Learning Invariant Representations for Reinforcement Learning without Reconstruction

Amy Zhang [*1 2 3] **Rowan McAllister** [*4] **Roberto Calandra** [3] **Yarin Gal** [5] **Sergey Levine** [4]

1. Introduction

Learning control from images is important for many real world applications. While deep reinforcement learning (RL) has enjoyed many successes in simulated tasks, learning control from real vision is more complex, especially outdoors, where images reveal detailed scenes of a complex and unstructured world. Furthermore, while many RL algorithms can *eventually* learn control from real images given unlimited data, data-efficiency is often a necessity in real trials which are expensive and constrained to real-time. Prior methods for data-efficient learning of simulated visual tasks typically use representation learning to summarize images by encoding them into smaller vectored representations better suited for RL. For example, sequential autoencoders aim to learn *lossless* representations of streaming observations—sufficient to reconstruct current observations and predict future observations—from which various RL algorithms can be trained (Hafner et al., 2018; Lee et al., 2019; Yarats et al., 2019). However, such methods are *task-agnostic*: the models represent all dynamic elements they observe in the world, whether they are relevant to the task or not. We argue such representations can easily "distract" RL algorithms with irrelevant information in the case of real images. The issues of distraction is less evident in popular simulation MuJoCo and Atari tasks, since any change in observation space is likely task-relevant. By contrast, visual images that autonomous cars observe contain predominately task-irrelevant information, like cloud shapes and architectural details, illustrated in Figure 5.

Rather than learning control-agnostic representations that focus on accurate reconstruction of clouds and buildings, we would rather achieve a more compressed representation from a *lossy* encoder, which only retains state information relevant to our task. If we would like to learn representations that capture only task-relevant elements of the state and are *invariant* to task-irrelevant information, intuitively we can utilize the reward signal to determine task-relevance. As *cumulative* rewards are our objective, state elements are relevant not only if they influence the current reward, but also if they influence state elements in the future that *in turn* influence future rewards. This recursive relationship can be distilled into a recursive notion of state abstraction: an ideal representation is one that is predictive of reward, and also predictive of itself in the future.

We propose learning such an invariant representation using the bisimulation metric, where the distance between two observation encodings correspond to how "behaviourally different" (Ferns and Precup, 2014) both observations are. Our main contribution is a practical representation learning method based on the bisimulation metric suitable for downstream control, which we call deep bisimulation for control (DBC). We additionally provide theoretical analysis that proves value bounds between the optimal value function of the true MDP and the optimal value function of the MDP constructed by the learned representation. Empirical evaluations demonstrate our non-reconstructive using bisimulation approach is substantially more robust to task-irrelevant distractors when compared to prior approaches that use reconstruction losses or contrastive losses. Our initial experiments insert natural videos into the background of MoJoCo control task as complex distraction. Our second setup is a high-fidelity highway driving task using CARLA (Dosovitskiy et al., 2017), showing that our representations can be trained effectively even on highly realistic images with many distractions, such as trees, clouds, and buildings.

2. Preliminaries

We start by introducing notation and outlining realistic assumptions about underlying structure in the environment. Then, we review state abstractions and metrics for state similarity. We assume the reader is familiar with MDPs, but background can be found in Appendix A.

Bisimulation is a form of state abstraction that groups states $\mathbf{s}_i$ and $\mathbf{s}_j$ that are "behaviorally equivalent" (Li et al., 2006). For any action sequence $\mathbf{a}_{0:\infty}$, the probabilistic sequence of rewards from $\mathbf{s}_i$ and $\mathbf{s}_j$ are identical. A more compact definition has a recursive form: two states are bisimilar if they share both the same immediate re-

[*]Equal contribution [1]McGill University [2]Mila [3]Facebook AI Research [4]UC Berkeley [5]University of Oxford. Correspondence to: Amy Zhang <amy.x.zhang@mail.mcgill.ca>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020.

ward and equivalent distributions over the next bisimilar states (Larsen and Skou, 1989; Givan et al., 2003).

Definition 1 (Bisimulation Relations (Givan et al., 2003)). *Given an MDP $\mathcal{M}$, an equivalence relation B between states is a bisimulation relation if, for all states $\mathbf{s}_i, \mathbf{s}_j \in \mathcal{S}$ that are equivalent under B (denoted $\mathbf{s}_i \equiv_B \mathbf{s}_j$) the following conditions hold: $\mathcal{R}(\mathbf{s}_i, \mathbf{a}) = \mathcal{R}(\mathbf{s}_j, \mathbf{a}), \mathcal{P}(G|\mathbf{s}_i, \mathbf{a}) = \mathcal{P}(G|\mathbf{s}_j, \mathbf{a}), \forall \mathbf{a} \in \mathcal{A}, \quad \forall G \in \mathcal{S}_B$, where $\mathcal{S}_B$ is the partition of $\mathcal{S}$ under the relation B (the set of all groups G of equivalent states), and $\mathcal{P}(G|\mathbf{s}, \mathbf{a}) = \sum_{\mathbf{s}' \in G} \mathcal{P}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$.*

