Learning Interaction-Based Representations for Reinforcement Learning Using Recursive State Similarity Metrics

Kevin Black
kevin3.black@gmail.com
UT Austin

Spring 2022

Abstract

Representation learning is a general technique for extracting features from an observation that are relevant to a particular task. In the context of reinforcement learning (RL), we investigate the problem of controllability-based representation learning, meaning we aim to extract only the elements of the observation that affect the agent's control over its environment. We aim to do so using environment interaction information alone, without an extrinsic signal such as pixel reconstruction, reward, or expert behavior. To this end, we introduce recursive state similarity metrics, a generalization of bisimulation metrics that can be used to capture complex information about environment dynamics. We present an algorithm that uses recursive state similarity metrics to learn controllability-based representations from offline interaction data. We test our algorithm in two 2D navigation environments and demonstrate the ability of the learned representations to accelerate RL on a goal-reaching task.

Contents

1 Introduction

Deep reinforcement learning (RL) has enjoyed enormous recent success in a wide variety of tasks. However, learning from high-dimensional images is an especially challenging problem, particularly in the real world, where observations are noisy and unstructured with many irrelevant details. While RL algorithms can still be successful on visual tasks, they often suffer from poor data efficiency, requiring millions of online environment interactions before performing well.

Prior methods often use representation learning for visual tasks, where each image is encoded as a low-dimensional vector. Intuitively, a good representation should encode only aspects of the observation that are "relevant" to the agent while ignoring any "irrelevant" distractors. The inclusion of irrelevant distractors diminishes the capacity of the representation to encode the information that does matter, which can slow down or prevent learning on downstream tasks. Traditional representation learning techniques, such as autoencoders, typically rely on pixel reconstruction as the learning signal. These techniques are fundamentally susceptible to irrelevant distractors since they encourage the representation to capture every last detail of the observation.

In RL, some prior methods have used alternative learning signals such as reward [20] or expert behavior [1] to encourage representations to capture information that is relevant to a particular task. In this work, we argue that there is a notion of relevancy based on *controllability* that generalizes across tasks. For example, a good representation in an autonomous driving environment should capture information about the road in front of the car and any immediate obstacles, since they have the potential to affect the agent's control. However, the mountains in the distance should be ignored, since they will never affect the agent's control.

One way to capture information about controllability is to learn an inverse dynamics model for the environment. A single-step inverse dynamics model captures *single-step controllability*, or the information from the observation relevant to the agent's control in the current timestep. In order to learn *multi-step controllability*, we extend ideas from bisimulation metrics [8]. Bisimulation metrics measure the similarity between states based on reward; two states are bisimilar if, for the same sequence of actions, they obtain similar rewards. They also admit a useful recursive definition: two states are bisimilar if, for the same action, they obtain similar single-step rewards and transition to bisimilar states. We use a bisimulation-like formulation based on single-step controllability rather than reward to learn a representation that captures information about multi-step controllability: i.e. the elements of the observation that affect that agent's control both now and in the future.

In this work, we first describe our extension of bisimulation, which we refer to as recursive state similarity metrics. We present an algorithm for representation learning based on these metrics and apply this algorithm to learn multi-step controllability-based representations from offline data. We test the ability of the representations to accelerate RL in two 2D navigation environments, as well as investigate the internal structure of the representations.

2 Related Work

Bisimulation This work builds heavily upon bisimulation metrics [8], which measure the similarity between states in a Markov decision process (MDP) based on their current and future reward. Zhang et al. [20] use bisimulation metrics to perform task-specific representation learning by matching ℓ_1 distance in the latent space with bisimulation distance. We also utilize this method of ℓ_1 distance matching. Agarwal et al. [1] extend the idea of bisimulation metrics to use expert behavior as the learning signal instead of reward and additionally introduce a contrastive method for learning representations based on a similarity metric. Castro [4], Castro et al. [5] propose further techniques for effectively learning representations based on bisimulation metrics. One key difference between our method and prior work is that prior work has used an *on-policy* state similarity metric in practice, which is tied to a particular policy. Since we desire controllability information that is policy-independent, we use the original off-policy formulation.

