_\theta(s), \bar{A}_\phi(Z_\theta(s),a)) = \sum_{s''\in[s']_\mathcal{Z}} T(s''|s,a)$ and Eq. 4 holds. As the distance for rewards $d(R(s), \bar{R}_\zeta(z)) \to 0$, we have that $\bar{R}_\zeta(z) = R(s)$ and Eq. 5 holds. Therefore, when the loss reaches zero we have an MDP homomorphism of $\mathcal{M}$. □

Note that Eq. 8 will not completely reach zero: negative samples are drawn uniformly. Thus, a positive sample may occasionally be treated as a negative sample. Refining the negative sampling can further improve this approach.

3.3 Planning and Acting

After constructing the abstract MDP we plan with VI [41] and lift the found policy to the original space by interpolating between Q-value embeddings. Given $\hat{\mathcal{M}} = (\mathcal{X}, \hat{\mathcal{A}}, \hat{T}_\phi, \hat{R}_\zeta, \gamma)$, VI finds a policy $\hat{\pi}$ that is optimal in $\hat{\mathcal{M}}$. For a new state $s^* \in \mathcal{S}$, we embed it in the representation space $\mathcal{Z}$ as $z^* = Z_\theta(s^*)$ and use a softmax over its distance to each $x \in \mathcal{X}$ to interpolate between their Q-values, i.e.

$$Q(z^*, a) = \sum_{x\in\mathcal{X}} w(z^*, x) Q(x, a) \tag{11}$$

$$w(z^*, x) = \frac{e^{-d(z_x, z^*)/\eta}}{\sum_{k\in\mathcal{X}} e^{-d(z_k, z^*)/\eta}} \tag{12}$$

where η is a temperature parameter that sets the 'softness' of the interpolation. We use the interpolated Q-values for greedy action selection for s^*, transition to s^{**} and iterate until the episode ends.

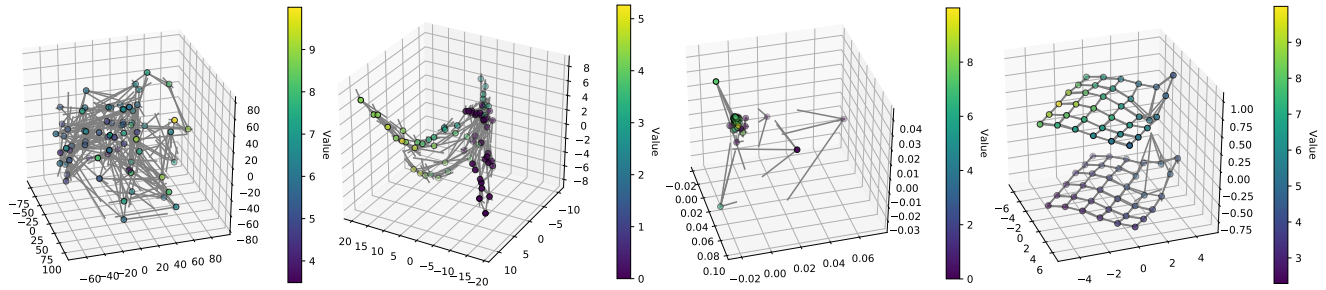
4 EXPERIMENTS

Here we show that in simple domains, our approach 1) succeeds at finding plannable MDP homomorphisms for discrete and continuous problems 2) requires less data than model-free approaches, 3) generalizes to new reward functions and data and 4) trains faster than approaches based on reconstructions. We focus on deterministic MDPs. While preliminary results on stochastic domains were promising, an in-depth discussion is beyond the scope of this paper.

4.1 Baselines

To evaluate our approach, we compare to a number of baselines:

(1) WM-AE: An auto-encoder approach inspired by World Models [16]. We follow their approach of training representations using a reconstruction loss, then learning latent dynamics on fixed representations. We experimented with a VAE [28], which did not perform well (see [30] for similar results). We thus use an auto-encoder to learn an embedding, then train an MLP to predict the next state from embedding and action.
(2) LD-AE: An auto-encoder with latent dynamics. We train an auto-encoder to reconstruct the input, and predict the next latent state. We experimented with reconstructing the next state, but this resulted in the model placing the next state embeddings in a different location than the latent transitions.
(3) DMDP-H: We evaluate the effectiveness of training without negative sampling. This is similar to DeepMDP [13]. However, unlike DeepMDP, DMDP-H uses action-embeddings, for a fairer comparison.
(4) GC-Model-Free: Finally, we compare to a goal-conditioned model-free baseline (REINFORCE with state-value baseline),