Exact partitioning with bisimulation relations is generally impractical in continuous state spaces, as the relation is highly sensitive to infinitesimal changes in the reward function or dynamics. For this reason, **Bisimulation Metrics** (Ferns et al., 2011; Ferns and Precup, 2014; Castro, 2020) softens the concept of state partitions, and instead defines a pseudometric space $(\mathcal{S}, d)$, where a distance function $d : \mathcal{S} \times \mathcal{S} \mapsto \mathbb{R}_{\geq 0}$ measures the "behavioral similarity" between two states[1]. Defining a distance d between states requires defining both a distance between rewards, and distance between state distributions. Prior works use the Wasserstein metric for the latter, originally used in the context of bisimulation metrics by van Breugel and Worrell (2001). The p^{th} Wasserstein metric is defined between two probability distributions $\mathcal{P}_i$ and $\mathcal{P}_j$ as $W_p(\mathcal{P}_i, \mathcal{P}_j; d) = (\inf_{\gamma' \in \Gamma(\mathcal{P}_i, \mathcal{P}_j)} \int_{\mathcal{S}\times\mathcal{S}} d(\mathbf{s}_i, \mathbf{s}_j)^p \, \mathrm{d}\gamma'(\mathbf{s}_i, \mathbf{s}_j))^{1/p}$, where $\Gamma(\mathcal{P}_i, \mathcal{P}_j)$ is the set of all couplings of $\mathcal{P}_i$ and $\mathcal{P}_j$. This is known as the "earth mover" distance, denoting the cost of transporting mass from one distribution to another (Villani, 2003). Finally, the bisimulation metric is the reward difference added to the Wasserstein distance between transition distributions:

Definition 2 (Bisimulation Metric). *From Theorem 2.6 in Ferns et al. (2011) with $c \in [0, 1)$: $d(\mathbf{s}_i, \mathbf{s}_j) = \max_{\mathbf{a}\in\mathcal{A}} (1 - c) \cdot |\mathcal{R}^{\mathbf{a}}_{\mathbf{s}_i} - \mathcal{R}^{\mathbf{a}}_{\mathbf{s}_j}| + c \cdot W_1(\mathcal{P}^{\mathbf{a}}_{\mathbf{s}_i}, \mathcal{P}^{\mathbf{a}}_{\mathbf{s}_j}; d)$.*

3. Learning Representations for Control with Bisimulation Metrics

We propose Deep Bisimulation for Control (DBC), a data-efficient approach to learn control policies from unstructured, high-dimensional observations. In contrast to prior work on bisimulation, which typically aims to learn a distance function of the form $d : \mathcal{S}\times\mathcal{S} \mapsto \mathbb{R}_{\geq 0}$ between observations, our aim is instead to learn *representations* $\mathcal{Z}$ under which ℓ_1 distances correspond to bisimulation metrics, and then use these representations to improve reinforcement learning. Our goal is to learn encoders $\phi : \mathcal{S} \mapsto \mathcal{Z}$ that capture representations of states that are suitable to control, while discarding any information that is *irrelevant* for control. Any representation that relies on reconstruction of the observation cannot do this, as these irrelevant details are still important for reconstruction. We hypothesize that bisimulation metrics can acquire this type of representation, without any reconstruction.

To train our encoder ϕ towards our desired relation $d(\mathbf{s}_i, \mathbf{s}_j) := ||\phi(\mathbf{s}_i) - \phi(\mathbf{s}_j)||_1$, we draw batches of observations pairs, and minimise the mean square error between the on-policy bisimulation metric and Euclidean distance in the latent space:

$$J(\phi) = \Big(||\mathbf{z}_i - \mathbf{z}_j||_1 - |\hat{\mathcal{R}}(\bar{\mathbf{z}}_i) - \hat{\mathcal{R}}(\bar{\mathbf{z}}_j)| - \gamma \cdot W_2\big(\hat{\mathcal{P}}(\cdot|\bar{\mathbf{z}}_i, \bar{\pi}(\bar{\mathbf{z}}_i)), \hat{\mathcal{P}}(\cdot|\bar{\mathbf{z}}_j, \bar{\pi}(\bar{\mathbf{z}}_j))\big)\Big)^2, \quad (1)$$

where $\mathbf{z}_i = \phi(\mathbf{s}_i)$, $\mathbf{z}_j = \phi(\mathbf{s}_j)$, $\bar{\mathbf{z}}$ denotes $\phi(\mathbf{s})$ with stop gradients, and $\bar{\pi}$ is the mean policy output.

Algorithm 1 Deep Bisimulation for Control (DBC)

```
for Time t = 0 to ∞ do
    Encode observation z_t = φ(s_t)
    Execute action a_t ~ π(z_t)
    Record data: D ← D ∪ {s_t, a_t, s_{t+1}, r_{t+1}}
    Sample batch B_i ~ D
    Permute batch randomly: B_j = permute(B_i)
    Train policy: E_{B_i}[J(π)]                          ▷ Algorithm 2
    Train encoder: E_{B_i,B_j}[J(φ)]                     ▷ Equation (1)
    Train dynamics: J(P̂,φ) = (P̂(φ(s_t), a_t) − z̄_{t+1})^2
    Train reward: J(R̂,P̂,φ) = (R̂(P̂(φ(s_t), a_t)) − r_{t+1})^2
end for
```

Algorithm 2 Train Policy (changes to SAC in blue)

```
Get value: V = min_{i=1,2} Q̂_i(φ̂(s)) − α log π(a|s)
Train critics: J(Q_i, φ) = (Q_i(φ(s)) − r − γV)^2
Train actor: J(π) = α log p(a|s) − min_{i=1,2} Q_i(s)
Train alpha: J(α) = −α log p(a|s)
Update target critics: Q̂_i ← τ_Q Q_i + (1 − τ_Q)Q̂_i
Update target encoder: φ̂ ← τ_φ φ + (1 − τ_φ)φ̂
```

Equation (1) uses both a reward model $\hat{\mathcal{R}}$ and dynamics model $\hat{\mathcal{P}}$, which have their own training steps in Algorithm 1. Our reward model is a deterministic network, and our dynamics model $\hat{\mathcal{P}}$ outputs a Gaussian distribution. For this reason, we use the 2-Wasserstein metric W_2 in Equation (1), as opposed to the 1-Wasserstein in Definition 2, since the W_2 metric has a convenient closed form: $W_2(\mathcal{N}(\mu_i, \Sigma_i), \mathcal{N}(\mu_j, \Sigma_j))^2 = ||\mu_i - \mu_j||_2^2 + ||\Sigma_i^{1/2} - \Sigma_j^{1/2}||_{\mathcal{F}}^2$, where $||\cdot||_{\mathcal{F}}$ is the Frobenius norm. For all other distances we continue using the ℓ_1 norm.