Controllability A wide range of prior work [14, 2, 16, 6] has used an inverse-dynamics-based objective to learn a controllability-based representation. Rakelly et al. [15] provide a unifying framework in which such objectives are called "inverse information" objectives since they maximize the predictive power of the action distribution that could have generated an observed transition. Inverse dynamics models can also be used to ground a latent space forward dynamics model [14, 2], which can encourage a representation to include information about controllability in future timesteps. We provide an alternative approach using state similarity metrics that is more explicit in how it captures future information as well as incorporates an interpretable discount factor.

3 Preliminaries

3.1 Markov Decision Processes

We assume that an environment has the underlying structure of a Markov Decision Process (MDP). Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma, s_0)$ define an MDP, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the transition dynamics, R is the reward function, γ is the discount factor, and s_0 is the initial state distribution. The goal of RL is to find a policy π that maximizes the cumulative expected return $\mathbb{E}_{\pi,P}\left[\sum_t \gamma^t R(s_t, a_t)\right]$ where $a_t \sim \pi(\cdot|s_t)$ and $s_{t+1} \sim P(\cdot|s_t, a_t)$. Though we deal with image observations, we do not explicitly consider the problem of partial observability.

3.2 Bisimulation

A bisimulation [11] is an equivalence relation that groups states together using reward. It has a convenient recursive formulation: two states are equivalent under a bisimulation relation if and only if, for every action, they produce the same reward and have next-state distributions that are also equivalent under the bisimulation relation. However, exact partitioning using bisimulation is impractical in continuous state spaces. Bisimulation metrics [8] relax the idea of bisimulation by defining a pseudometric space $(\mathcal{S}, d)$ where the distance function $d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ measures the "soft" bisimilarity between states. d is defined recursively as follows:

$$d(x, y) = \max_{a \in \mathcal{A}} \left[(1 - \gamma)|R(x, a) - R(y, a)| + \gamma \mathcal{W}_1^d(P(\cdot|x, a), P(\cdot|y, a) \right] \qquad x, y \in \mathcal{S} \tag{1}$$

The first term captures the single-step difference in reward. The second term captures future differences in reward using $\mathcal{W}_d^1$, the Wasserstein (or earth mover's) distance [19] between next-state distributions under the distance function d. Intuitively, two states are bisimilar if, for all actions, they produce similar reward and transition to states that are also bisimilar. The discount factor $\gamma \in (0, 1)$ ensures that the further in the future a transition occurs, the less it will affect the bisimilarity between states in the present.

4 Methods

4.1 Recursive State Similarity Metrics

We extend the idea of bisimulation metrics by defining the generalized recursive state similarity metric as follows:

$$d(x, y) = \underset{a \in \mathcal{A}}{\Phi} \left[(1 - \gamma)C(x, y, a) + \gamma \mathcal{W}_1^d(P(\cdot|x, a), P(\cdot|y, a)) \right] \qquad x, y \in \mathcal{S} \tag{2}$$

Instead of reward, we allow for any existing distance function between states $C(x, y, a)$ that may or may not depend on the action. Instead of max, we allow for any arbitrary reduction over the actions Φ. The bisimulation metric uses max to account for the *worst-case* sequence of actions that maximize the difference between the two states; therefore, two states will be bisimilar if and only if they produce similar rewards for *all* action sequences. In the case of bisimulation, this allows for some appealing properties such as connections to the optimal value function [7]. However, in general, we may not care about *all* action sequences. For example, it may be desirable for Φ to be the expected value over some policy. In this work, we use Φ = mean, i.e. the expected value over the uniform policy. This reflects the fact that we care about all action sequences equally and, given these action sequences, we are interested in the "average case" of their contribution to the distance between two states.