(a) WM-AE Baseline (b) LD-AE Baseline (c) DMDP-H Baseline (d) This paper

Figure 3: Abstract MDP for three approaches in the single object room domain. Nodes are PCA projections of abstract states, edges are predicted $\bar{T}_\phi$, colors are predicted values.

to contrast our approach with directly optimizing the policy[1]. We include the goal state as input for a fair comparison.

To fairly compare the planning approaches, we perform a grid search over the softness of the transition function by evaluating performance on the train goals in $\tau \in [1, 0.1, 0.001, 0.0001, 0.00001, 1e-20]$. Unless otherwise stated, the planning approaches are all trained on datasets of 1000 trajectories, sampled with a random policy. The learning rate is set to 0.001 and we use Adam [27]. For the hinge loss, we use $\epsilon = 1$. The latent dimensionality is set to 50 everywhere. Our approach is trained for 100 epochs. WM-AE is trained for 1000 epochs in total: 500 for the auto-encoder and 500 for the dynamics. LD-AE is trained for 1000 epochs. For constructing the abstract MDP we sample 1024 states from $\mathcal{D}$, project unto $\mathcal{Z}$ and prune duplicates. For planning we use VI with discount factor $\gamma = 0.9$, 500 backups and interpolation parameter (Eq. 12) $\eta = 1e-20$. The learning rate for the model-free baseline was chosen by fine-tuning on the training goals. For the model-free baseline, we use a learning rate of $5e-4$ and we train for 500k steps (more than five times the number of samples the planning approaches use). Network Z_θ has 2 convolutional layers (both 16 channels, 3×3 filters) and 3 fully connected layers (input$\to 64 \to 32 \to |z|$). Networks T_ϕ and R_ξ each have 2 fully connected layers. We use ReLU non-linearities between layers.

4.2 Object Collection

We test our approach on an object collection task inspired by the key task in [11], with major differences: rather than searching for three keys in a labyrinth, the agent is placed in a room with some objects. Its task is to collect the key. On every time step, the agent receives a state—a $3 \times 48 \times 48$ pixel image (a channel per object, including the agent), as shown in Figure 4—and a goal state of the same size. At train time, the agent receives reward of 1 on collection of the *key* object, and a reward of -1 if it grabs the wrong object, and a reward of -0.1 on every time step. The episode ends if the agent picks up one (or more) of the objects and delivers it to one of the four corners (randomly sampled at episode start), receiving an additional delivery reward of 1. At test time, the agent is tasked with retrieving one of the objects chosen at random, and delivering to a randomly chosen location, encoded as a desired goal state. This task will evaluate how easily the trained agent adapts to new goals/reward functions. The agent can interact with the environment until it reaches the goal or 100 steps have passed. For both tasks, we compare to the model-free baseline. We also compare to the DMDP-H, WD-AE and LD-AE baselines. We additionally perform a grid search over the hinge, number of state samples for discretization and η hyperparameters for insight in how these influence the performance. This showed that our approach is robust with respect to the hinge parameter, but it influences the scale of the embeddings. The results decrease only when using 256 or fewer state samples. Lastly, η is robust for values lower than 1. We opt for a low value of η, to assign most weight to the Q-value of the closest state.

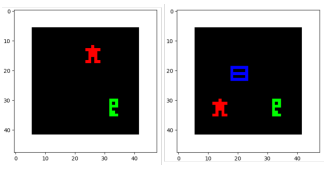


Figure 4: Example states in the object collection domain for the single object and double object tasks.