Incorporating control. We combine our representation learning approach (Algorithm 1) with the soft actor-critic (SAC) algorithm (Haarnoja et al., 2018) to devise a practical reinforcement learning method. We modified SAC slightly in Algorithm 2 to allow the value function to backprop to our encoder, which can improve performance further (Yarats et al., 2019; Rakelly et al., 2019). Although, in principle, our method could be combined with any RL

[1]Note that d is a pseudometric, meaning the distance between two different states can be zero, corresponding to behavioral equivalence.

algorithm, including the model-free DQN (Mnih et al., 2015), or model-based PETS (Chua et al., 2018). Implementation details and hyperparameter values of DBC are summarized in the appendix, Table 1. We train DBC by iteratively updating four components in turn: a dynamics model $\hat{\mathcal{P}}$, reward model $\hat{\mathcal{R}}$, encoder ϕ with Equation (1), and policy π (in this case, with SAC). A single loss function would be less stable, and require balancing components. The inputs of each loss function $J(\cdot)$ in Algorithm 1 represents which components are updated. After each training step, the policy π is used to step in the environment, the data is collected in a replay buffer $\mathcal{D}$, and a batch is randomly selected to repeat training.

4. Generalization & Links to Causal Inference

While DBC enables representation learning without pixel reconstruction, it leaves open the question of how good the resulting representations really are. In this section, we present connections to causal inference that instruct on the generalization prropeties of the representation learned via DBC. Additional bounds on the value function are presented in the appendix.

MDP dynamics have a strong connection to causal inference and causal graphs, which are directed acyclic graphs (Jonsson and Barto, 2006; Schölkopf, 2019; Zhang et al., 2020). Specifically, the state and action at time t causally affect the next state at time $t+1$. In this work, we care about the components of the state space that causally affect current and future reward. Deep bisimulation for control representations connect to *causal feature sets*, or the minimal feature set needed to predict a target variable (Zhang et al., 2020). This connection tells us that these features are the minimal sufficient statistic of the current and future reward, and therefore consist of (and only consist of) the *causal ancestors* of the reward variable $\mathcal{R}$.

Definition 3 (Causal Ancestors)**.** *In a causal graph where nodes correspond to variables and directed edges between a parent node P and child node C are causal relationships, the causal ancestors $AN(C)$ of a node are all nodes in the path from C to a root node.*

If there are interventions on *distractor variables*, or variables that control the rendering function q and therefore the rendered observation but do not affect the reward, the causal feature set will be robust to these interventions, and correctly predict current and future reward in the linear function approximation setting (Zhang et al., 2020). As an example, in the context of autonomous driving, an intervention can be a change in weather, or a change from day to night which affects the observation space but not the dynamics or reward. Finally, we show that a representation based on the bisimulation metric generalizes to other reward functions with the same causal ancestors, with an

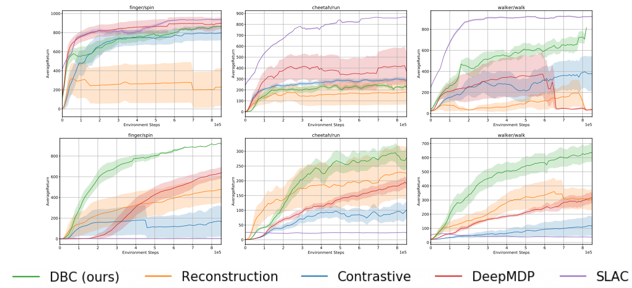

Figure 1. Results comparing out DBC method to baselines on 10 seeds with 1 standard error shaded in the default setting (top) and natural video setting (bottom).

example causal graph in Figure 6.

Theorem 1 (Task Generalization)**.** *Given an encoder $\phi : O \mapsto S$ that maps observations to a latent bisimulation metric representation where $||\phi(\mathbf{s}_i) - \phi(\mathbf{s}_j)||_2 := \tilde{d}(\mathbf{s}_i, \mathbf{s}_j)$, S encodes information about all the causal ancestors of the reward $AN(R)$.*

Proof in appendix. This result shows that the learned representation will generalize to unseen reward functions, as long as the new reward function has a subset of the same causal ancestors. As an example, a representation learned for a robot to walk will likely generalize to learning to run, because the reward function depends on forward velocity and all the factors that contribute to forward velocity. Theorem 1 shows that the learned representation will be robust to spurious correlations, or changes in factors that are not in $AN(R)$. This complements Theorem 5, that the representation is a minimal sufficient statistic of the optimal value function, improving generalization over non-minimal representations. We show empirical validation of these findings in Section 5.2.

5. Experiments

Our central hypothesis is that our non-reconstructive bisimulation based representation learning approach should be substantially more robust to task-irrelevant distractors. To that end, we evaluate our method in a clean setting without distractors, as well as a much more difficult setting with distractors. We compare against several baselines. The first is Stochastic Latent Actor-Critic (SLAC, Lee et al. (2019)), a state-of-the-art method for pixel observations on DeepMind Control that learns a dynamics model with a reconstruction loss. The second is DeepMDP (Gelada et al., 2019), a recent method that also learns a latent representation space using a latent dynamics model, reward model, and distributional Q learning, but for which they needed a reconstruction loss to scale up to Atari. Finally, we compare against two methods using the same architecture as ours but exchange our bisimulation loss with (1) a reconstruction loss (Reconstruction) and (2) contrastive predictive coding (Oord et al., 2018) (Contrastive) to ground the dynamics model and learn a latent representation.