4.2 Learning The Multi-Step Representation

Algorithm 1 Learning The Multi-Step Representation

```
1: for Time $t = 0$ to $\infty$ do
2:     Sample batch $B$ from dataset
3:     Train forward model $\hat{P}$ in the latent space of $\phi_{\bar{\theta}}$ using MSE loss
4:     Train representation: $\mathbb{E}_{(x,y)\sim B\times B}[L_\theta(x,y)]$        ▷ Eq. (3)
5:     Update target parameters: $\bar{\theta} \leftarrow \tau\theta + (1-\tau)\bar{\theta}$
6: end for
```

Let us first assume oracle access to a "base case" distance function $C(x, y, a)$. Following Zhang et al. [20], we learn a representation by encouraging ℓ_1 distances in the representation space to directly match the similarity metric from equation (2). Let $\phi_\theta : \mathcal{S} \to \mathcal{Z}$ be an encoder with parameters θ mapping observed states to the latent representation space $\mathcal{Z}$. We draw batches of state pairs (x, y) and train the encoder with the following mean-squared error loss:

$$L_\theta(x,y) = \Big(||\phi_\theta(x) - \phi_\theta(y)||_1 - \hat{d}(x,y)\Big)^2 \tag{3}$$

$\hat{d}$ is an approximation of equation (1), defined by:

$$\hat{d}(x,y) = \underset{a\in\mathcal{A}}{\Phi}\Big[(1-\gamma)C(x,y,a) + \gamma\mathcal{W}_1^{||\cdot||_1}(\hat{P}(\cdot|\phi_{\bar{\theta}}(x),a), \hat{P}(\cdot|\phi_{\bar{\theta}}(y),a))\Big] \tag{4}$$

$\hat{P}$ is a latent space forward dynamics model trained separately using mean-squared error (MSE) loss. For learning stability, it is trained using representations from a target encoder $\phi_{\bar{\theta}}$, where $\bar{\theta}$ is an exponential moving average of the online encoder parameters θ as introduced in Mnih et al. [13]. The forward dynamics model does not pass gradients back into the target encoder.

We make several simplifying assumptions. In this work, we use Φ = mean. We assume deterministic transition dynamics, meaning the forward dynamics are modeled as a deterministic function $\hat{P} : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$. We then have $\mathcal{W}_1^{||\cdot||_1}(\hat{P}(\cdot|\phi_{\bar{\theta}}(x),a), \hat{P}(\cdot|\phi_{\bar{\theta}}(y),a)) = ||\hat{P}(\phi_{\bar{\theta}}(x),a) - \hat{P}(\phi_{\bar{\theta}}(y),a)||_1$ so no Wasserstein distance needs to be computed. However, stochastic transition dynamics can be handled as in Zhang et al. [20] by modeling the latent forward dynamics as a Gaussian distribution, for which there exists a closed-form formula for the Wasserstein distance. Finally, we assume a discrete action space, so that the inner terms of equation (4) can be easily evaluated for every action. An extension to continuous action spaces is left for future work.

The entire learning procedure is summarized in Algorithm 1.

4.3 Learning The Single-Step Representation

To instantiate the "base case" distance function $C(x, y, a)$, we learn a separate single-step representation based on an inverse dynamics objective.

Let ψ_μ be an encoder with parameters μ that maps states into latent representation space. We draw batches of (state, action, next state) samples (x, a, x'), and train with the following loss:

$$L_\mu(x, a, x') = L_{inv}[a, \text{ID}(\psi_\mu(x), \psi_\mu(x'))] \tag{5}$$

ID is an inverse dynamics model trained to predict the action that caused the transition between consecutive encoded states. In environments with discrete action spaces, L_{inv} is the standard cross-entropy loss.

Intuitively, the inverse dynamics objective encourages the representation to encode information that is predictive of the agent's actions. From an information-theoretic perspective, it maximizes the mutual information between pairs of consecutive encoded states $(\psi_\mu(x), \psi_\mu(x'))$ and the action distribution [15].

We extract a similarity metric from ψ_μ by defining $C(x, y, a) = ||\psi_\mu(x) - \psi_\mu(y)||_1$.