4.2.1 Single Object Task. We first evaluate a simple task with only one object (a key). The agent's task is to retrieve the key, and move to one of four delivery locations in the corners of the room. The delivery location is knowledge supplied to the agent in the form of a goal state that places the agent in the correct corner and shows that there is no key. These goal states are also supplied to the baseline, during training and testing. Additionally, we perform an ablation study on the effect of the reward loss. The average episode lengths are shown in Table 1. Our approach outperforms all baselines, both at train and at test time. There is no clear preference in terms of the number of negative samples — as long as $J > 0$ — the result for all values of J are quite close together. The DMDP-H approach fails to find a reasonable policy, possibly due to the sparse rewards in this task providing little pull against state collapse. Out of the planning baselines, WM-AE performs best, probably because visually salient features are aligned with decision making features in this task. Finally, the model-free approach is the best performing baseline on the training goals, but does not generalize to test goals. The results of the reward ablation are shown in Table 2. While

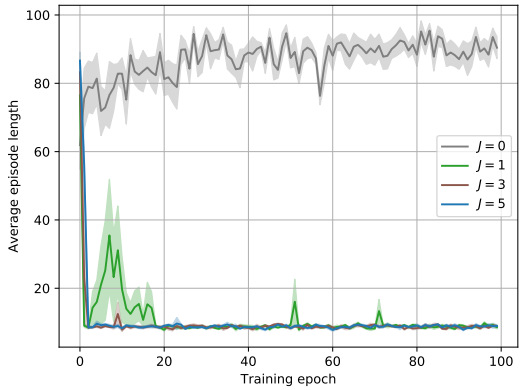
[1]Deep reinforcement learning algorithms such as our baseline may fail catastrophically depending on the random seed [19]. For a fair comparison, we train the baseline on 6 random seeds, then remove those seeds where the method fails to converge for the train setting.

Avg. ep. length ↓				
Task	**Single Object**		**Double Object**	
Goal Set	Train	Test	Train	Test
GC-Model-free	10.00 ± 0.11	67.25 ± 6.81	10.10 ± 0.69	38.25 ± 15.30
WM-AE	12.96 ± 8.93	10.03 ± 5.56	29.61 ± 19.42	22.53 ± 22.12
LD-AE	23.46 ± 27.10	21.04 ± 21.71	60.26 ± 29.14	52.72 ± 27.32
DMDP-H ($J = 0$)	82.88 ± 11.62	85.69 ± 7.98	81.24 ± 2.45	81.17 ± 2.69
Ours, $J = 1$,	8.61 ± 0.35	**7.53** ± 0.24	8.53 ± 0.36	**8.38** ± 0.07
Ours, $J = 3$	8.68 ± 0.27	7.63 ± 0.19	8.61 ± 0.38	8.95 ± 0.63
Ours, $J = 5$	**8.57** ± 0.48	7.74 ± 0.22	**8.26** ± 0.84	8.96 ± 1.15

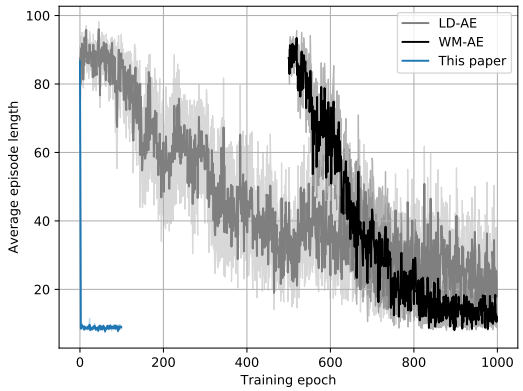
Table 1: Comparing average episode length of 100 episodes on the object collection domain. Reporting mean and standard deviation over 5 random seeds for the planning approaches. The model free approach is averaged over 4 random seeds for the single object domain, 3 random seeds for the double object domain.

Avg. ep. length ↓				
	Reward Loss		**No Reward Loss**	
Goal Set	Train	Test	Train	Test
DMDP-H ($J = 0$)	82.88 ± 11.62	85.69 ± 7.98	87.03 ± 3.08	84.08 ± 3.02
Ours, $J = 1$	8.61 ± 0.35	7.53 ± 0.24	74.32 ± 19.90	68.54 ± 17.29
Ours, $J = 3$	8.68 ± 0.27	7.63 ± 0.19	8.54 ± 0.36	7.44 ± 0.21
Ours, $J = 5$	8.57 ± 0.48	7.74 ± 0.22	8.52 ± 0.19	7.53 ± 0.20

Table 2: Ablation study of the effect of the reward loss. Comparing average episode length of 100 episodes for the single object room domain. Reporting mean and standard deviation over 5 random seeds.



(a) Comparison of different values of J.



(b) Comparison of this paper and the WM-AE and LD-AE baselines. WM-AE can not be evaluated until the auto-encoder has finished training and training of the dynamics model begins.