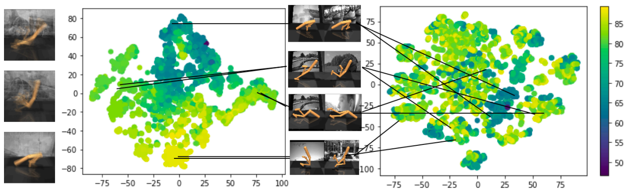

Figure 2. t-SNE of latent spaces learned with DBC (left) and VAE (right) after training has completed, color-coded with predicted state values. Neighboring points in the DBC embedding space have similar states and correspond to observations with the same task-related information (depicted as pairs of images with their corresponding embeddings), whereas no such structure is seen in the embedding space learned by VAE, where the same image pairs are mapped far away from each other. On the left are 3 examples of 10 neighboring points, averaged.

5.1. Control with Background Distraction.

We now benchmark DBC and the previously described baselines on the DeepMind Control (DMC) suite (Tassa et al., 2018) in two settings and nine environments (Figure 7), `finger_spin`, `cheetah_run`, and `walker_walk` and additional envs in the appendix.

Default Setting. Here, the pixel observations have simple backgrounds as shown in Figure 7 (top) with training curves for our DBC and baselines. We see SLAC, a recent state-of-the-art model-based representation learning method that uses reconstruction, generally performs best.

Natural Video Setting. Next, we incorporate natural video from the Kinetics dataset (Kay et al., 2017) as background (Zhang et al., 2018), shown in Figure 7 (bottom). The results confirm our hypothesis: although a number of prior methods can learn effectively in the absence of complex distractors, when distractors are introduced, our non-reconstructive bisimulation based method attains substantially better results. To visualize the representation learned with our bisimulation metric loss function in Equation (1), we use a t-SNE plot (Figure 2).

5.2. Generalization Experiments

We test generalization of DBC in two ways. First, we show that the learned representation can generalize to different types of distractors, by training and testing in different backgrounds. Second, we show that our learned representation can be useful on new reward functions.

Generalizing over backgrounds. In the first experiment, we train on the `simple distractors` setting and evaluate on `natural video`. Figure 3 shows an example of the `simple distractors` setting and performance during training time of two experiments, blue being the zero-shot transfer to the `natural video` setting, and orange the baseline which trains on `natural video`. This result empirically validates that the representations learned by our method are able to effectively learn to ignore the background, *regardless* of what the background is.

Generalizing over reward functions. We evaluate (Figure 3) the generalization capabilities of the learned representation by training SAC with new reward functions `walker_stand` and `walker_run` using the fixed representation learned from `walker_walk`. This is empirical evidence that confirms Theorem 1: if the new reward functions are causally dependent on a subset of the same factors that determine the original reward function, then our representation should be sufficient.

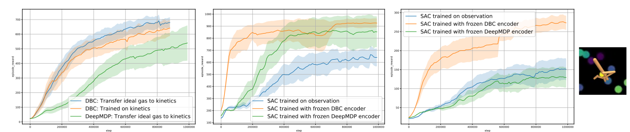

Figure 3. Gen. of a model trained on `simple distractors` env and evaluated on `kinetics` (left). Gen. of an encoder trained on `walker_walk` env and evaluated on `walker_stand` (center) and `walker_run` (right). 10 seeds, 1 std err shaded.

5.3. Autonomous Driving with Visual Redundancy

Real-world control systems such as robotics and autonomous vehicles must contend with a huge variety of task-irrelevant information, such as irrelevant *objects* (e.g. clouds) and irrelevant *details* (e.g. obstacle color). To evaluate DBC on tasks with more realistic observations, we construct a highway driving scenario with photo-realistic visual observations using the CARLA simulator (Dosovitskiy et al., 2017) shown in Figure 4. The agent's goal is to drive as far as possible down CARLA's Town04's figure-8 the highway in 1000 time-steps without colliding into the 20 other moving vehicles or barriers. Our objective function rewards highway progression an penalises collisions: $r_t = \mathbf{v}_{\text{ego}}^\top \hat{\mathbf{u}}_{\text{highway}} \cdot \Delta t - \lambda_i \cdot \text{impulse} - \lambda_s \cdot |\text{steer}|$, where $\mathbf{v}_{\text{ego}}$ is the velocity vector of the ego vehicle, projected onto the highway's unit vector $\hat{\mathbf{u}}_{\text{highway}}$, and multiplied by time discretization $\Delta t = 0.05$ to measure highway progression in meters. Code, additional description of the environment, and install instructions in appendix. **Results** in Figure 4 compare the same baselines as before, except for SLAC which is easily distracted (Figure 1). Instead we used SAC, which does not explicitly learn a representation, but performs surprisingly well from raw images. DeepMDP performs well too, perhaps given its similarly to bisimulation. But, Reconstruction and Contrastive methods again perform poorly with complex images. Figure 14 in appendix provides insight into the representation space as a t-SNE with corresponding observations.

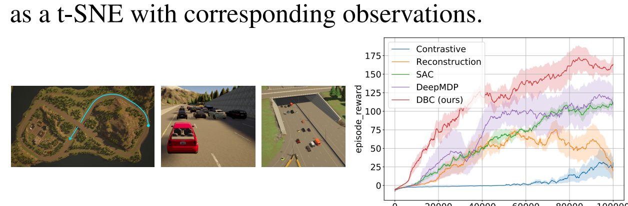

Figure 4. **1st**: Highway loop, **2nd**: third-person view of ego car (red), **3rd**: traffic during episode, **4th**: Performance comparison with 3 seeds. The final performance of our method is 46.8% better than the next best baseline (SAC).