5 Experiments

5.1 Environments

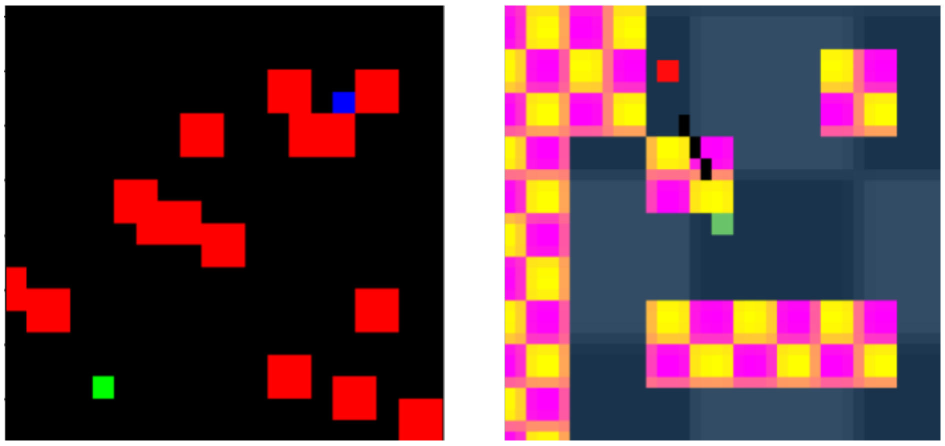

Figure 1: An image observation from Nav2D (left) and Egocentric Point Mass (right)

Experiments are carried out in two navigation environments (Figure 1). The first is a simple discrete 2D navigation environment (Nav2D). The agent, in green, can move up, down, left, or right. The goal is to reach the blue square, and the agent is unable to move through the red obstacles. The obstacles, goal location, and starting agent position are randomized every episode. A reward of 0 is provided for reaching the goal square, and a reward of -1 is provided for every timestep that the goal square is not reached.

The second environment is referred to as Egocentric Point Mass. The reward function, as well as the goal and obstacle mechanics, are the same as in Nav2D. However, the actions control the acceleration of the agent rather than the displacement, and the agent is simulated as a point mass using MuJoCo [17]. The observations are egocentric, meaning the camera moves such that the agent is always at the center, and the entire map is not in view. A black line points in the direction of the goal. The purpose of this environment is to require the agent to have an increased awareness of distant obstacles, particularly in the direction that it is moving, to avoid colliding with them and wasting time. The egocentric view is more realistic with respect to real-life robotic navigation tasks.

5.2 Data Collection

Offline datasets are collected using an ϵ-greedy algorithm: with probability ϵ, the agent takes a uniformly random action, and with probability $1 - \epsilon$, the agent takes a step along an optimal path toward the goal. This ensures some amount of obstacle collisions to allow for controllability-based learning. Additionally, goal information is masked out in the offline datasets.

5.3 Quantitative Results

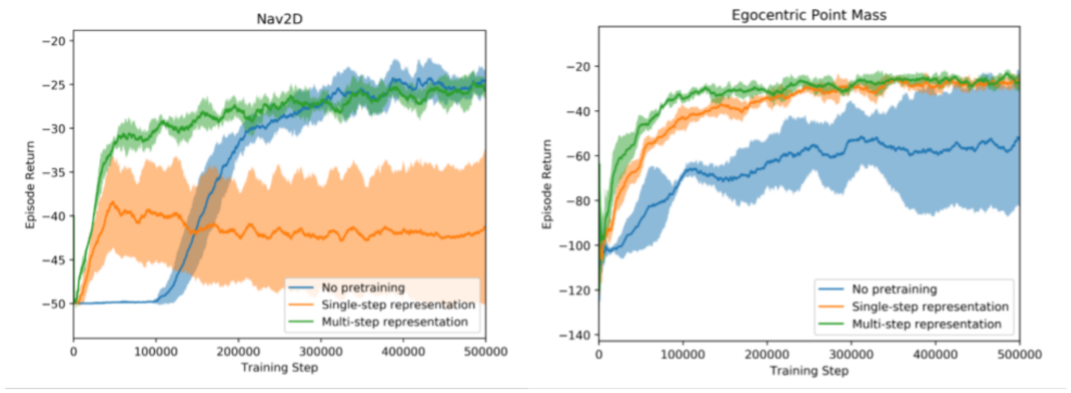

Figure 2: Results of running RL in both environments. All curves are averaged over 5 seeds, with one standard error shaded in. The Q-network is initialized with either the multi-step encoder weights, the single-step encoder weights, or random weights ("No pretraining").