Figure 5: Average episode length per training epoch for the single object domain. Reported mean and standard error over 5 random seeds.

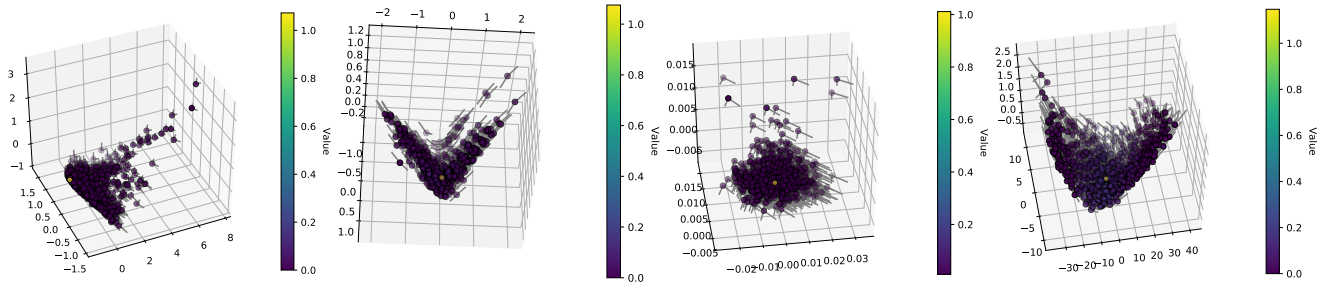
removing the reward loss does not influence performance much for $J = 0$, $J = 3$ and $J = 5$, when $J = 1$ the reward prediction is needed to separate the states. Without the reward, the single negative sample does not provide enough pull for complete separation.

We show the latent spaces found for the baselines and our approach in Figure 3. Our approach has found a double grid structure - representing the grid world before, and after picking up the key. The baselines are reasonably plannable after training for long enough, but the latent spaces aren't as nicely structured as our approach. This mirrors results in earlier work [30]. Thus, while pixel reconstruction losses may be able to find reasonable representations for certain problems, these rely on arbitrarily complex transition functions. Moreover, due to their need to train a pixel reconstruction loss they take much longer to find useable representations. This is shown in Figure 5b, where the performance after planning for each training epoch is plotted and compared. Additionally, we observe state collapse for DMDP-H in Figure 3c, and this is reflected in a high average episode length after planning.

4.2.2 Double Object Task. We now extend the task to two objects: a key and an envelope. The agent's task at train time is still to retrieve the key. At test time, the agent has to pick up the key or the envelope (randomly chosen) and deliver it to one of the corners. We show results in Table 1. Again, our method performs well on both train and test set, having clearly learned a useful abstract representation, that generalizes to new goals. The WM-AE baseline again fares better than the LD-AE baseline, and DMDP-H fails to find a plannable representation. The model-free baseline performs slightly worse than our method on this task, even after seeing much more data. Additionally, even though it performs reasonably well on the training goals, it does not generalize to new goals at all. The WM-AE performs worse on this task than our approach, but generalizes much better than the model-free baseline, due to its planning, while the LD-AE baseline does not find plannable representations of this task.

4.3 Continuous State Spaces

We evaluate whether we can use our method to learn plannable representations for continuous state spaces. We use OpenAI's CartPole-v0 environment [5]. We include again a model-free baseline that is trained until completion as a reference for the performance of a good policy. We also compare DMDP-H, WD-AE and LD-AE. We expect that the latter two would perform well here; after all, the representation that they reconstruct is already quite compact. We



(a) WM-AE Baseline (b) LD-AE Baseline (c) DMDP-H Baseline (d) This paper

Figure 6: Abstract MDP for four approaches in CartPole. Nodes are PCA projections of abstract states, edges are predicted $\bar{T}_\phi$, colors are predicted values.