References

Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic Markov decision processes. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. *arXiv:2002.05709 [cs, stat]*, February 2020. URL http://arxiv.org/abs/2002.05709. arXiv: 2002.05709.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Neural Information Processing Systems (NeurIPS)*, pages 4754–4765, 2018.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov decision processes. In *Uncertainty in Artificial Intelligence (UAI)*, pages 162–169, 2004. ISBN 0-9749039-0-6. URL http://dl.acm.org/citation.cfm?id=1036843.1036863.

Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous Markov decision processes. *Society for Industrial and Applied Mathematics*, 40(6):1662–1714, December 2011. ISSN 0097-5397. doi: 10.1137/10080484X. URL https://doi.org/10.1137/10080484X.

Norman Ferns and Doina Precup. Bisimulation metrics are optimal value functions. In *Uncertainty in Artificial Intelligence (UAI)*, pages 210–219, 2014.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *International Conference on Machine Learning (ICML)*, volume 97, pages 2170–2179, Jun 2019.

Robert Givan, Thomas L. Dean, and Matthew Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147:163–223, 2003.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.

Anders Jonsson and Andrew Barto. Causal graph based decomposition of factored MDPs. *J. Mach. Learn. Res.*, 7:2259–2301, December 2006. ISSN 1532-4435.

Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *Computing Research Repository (CoRR)*, 2017. URL http://arxiv.org/abs/1705.06950.

Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010.

Sascha Lange, Martin Riedmiller, and Arne Voigtländer. Autonomous reinforcement learning on raw visual input data in a real world application. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2012. doi: 10.1109/IJCNN.2012.6252823.

K. G. Larsen and A. Skou. Bisimulation through probabilistic testing (preliminary report). In *Symposium on Principles of Programming Languages*, page 344–352. Association for Computing Machinery, 1989. ISBN 0897912942. doi: 10.1145/75277.75307. URL https://doi.org/10.1145/75277.75307.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. arXiv:2003.06417, 2020.

Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for MDPs. In *ISAIM*, 2006.

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, February 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/nature14236.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *arXiv preprint arXiv:1903.08254*, 2019.

Bernhard Schölkopf. Causality for machine learning, 2019.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind control suite. Technical report, DeepMind, January 2018. URL https://arxiv.org/abs/1801.00690.

Jonathan Taylor, Doina Precup, and Prakash Panagaden. Bounding performance loss in approximate MDP homomorphisms. In *Neural Information Processing (NeurIPS)*, pages 1649–1656, 2009. URL http://papers.nips.cc/paper/3423-bounding-performance-loss-in-approximate-mdp-homomorphisms.pdf.

Franck van Breugel and James Worrell. Towards quantitative verification of probabilistic transition systems. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming*, pages 421–432. Springer, 2001. ISBN 978-3-540-48224-6. doi: 10.1007/3-540-48224-5_35.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

Cédric Villani. *Topics in optimal transportation*. American Mathematical Society, 01 2003.

Niklas Wahlström, Thomas Schön, and Marc Deisenroth. From pixels to torques: Policy learning with deep dynamical models. *arXiv preprint arXiv:1502.02251*, 2015.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Neural Information Processing Systems (NeurIPS)*, pages 2728–2736, 2015.

Denis Yarats and Ilya Kostrikov. Soft actor-critic (SAC) implementation in PyTorch. https://github.com/denisyarats/pytorch_sac, 2020.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *arXiv preprint arXiv:1910.01741*, 2019.

Amy Zhang, Yuxin Wu, and Joelle Pineau. Natural environment benchmarks for reinforcement learning. *Computing Research Repository (CoRR)*, abs/1811.06032, 2018. URL http://arxiv.org/abs/1811.06032.

Amy Zhang, Clare Lyle, Shagun Sodhani, Angelos Filos, Marta Kwiatkowska, Joelle Pineau, Yarin Gal, and Doina Precup. Invariant causal prediction for block mdps. In *International Conference on Machine Learning (ICML)*, 2020.

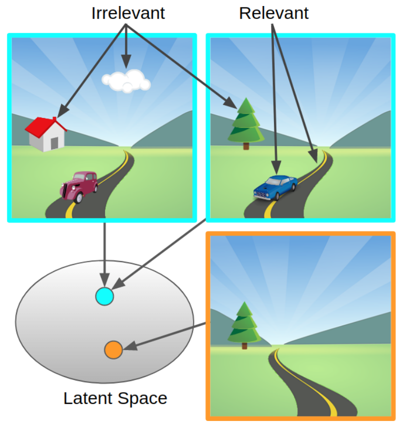

Figure 5. Robust representations of the visual scene should be insensitive to irrelevant objects (e.g., clouds) or details (e.g., car types), and encode two observations equivalently if their relevant details are equal (e.g., road direction and locations of other cars).

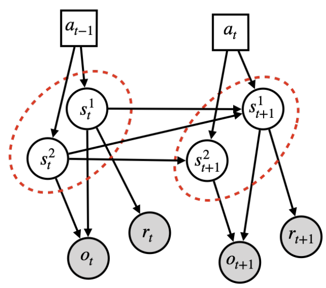

Figure 6. Causal graph of two time steps. Reward depends only on $\mathbf{s}^1$ as a causal parent, but $\mathbf{s}^1$ causally depends on $\mathbf{s}^2$, so AN(R) is the set $\{\mathbf{s}^1, \mathbf{s}^2\}$.