Figure 2 shows training curves from running RL with weights initialized using the single- or multi-step encoder. Training is carried out using double deep Q-learning [9] with hindsight experience replay (HER) [3]. HER is necessary for solving both tasks due to the sparsity of the reward function.

The multi-step representation outperforms the baseline with no pretraining and the single-step representation in both environments. All initializations eventually solve the task, except for the single-step representation in the Nav2D environment; we hypothesize that the single-step initialization may cause Q-learning to get trapped in a local optimum in this environment. The primary advantage of the multi-step representation is improved sample efficiency, allowing the agent to effectively solve the task in less than 100,000 environment interactions. While the performance gap between the single-step representation and the multi-step representation is smaller in the Egocentric Point Mass environment, the multi-step representation achieves significantly better returns during the first 200,000 training steps, representing more efficient navigation to the goal.

5.4 Visualizing Representations

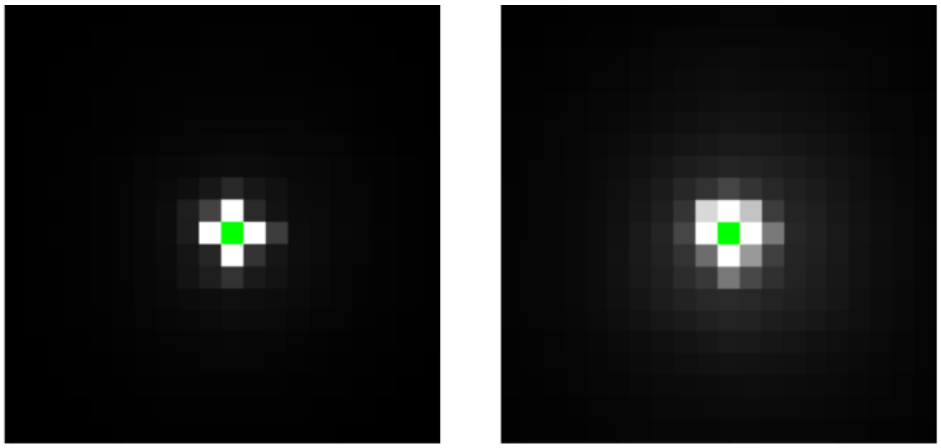

Figure 3: A visualization of obstacle attention for the single-step representation ψ_μ (left) and the multi-step representation ϕ_θ (right) with a discount factor of $\gamma = 0.75$. The agent is the green pixel, and the intensity of all other pixels is proportional to the amount that the representation changes (measured by ℓ_1 distance) when an obstacle is placed at that pixel.

We perform several visualizations to gain an intuitive understanding of the learned representations. Figure 3 compares obstacle attention between the single-step and multi-step representations. As expected, the single-step representation mostly attends to information relevant in the *current* timestep — in this case, whether or not there is an obstacle right next to the agent — since this is the only information required for inverse dynamics. The multi-step representation extends this information through time, accounting for obstacles that may be encountered in future timesteps. However, the attention fades as the distance from the agent increases, due to the discount factor $\gamma = 0.75$.

Figure 4 visualizes the multi-step representation in a version of the 2D navigation environment where the agent is stuck in a fixed corridor. The representation space is structured intuitively, producing different "branches" depending on whether the corridor is partially