Average episode length ↑	Standard	Only 100 trajectories
GC-Model-free	**197.85** ± 2.16	23.84 ± 0.88
WM-AE	150.61 ± 30.48	114.47 ± 17.32
LD-AE	157.10 ± 11.14	154.73 ± 50.49
DMDP-H ($J = 0$)	39.32 ± 9.02	72.81 ± 20.16
Ours, $J = 1$,	174.64 ± 22.43	127.37 ± 44.02
Ours, $J = 3$	166.05 ± 24.73	148.30 ± 67.27
Ours, $J = 5$	186.31 ± 12.28	**171.53** ± 34.18

Table 3: CartPole results. Comparing average episode length over 100 episodes, reporting mean and standard deviation over 5 random seeds. The left column has standard settings, in the right column only 100 trajectories are encountered, and planning models are trained for only 100 epochs.

additionally evaluate performance when the amount of data is limited to only 100 trajectories (and we limit the number of training epochs for all planning approaches to 100 epochs). We plot the found latent space for our approach and the baselines in Figure 6. The goal in this problem is to reach the all-zero reward vector, which we set as the goal state with reward 1, and all other states to reward 0. For our approach and both auto-encoder baselines, the latent space forms a bowl with the goal in its center. The DMDP-H again shows a shrunk latent space, and does not have this bowl structure. Results are shown in Table 3. Our approach performs best out of all planning approaches. When trained fully, the model-free approach performs better. However, when we limit the number of environmental interactions to 100 trajectories, we see that the planning approach still finds a reasonable policy, while the model-free approach fails completely. This indicates that our approach is more data efficient.

4.4 Generalizing over Goals and Objects

In many tasks we need to be able to generalize not only over goals, but also object instances. We evaluate if our abstract state space generalizes to unseen objects in a problem class. For this we construct an object manipulation task. On each episode, an image of a piece of clothing is sampled from a set of training images in Fashion MNIST [52], and a goal translation of the image is sampled from a set of train goals (translations with negative x-offset: $(-3, \cdot)$ up to and including $(-1, \cdot)$). Thus, the underlying state space is a 7×7 grid. The translated image is provided to the agent as a goal state. The agent receives a reward of +1 if she moves the clothing to the correct translation. See Figure 8. At test time, we evaluate performance on test goals (translations with positive x-offset: $(1, \cdot)$ up to and including $(3, \cdot)$, seen before as states for training images but never as goals) and test images. The latent spaces for each of the four representation learning approaches are shown in Figure 7. For DMDP-H, the latent space collapses to all but a few points. For WD-AE and LD-AE, the latent space does not exhibit clear structure. For our approach, there is a clear square grid structure present in the latent space. However, the underlying translations for the images do not neatly align across images. Clustering such states together is interesting future work.

Results are shown in Table 4. The goal-conditioned model-free baseline has an easy time finding a good policy for the training setting. It also generalizes well to unseen images. However, it has trouble generalizing to new goal locations for both train and test images. Our planning approach, on the other hand, loses some performance on the training setting, but easily generalizes to both test images and test goals. Neither WM-AE nor LD-AE find good policies in this problem. They have a difficult time learning plannable representations because their focus is on reconstructing individual images.

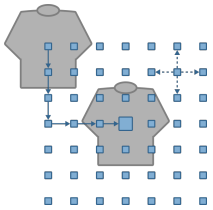
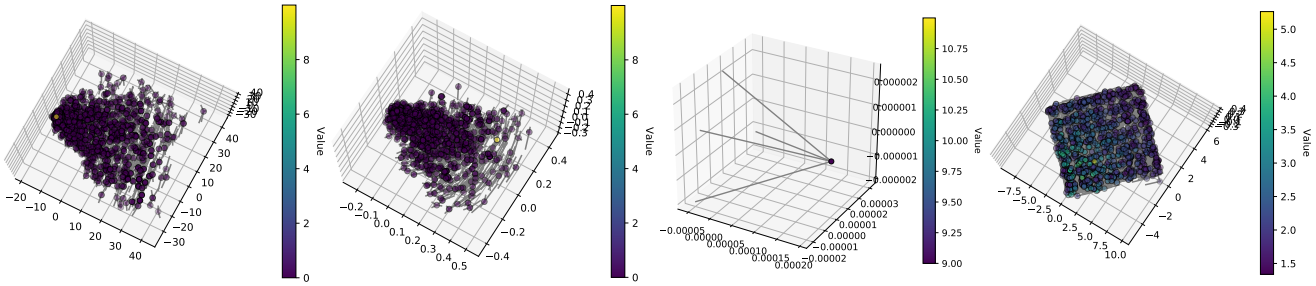


Figure 8: Transitions in the image manipulation task.