A. Additional Background

We assume the underlying environment is a **Markov decision process** (MDP), described by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{P}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ the probability of transitioning from state $\mathbf{s} \in \mathcal{S}$ to state $\mathbf{s}' \in \mathcal{S}$, and $\gamma \in [0, 1)$ a discount factor. An "agent" chooses actions $\mathbf{a} \in \mathcal{A}$ according to a policy function $\mathbf{a} \sim \pi(\mathbf{s})$, which updates the system state $\mathbf{s}' \sim \mathcal{P}(\mathbf{s}, \mathbf{a})$, yielding a reward $r = \mathcal{R}(\mathbf{s}) \in \mathbb{R}$. The agent's goal is to maximize the expected cumulative discounted rewards by learning a good policy: $\max_\pi \mathbb{E}_{\mathcal{P}}[\sum_{t=0}^{\infty}[\gamma^t \mathcal{R}(\mathbf{s}_t)]$.

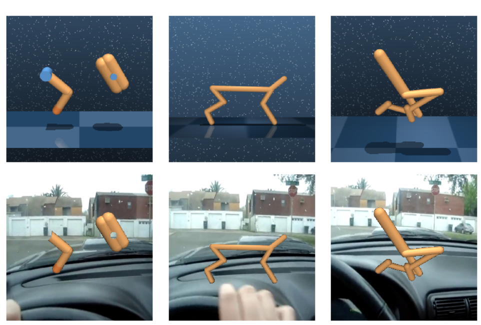

Figure 7. Pixel observations in DMC in the default setting (top row) of the finger spin (left column), cheetah (middle column), and walker (right column), and natural video distractors (bottom row).

B. Additional Figures

C. Related Work

Our work builds on the extensive prior research on bisimulation in MDP state aggregation.

Reconstruction-based Representations. Early works on deep reinforcement learning from images (Lange and Riedmiller, 2010; Lange et al., 2012) used a two-step learning process where first an auto-encoder was trained using reconstruction loss to learn a low-dimensional representation, and subsequently a controller was learned using this representation. This allows effective leveraging of large, unlabeled datasets for learning representations for control. In practice, there is no guarantee that the learned representation will capture useful information for the control task, and significant expert knowledge and tricks are often necessary for these approaches to work. In model-based RL, one solution to this problem has been to jointly train the encoder and the dynamics model end-to-end (Watter et al., 2015; Wahlström et al., 2015) – this proved effective in learning useful task-oriented representations. Hafner et al. (2018) and Lee et al. (2019) learn latent state models using a reconstruction loss, but these approaches suffer from the difficulty of learning accurate long-term predictions and often still require significant manual tuning. Gelada et al. (2019) also propose a latent dynamics model-based method and connect their approach to bisimulation metrics, using a reconstruction loss in Atari. They show that ℓ_2 distance in the DeepMDP representation upper bounds the bisimulation distance, whereas our objective directly learns a representation where distance in latent space *is* the bisimulation metric. Further, their results rely on the assumption that the learned representation is Lipschitz, whereas we show that, by directly learning a bisimilarity-based representation, we guarantee a representation that generates a Lipschitz MDP. We show experimentally that our *non-reconstructive* DBC method is substantially more robust to complex distractors.

Contrastive-based Representations. Contrastive losses are a self-supervised approach to learn useful representations by enforcing similarity constraints between data (van den Oord et al., 2018; Chen et al., 2020). Similarity functions can be provided as domain knowledge in the form of heuristic data augmentation, where we maximize similarity between augmentations of the same data point (Laskin et al., 2020) or nearby image patches (Hénaff et al., 2019), and minimize similarity between different data points. In the absence of this domain knowledge, contrastive representations can be trained by predicting the future (van den Oord et al., 2018). We compare to such an approach in our experiments, and show that DBC is substantially more robust. While contrastive losses do not require reconstruction, they do not inherently have a mechanism to determine downstream task relevance without manual engineering, and when trained only for prediction, they aim to capture all predictable features in the observation, which performs poorly on real images for the same reasons world models do. A better method would be to incorporate knowledge of the downstream task into the similarity function in a data-driven way, so that images that are very different pixel-wise (e.g. lighting or texture changes), can also be grouped as similar w.r.t. downstream objectives.

Bisimulation. Various forms of state abstractions have been defined in Markov decision processes (MDPs) to group states into clusters whilst preserving some property (e.g. the optimal value, or all values, or all action values from each state) (Li et al., 2006). The strictest form, which generally preserves the most properties, is *bisimulation* (Larsen and Skou, 1989). Bisimulation only groups states that are indistinguishable w.r.t. reward sequences output given any action sequence tested. A related concept is bisimulation metrics (Ferns and Precup, 2014), which measure how "behaviorally similar" states are. Ferns et al. (2011) defines the bisimulation metric with respect to continuous MDPs, and propose a Monte Carlo algorithm for learning it using an exact computation of the Wasserstein distance between empirically measured transition distributions. However, this method does not scale well to large state spaces. Taylor et al. (2009) relate MDP homomorphisms to lax probabilistic bisimulation, and define a lax bisimulation metric. They then compute a value bound based on this metric for MDP homomorphisms, where approximately equivalent state-action pairs are aggregated. Most recently, Castro (2020) propose an algorithm for computing *on-policy* bisimulation metrics, but does so directly, without learning a representation. They focus on deterministic settings and the policy evaluation problem. We believe our work is the first to propose a gradient-based method for directly learning a *representation space* with the properties of bisimulation metrics and show that it works in the policy optimization setting.