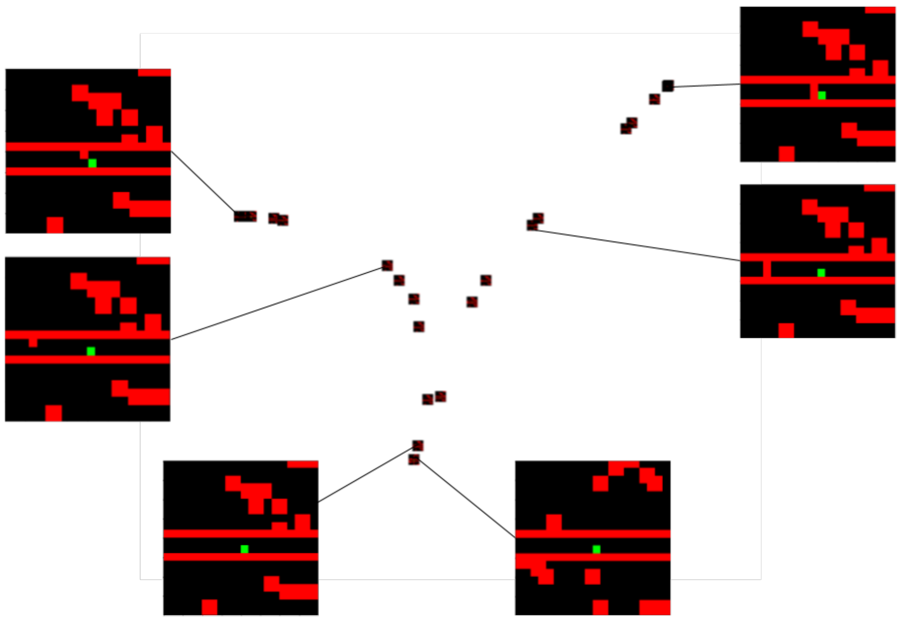

Figure 4: A t-SNE [18] projection of the state similarity representation in the Nav2D corridor environment. States are close together (bottom) when only the unreachable obstacles outside of the corridor change. States move upward along the left "branch" when the corridor is partially blocked, with the states being further away the closer the obstacle is to the agent. States move upward along the right "branch" when the corridor is fully blocked.

or fully blocked, as this represents fundamentally different future dynamics. Unreachable obstacles outside the corridor do not affect the representation at all.

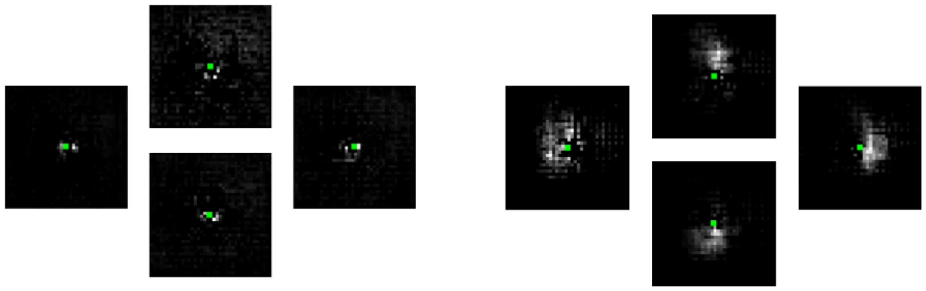

Figure 5: A visualization of obstacle attention in the Egocentric Point Mass environment, separated by agent velocity, for the single-step (left) and multi-step (right) representations. The four images on each side show average obstacle attention for states where the agent is moving up, down, left, and right. For the multi-step representation, there is increased attention in the direction the agent is moving, whereas with the single-step representation there is no such effect.

Figure 5 visualizes obstacle attention in the Egocentric Point Mass domain separated by agent velocity. We see again that the multi-step representation generally attends to obstacles that are further away. However, there is additional structure to the representation in that it mostly attends to obstacles in the direction that the agent is moving. This is because obstacles in the direction that the agent is moving are more likely to affect the agent's control in the near future.

6 Conclusion

We extend prior work in bisimulation metrics to develop a generalized recursive state similarity metric along with an algorithm leveraging it to learn state representations. We apply this algorithm to an inverse-dynamics-based objective to learn a multi-step controllability-based representation from environment interaction alone without any extrinsic signal such as reward or expert behavior. We demonstrate the ability of this representation to accelerate reinforcement learning in two 2D navigation domains. Additionally, we qualitatively demonstrate the ability of the state similarity metric to produce representations that encode meaningful, long-horizon information about objects in the environment.