5 RELATED WORK

This paper proposes a method for learning action equivariant mappings of MDPs, and using these mappings for constructing plannable abstract MDPs. We learn action equivariant maps by minimizing MDP homomorphism metrics [46]. As a result, when the loss reaches zero the learned mapping is an MDP homomorphism [43]. MDP homomorphism metrics are a generalization of bisimulation metrics [10, 35]. Other works [6, 20, 51] consider equivariance to symmetry group actions in learning. Here, we use a more general



(a) WM-AE Baseline (b) LD-AE Baseline (c) DMDP-H Baseline (d) This paper

Figure 7: Abstract MDP for four approaches in planning in Fashion MNIST. Nodes are PCA projections of abstract states, edges are predicted $\bar{T}_\phi$, colors are predicted values.

Avg. ep. length ↓ Dataset	Train		Test	
Goal Set	Train	Test	Train	Test
GC-Model-free	**4.82** ± 0.33	9.67 ± 5.01	**4.75** ± 0.12	8.17 ± 2.67
WM-AE	59.95 ± 4.06	63.27 ± 3.36	64.27 ± 5.33	63.41 ± 2.04
LD-AE	56.39 ± 7.07	49.35 ± 4.05	51.45 ± 6.79	51.70 ± 3.97
DMDP-H ($J = 0$)	62.86 ± 3.87	66.68 ± 4.40	65.93 ± 4.98	64.86 ± 1.57
Ours, $J = 1$,	5.07 ± 0.87	**5.27** ± 0.56	5.69 ± 0.93	5.63 ± 0.96
Ours, $J = 3$	5.60 ± 0.97	5.46 ± 0.97	6.44 ± 1.12	5.42 ± 0.89
Ours, $J = 5$	5.36 ± 0.71	5.67 ± 1.20	6.36 ± 1.21	**5.34** ± 0.93

Table 4: Comparing average episode length of 100 episodes for planning in Fashion MNIST. Reporting mean and standard deviation over 5 random seeds.

version of equivariance under MDP actions for learning representations of MDPs. We learn representations of MDPs by 1) predicting the next latent state, 2) predicting the reward and 3) using negative sampling to prevent state collapse. Much recent work has considered self-supervised representation learning for MDPs. Certain works focus on predicting the next state using a contrastive loss [2, 30, 40], disregarding the reward function. However, certain states may be erroneously grouped together without a reward function to distinguish them. Gelada et al. [13] include both rewards and transitions to propose an objective based on stochastic bisimulation metrics [10, 15, 35]. However, at training time they focus on deterministically predicting the next latent state. Their proposed objective does not account for the possibility of latent space collapse, and for complex tasks they require a pixel reconstruction term. This phenomenon is also observed by François-Lavet et al. [11], who prevent it with two entropy maximization losses.

Many approaches to representation learning in MDPs depend (partially) on learning to reconstruct the input state [3, 7, 16, 17, 21, 25, 33, 47–50, 54]. A disadvantage of reconstruction losses is training a decoder, which is time consuming and usually not required for decision making tasks. Additionally, such losses emphasize visually salient features over features relevant to decision making.

Other approaches that side-step the pixel reconstruction loss include predicting which action caused the transition between two states [1], predicting the number of time steps between two states [4] or predicting objects in an MDP state using supervised learning [53]. Jonschkowski and Brock [24] identify a set of priors about the world and uses them to formulate self-supervised objectives. In Ghosh et al. [14], the similarity between two states is the difference in goal-conditioned policies needed to reach them from another state. Schrittwieser et al. [44] learn representations for tree-based search that must predict among others a policy and value function, and are thus not policy-independent. Earlier work on decoupling representation learning and planning exists [7, 49, 53]. However, these works use objectives that include a pixel reconstruction term [7, 49] or require labeling of objects in states for use in supervised learning [53].

Other work on planning algorithms in deep learning either assumes knowledge of the state graph [26, 34, 38, 45], builds a graph out of observed transitions [31] or structures the neural network architecture as a planner [9, 11, 39], which limits the search depth.