D. Additional Theorems and Proofs

As evidenced by Definition 2, the bisimulation metric has no direct dependence on the observation space. Pixels can change, but bisimilarity will stay the same. Instead, bisimilarity is grounded in a recursion of future transition probabilities and rewards, which is closely related to the optimal value function. In fact, the bisimulation metric gives tight bounds on the optimal value function with discount factor γ. We show this using the property that the optimal value function is Lipschitz with respect to the bisimulation metric, see Theorem 5 in Appendix (Ferns et al., 2004). This result also implies that the closer two states are in terms of $\tilde{d}$, the more likely they are to share the same optimal actions. This leads us to a generalization bound on the optimal value function of an MDP constructed from a representation space using bisimulation metrics, $||\phi(\mathbf{s}_i) - \phi(\mathbf{s}_j)||_2 := \tilde{d}(\mathbf{s}_i, \mathbf{s}_j)$. We can construct a partition of this space for some $\epsilon > 0$, giving us n partitions where $\frac{1}{n} < (1-c)\epsilon$. We denote ϕ as the encoder that maps from the original state space $\mathcal{S}$ to each ϵ-cluster.

Theorem 2 (Value bound based on bisimulation metrics)**.** *Given an MDP $\bar{\mathcal{M}}$ constructed by aggregating states in an ϵ-neighborhood, and an encoder ϕ that maps from states in the original MDP $\mathcal{M}$ to these clusters, the optimal value functions for the two MDPs are bounded as*

$$|V^*(\mathbf{s}) - V^*(\phi(\mathbf{s}))| \leq \frac{2\epsilon}{(1-\gamma)(1-c)}. \tag{2}$$

Proof. From Theorem 5.1 in Ferns et al. (2004) we have:

$$(1-c)|V^*(\mathbf{s}) - V^*(\phi(\mathbf{s}))| \leq g(\mathbf{s}, \tilde{d}) + \frac{\gamma}{1-\gamma} \max_{u \in S} g(u, \tilde{d})$$

where g is the average distance between a state and all other states in its equivalence class under the bisimulation metric $\tilde{d}$. By specifying a ϵ-neighborhood for each cluster of states we can replace g:

$$\begin{aligned}(1-c)|V^*(\mathbf{s}) - V^*(\phi(\mathbf{s}))| &\leq 2\epsilon + \frac{\gamma}{1-\gamma}2\epsilon \\ |V^*(\mathbf{s}) - V^*(\phi(\mathbf{s}))| &\leq \frac{1}{1-c}(2\epsilon + \frac{\gamma}{1-\gamma}2\epsilon) \\ &= \frac{2\epsilon}{(1-\gamma)(1-c)}.\end{aligned}$$

□

As $\epsilon \to 0$ the optimal value function of the aggregated MDP converges to the original. Further, by defining a learning error for ϕ, $\mathcal{L} := \sup_{\mathbf{s}_i, \mathbf{s}_j \in \mathcal{S}} \big| ||\phi(\mathbf{s}_i) - \phi(\mathbf{s}_j)||_2 - \tilde{d}(\mathbf{s}_i, \mathbf{s}_j) \big|$, we can update the bound in Theorem 2 to incorporate $\mathcal{L}$: $|V^*(\mathbf{s}) - V^*(\phi(\mathbf{s}))| \leq \frac{2\epsilon + 2\mathcal{L}}{(1-\gamma)(1-c)}$.

Theorem 3 (Connections to causal feature sets (Thm 1 in Zhang et al. (2020)))**.** *If we partition observations using the bisimulation metric, those clusters (a bisimulation partition) correspond to the causal feature set of the observation space with respect to current and future reward.*

Theorem 4. *Let $\mathfrak{met}$ be the space of bounded pseudometrics on S and π a policy that is continuously improving. Define $\mathcal{F} : \mathfrak{met} \mapsto \mathfrak{met}$ by*

$$\mathcal{F}(d, \pi)(\mathbf{s}_i, \mathbf{s}_j) = (1-c)|r^\pi_{\mathbf{s}_i} - r^\pi_{\mathbf{s}_j}| + cW(d)(\mathcal{P}^\pi_{\mathbf{s}_i}, \mathcal{P}^\pi_{\mathbf{s}_j}). \tag{3}$$

Then $\mathcal{F}$ has a least fixed point $\tilde{d}$ which is a π^*-bisimulation metric.

Proof. Ideally, to prove this theorem we show that $\mathcal{F}$ is monotonically increasing and continuous, and apply Fixed Point Theorem to show the existence of a fixed point that $\mathcal{F}$ converges to. Unfortunately, we can show that $\mathcal{F}$ under π as π monotonically converges to π^* is *not* also monotonic, unlike the original bisimulation metric setting (Ferns et al., 2004) and the policy evaluation setting (Castro, 2020). We start the iterates $\mathcal{F}^n$ from bottom $\perp$, denoted as $\mathcal{F}^n(\perp)$. In Ferns et al. (2004) the $\max_{a \in A}$ can be thought of as learning a policy between every two pairs of states to maximize their distance, and therefore this distance can only stay the same or grow over iterations of $\mathcal{F}$. In Castro (2020), π is fixed, and under a deterministic MDP it can also be shown that distance between states $d_n(s,t)$ will only expand, not contract as n increases. In the policy iteration setting, however, with π starting from initialization π_0 and getting updated:

$$\pi_i(s) = \arg\max_{a \in A} \sum_{s' \in S} [r^a_{ss'} + \gamma V^{\pi_{i-1}}(s')], \tag{4}$$

there is no guarantee that the distance between two states $d^{\pi_{i-1}}_{n-1}(s,t) < d^{\pi_i}_n(s,t)$ under policy iterations π_{i-1}, π_i and distance metric iterations d_{n-1}, d_n for $i, n \in \mathbb{N}$, which is required for monotonicity.