There are many directions for future work. One is to apply these methods to more complex, real-world tasks. An advantage of controllability-based representation learning is its ability to learn from data without reward labels. This is particularly attractive for the real world, where collecting labeled data is particularly expensive. Robotic manipulation and autonomous driving may be good candidates. Another direction is to apply the controllability-based representation to exploration or skill learning problems, where it may be more useful than with plain reinforcement learning.

7 Acknowledgements

I would like to thank my research advisor, Scott Niekum, for his mentorship. I would also like to thank Caleb Chuck for his mentorship and assistance with this project. Finally, I would like to thank my entire thesis committee: Scott Niekum, Yuke Zhu, and Amy Pavel for their time and effort.

References

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

[2] Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 5092–5100, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

[3] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf`.

[4] Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 10069–10076. AAAI Press, 2020. URL `https://aaai.org/ojs/index.php/AAAI/article/view/6564`.

[5] Pablo Samuel Castro, Tyler Kastner, P. Panangaden, and Mark Rowland. Mico: Learn-

ing improved representations via sampling-based state similarity for markov decision processes. *ArXiv*, abs/2106.08229, 2021.

[6] Jongwook Choi, Yijie Guo, Marcin Moczulski, Junhyuk Oh, Neal Wu, Mohammad Norouzi, and Honglak Lee. Contingency-aware exploration in reinforcement learning. *arXiv preprint arXiv:1811.01483*, 2018.

[7] Norm Ferns and Doina Precup. Bisimulation metrics are optimal value functions. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, page 210–219, Arlington, Virginia, USA, 2014. AUAI Press. ISBN 9780974903910.

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

[9] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2094–2100. AAAI Press, 2016.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

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

[12] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9628–9639, Red Hook, NY, USA, 2018. Curran Associates Inc.

[13] 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. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb 2015. ISSN 1476-4687. doi: 10.1038/nature14236. URL `https://doi.org/10.1038/nature14236`.

[14] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2778–2787. JMLR.org, 2017.

[15] Kate Rakelly, Abhishek Gupta, Carlos Florensa, and Sergey Levine. Which mutual-information representation learning objectives are sufficient for control? *Advances in Neural Information Processing Systems*, 34, 2021.

[16] Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307*, 2016.

[17] 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, 2012. doi: 10.1109/IROS.2012.6386109.

[18] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[19] Cédric Villani. *Optimal transport: old and new.* Springer, 2008.

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

A Training Details

A.1 Data Collection

In both environments, a dataset of 1,000,000 interactions is used for training the representations. In Nav2D, the data is collected using $\epsilon = 0.5$. In Egocentric Point Mass, the data is collected using $\epsilon = 0.2$.

A.2 Representation Learning

In the Nav2D domain, both the single-step and multi-step encoders are 6-layer convolutional networks with ReLU activations that induce a latent dimension of 128. In the Egocentric Point Mass domain, both encoders are 7-layer networks that induce a latent dimension of 196. As in CoordConv [12], two pixel coordinate channels are concatenated to the image observations before passing them through the network. The inverse dynamics model ID, as well as the forward dynamics model $\hat{P}$, are both fully connected networks with one hidden layer of 256 neurons and ReLU activations. The multi-step components $(\phi_\theta, \hat{P})$ are trained with a learning rate of 0.0001, $\tau = 0.005$, $\gamma = 0.75$. The interaction-based components (ψ_μ, ID) are trained with a learning rate of 0.0002. Both components are optimized using Adam [10]. The single-step component is first trained for 500,000 steps, and then the multi-step representation is trained for 500,000 steps using the frozen single-step encoder.

A.3 Reinforcement Learning

The representations are evaluated using double deep Q-learning [9] with hindsight experience replay (HER) [3], where the convolution portion of the Q-network is initialized with the parameters from the encoder ϕ_θ or ψ_μ. The hyperparameters used are a batch size of 32, $\gamma = 0.99$, $\tau = 0.005$, a learning rate of 0.0001, and ϵ-greedy exploration that starts with $\epsilon = 0.9$ and decays linearly to $\epsilon = 0.2$ over the first 10,000 steps.

B Code

The code for this project can be found at `https://github.com/kvablack/nav2d-representation`.