6 CONCLUSION

This paper proposes the use of 'equivariance under actions' for learning representations in deterministic MDPs. Action equivariance is enforced by the use of MDP homomorphism metrics in defining a loss function. We also propose a method of constructing plannable abstract MDPs from continuous latent spaces. We prove that for deterministic MDPs, when our objective function is zero and our method for constructing abstract MDP is used, the map we learn is an MDP homomorphism. Additionally, we show empirically that our approach is data-efficient and fast to train, and generalizes well to new goal states and instances with the same environmental dynamics. Potential future work includes an extension to stochastic MDPs and clustering states on the basis of MDP metrics. Using a clustering approach as part of model training, we can learn the prototypical states rather than sampling them. This comes at the cost of having to backpropagate through a discretization step, which in early experiments (using Gumbel-Softmax [23]) led to instability.

7 ACKNOWLEDGMENTS

We thank Patrick Forré, Jorn Peters and Michael Herman for helpful comments. T.K. acknowledges funding by SAP SE. F.A.O. received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 758824 —INFLUENCE).

REFERENCES

[1] Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. 2016. Learning to Poke by Poking: Experiential Learning of Intuitive Physics. In *Advances in Neural Information Processing Systems*.

[2] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Cote, and R Devon Hjelm. 2019. Unsupervised State Representation Learning in Atari. In *Advances in Neural Information Processing Systems*.

[3] Masataro Asai. 2019. Unsupervised Grounding of Plannable First-Order Logic Representation from Images. In *International Conference on Automated Planning and Scheduling*.

[4] Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando de Freitas. 2018. Playing Hard Exploration Games by Watching YouTube. In *Advances in Neural Information Processing Systems*.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv:1606.01540

[6] Taco S. Cohen and Max Welling. 2016. Group Equivariant Convolutional Networks. In *International Conference on Machine Learning*.

[7] Dane Corneil, Wulfram Gerstner, and Johanni Brea. 2018. Efficient Model-Based Deep Reinforcement Learning with Variational State Tabulation. In *International Conference on Machine Learning*.

[8] Thomas Dean and Robert Givan. 1997. Model Minimization in Markov Decision Processes. In *AAAI Conference on Artifical Intelligence/IAAI Conference on Innovative Applications of Artificial Intelligence*.

[9] Gregory Farquhar, Tim Rocktäschel, Maximilian Igl, and SA Whiteson. 2018. TreeQN and ATreeC: Differentiable Tree-Structured Models for Deep Reinforcement Learning. In *International Conference on Learning Representations*.

[10] Norm Ferns, Prakash Panangaden, and Doina Precup. 2004. Metrics for Finite Markov Decision Processes. In *Conference on Uncertainty in Artificial Intelligence*.

[11] Vincent François-Lavet, Yoshua Bengio, Doina Precup, and Joelle Pineau. 2019. Combined Reinforcement Learning via Abstract Representations. In *AAAI Conference on Artificial Intelligence*.

[12] Victor Garcia and Joan Bruna. 2018. Few-shot Learning with Graph Neural Networks. In *International Conference on Learning Representations*.

[13] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. 2019. DeepMDP: Learning Continuous Latent Space Models for Representation Learning. In *International Conference on Machine Learning*.

[14] Dibya Ghosh, Abhishek Gupta, and Sergey Levine. 2019. Learning Actionable Representations with Goal Conditioned Policies. In *International Conference on Learning Representations*.

[15] Robert Givan, Thomas Dean, and Matthew Greig. 2003. Equivalence Notions and Model Minimization in Markov Decision Processes. In *Artificial Intelligence*, Vol. 147. Elsevier.

[16] David Ha and Jürgen Schmidhuber. 2018. World models. arXiv:1803.10122

[17] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. 2019. Learning Latent Dynamics for Planning from Pixels. In *International Conference on Machine Learning*.

[18] Juris Hartmanis and R. E. Stearns. 1966. *Algebraic Structure Theory Of Sequential Machines*. Prentice-Hall, Inc.

[19] Peter Henderson, Riashat Islam, Joelle Pineau, David Meger, Doina Precup, and Philip Bachman. 2018. Deep Reinforcement Learning that Matters. In *AAAI Conference on Artificial Intelligence*.

[20] Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. 2018. Towards a Definition of Disentangled Representations. arXiv:1812.02230

[21] Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. 2018. Deep Variational Reinforcement Learning for POMDPs. In *International Conference on Machine Learning*.

[22] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. 2017. Reinforcement Learning with Unsupervised Auxiliary Tasks. In *International Conference on Learning Representations*.

[23] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*.

[24] Rico Jonschkowski and Oliver Brock. 2015. Learning State Representations with Robotic Priors. In *Autonomous Robots*, Vol. 39. Springer.

[25] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. 2020. Model-Based Reinforcement Learning for Atari. In *International Conference on Learning Representations*.

[26] Peter Karkus, David Hsu, and Wee Sun Lee. 2017. QMDP-net: Deep Learning for Planning under Partial Observability. In *Advances in Neural Information Processing Systems*.

[27] Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

[28] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*.

[29] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. 2018. Neural Relational Inference for Interacting Systems. In *International Conference on Machine Learning*.

[30] Thomas Kipf, Elise van der Pol, and Max Welling. 2020. Contrastive Learning of Structured World Models. In *International Conference on Learning Representations*.

[31] Martin Klissarov and Doina Precup. 2018. Diffusion-Based Approximate Value Functions. In *ICML ECA Workshop*.

[32] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. 2018. Model-Ensemble Trust-Region Policy Optimization. In *International Conference on Learning Representations*.

[33] Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart Russell, and Pieter Abbeel. 2018. Learning Plannable Representations with Causal InfoGAN. In *Advances in Neural Information Processing Systems*.

[34] Lisa Lee, Emilio Parisotto, Devendra Singh Chaplot, Eric P. Xing, and Ruslan Salakhutdinov. 2018. Gated Path Planning Networks. In *International Conference on Machine Learning*.

[35] Lihong Li, Thomas J. Walsh, and Michael L. Littman. 2006. Towards a Unified Theory of State Abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics*.

[36] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *International Conference on Machine Learning*.

[37] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level Control through Deep Reinforcement Learning. In *Nature*, Vol. 518.

[38] Sufeng Niu, Siheng Chen, Colin Targonski, Melissa Smith, Jelena Kova evi, and Hanyu Guo. 2018. Generalized Value Iteration Networks: Life Beyond Lattices. In *AAAI Conference on Artificial Intelligence*.

[39] Junhyuk Oh, Satinder Singh, and Honglak Lee. 2017. Value Prediction Network. In *Advances in Neural Information Processing Systems*.

[40] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. arXiv:1807.03748

[41] Martin L Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

[42] Balaraman Ravindran and Andrew G. Barto. 2001. *Symmetries and Model Minimization in Markov Decision Processes*. Technical Report. University of Massachusetts.

[43] Balaraman Ravindran and Andrew G. Barto. 2004. Approximate Homomorphisms: A Framework for Non-Exact Minimization in Markov Decision Processes. In *International Conference on Knowledge Based Computer Systems*.

[44] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. 2019. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. arXiv:911.08265

[45] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. 2016. Value Iteration Networks. In *Advances in Neural Information Processing Systems*.

[46] Jonathan Taylor, Doina Precup, and Prakash Panagaden. 2008. Bounding Performance Loss in Approximate MDP Homomorphisms. In *Advances in Neural Information Processing Systems*.

[47] Valentin Thomas, Jules Pondard, Emmanuel Bengio, Marc Sarfati, Philippe Beaudoin, Marie-Jean Meurs, Joelle Pineau, Doina Precup, and Yoshua Bengio. 2017. Independently Controllable Factors. arXiv:1708.01289

[48] Angelina Wang, Thanard Kurutach, Kara Liu, Pieter Abbeel, and Aviv Tamar. 2019. Learning Robotic Manipulation through Visual Planning and Acting. In *Robotics: Science and Systems*.

[49] Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. 2015. Embed to Control: a Locally Linear Latent Dynamics Model for Control from Raw Images. In *Advances in Neural Information Processing Systems*.

[50] Nicholas Watters, Loic Matthey, Matko Bosnjak, Christopher P Burgess, and Alexander Lerchner. 2019. COBRA: Data-Efficient Model-Based RL through Unsupervised Object Discovery and Curiosity-Driven Exploration. arXiv:1905.09275

[51] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. 2017. Harmonic Networks: Deep Translation and Rotation Equivariance. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[52] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747

[53] Amy Zhang, Adam Lerer, Sainbayar Sukhbaatar, Rob Fergus, and Arthur Szlam. 2018. Composable Planning with Attributes. In *International Conference on Machine Learning*.

[54] Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine. 2019. SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning. In *International Conference on Machine Learning*.