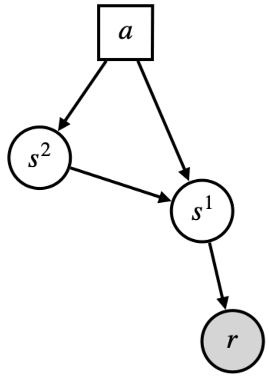

Figure 8. Causal graph of transition dynamics. Reward depends only on $\mathbf{s}^1$ as a causal parent, but $\mathbf{s}^1$ causally depends on $\mathbf{s}^2$, so AN(R) is the set $\{\mathbf{s}^1, \mathbf{s}^2\}$.

Instead, we show that using the policy improvement theorem which gives us

$$V_i^{\pi}(s) \geq V_{i-1}^{\pi}(s), \forall s \in S, \tag{5}$$

π will converge to a fixed point using the Fixed Point Theorem, and taking the result by Castro (2020) that $\mathcal{F}^{\pi}$ has a fixed point for every $\pi \in \Pi$, we can show that a fixed point bisimulation metric will be found with policy iteration. □

Theorem 5 (V^* is Lipschitz with respect to $\tilde{d}$)**.** *Let V^* be the optimal value function for a given discount factor γ. If $c \geq \gamma$, then V^* is Lipschitz continuous with respect to $\tilde{d}$ with Lipschitz constant $\frac{1}{1-c}$, where $\tilde{d}$ is the bisimilarity metric.*

$$|V^*(\mathbf{s}_i) - V^*(\mathbf{s}_j)| \leq \frac{1}{1-c}\tilde{d}(\mathbf{s}_i, \mathbf{s}_j). \tag{6}$$

See Theorem 5.1 in Ferns et al. (2004) for proof.

Theorem 1. *Given an encoder $\phi : O \mapsto S$ that maps observations to a latent bisimulation metric representation where $||\phi(\mathbf{s}_i) - \phi(o_j)||_2 := \tilde{d}(o_i, o_j)$, S encodes information about all the causal ancestors of the reward $AN(R)$.*

Proof. We assume a MDP with a state space $\mathcal{S} := \{S^1, ..., S^k\}$ that can be factorized into k variables with 1-step causal transition dynamics described by a causal graph $\mathcal{G}$ (example in Figure 8). We break the proof up into two parts: 1) show that if a factor $S^i \notin AN(R)$ changes, the bisimulation distance between the original state $\mathbf{s}$ and the new state $\mathbf{s}'$ is 0. and 2) show that if a factor $S^j \in AN(R)$ changes, the bisimulation distance can be > 0.

1) If $S^i \notin AN(R)$, an intervention on that factor does not affect current or future reward.

$$\begin{aligned}\tilde{d}(\mathbf{s}, \mathbf{s}') &= \max_{a \in A}(1-c)|r_{\mathbf{s}}^{\mathbf{a}} - r_{\mathbf{s}'}^{\mathbf{a}}| + cW(\tilde{d})(\mathcal{P}_{\mathbf{s}}^{\mathbf{a}}, \mathcal{P}_{\mathbf{s}'}^{\mathbf{a}}) \\ &= \max_{a \in A} cW(\tilde{d})(\mathcal{P}_{\mathbf{s}}^{\mathbf{a}}, \mathcal{P}_{\mathbf{s}'}^{\mathbf{a}}) \quad \text{s and s}' \text{ have the same reward.}\end{aligned}$$

If S^i does not affect future reward, then states $\mathbf{s}_i$ and $\mathbf{s}_j$ will have the same future reward conditioned on all future actions. This gives us

$$\tilde{d}(\mathbf{s}, \mathbf{s}') = 0.$$

2) If there is an intervention on $S^j \in AN(R)$ then current and/or future reward can change. If current reward changes, then we already have $\max_{a \in A}(1-c)|r_{\mathbf{s}}^{\mathbf{a}} - r_{\mathbf{s}'}^{\mathbf{a}}| > 0$, giving us $\tilde{d}(\mathbf{s}, \mathbf{s}') > 0$. If only future reward changes, then those future states will have nonzero bisimilarity, and $\max_{a \in A} cW(\tilde{d})(P_{\mathbf{s}}^{\mathbf{a}}, P_{\mathbf{s}'}^{\mathbf{a}}) > 0$, giving us $\tilde{d}(\mathbf{s}, \mathbf{s}') > 0$. □

E. Additional Results

We add an additional setting as shown in Figure 9. We incorporate simple, easy to predict distractors into the background – different colored balls that obey the dynamics of an ideal gas, no attraction or repulsion between each pair of objects.

In Figure 10 we show performance on the default setting on 9 different environments from DMC. Figures 11 and 12 give performance on the simple distractors and natural video settings for all 9 environments.

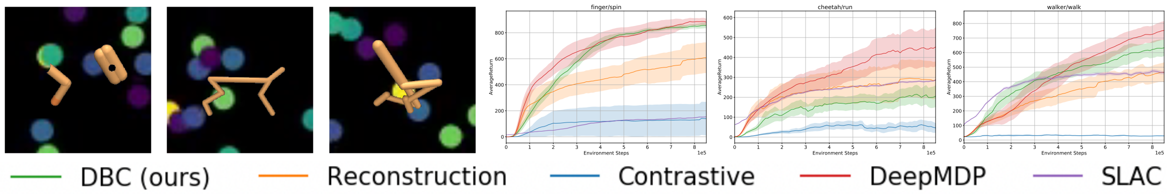

Figure 9. **Left observations**: Pixel observations in DMC in the simple distractor setting of the finger spin (left column), cheetah (middle column), and walker (right column). **Right training curves**: Results for DBC in comparison to baselines with reconstruction loss, contrastive loss, DeepMDP, and SLAC on 10 seeds with 1 standard error shaded in the default setting. The grid-location of each graph corresponds to the grid-location of each observation.

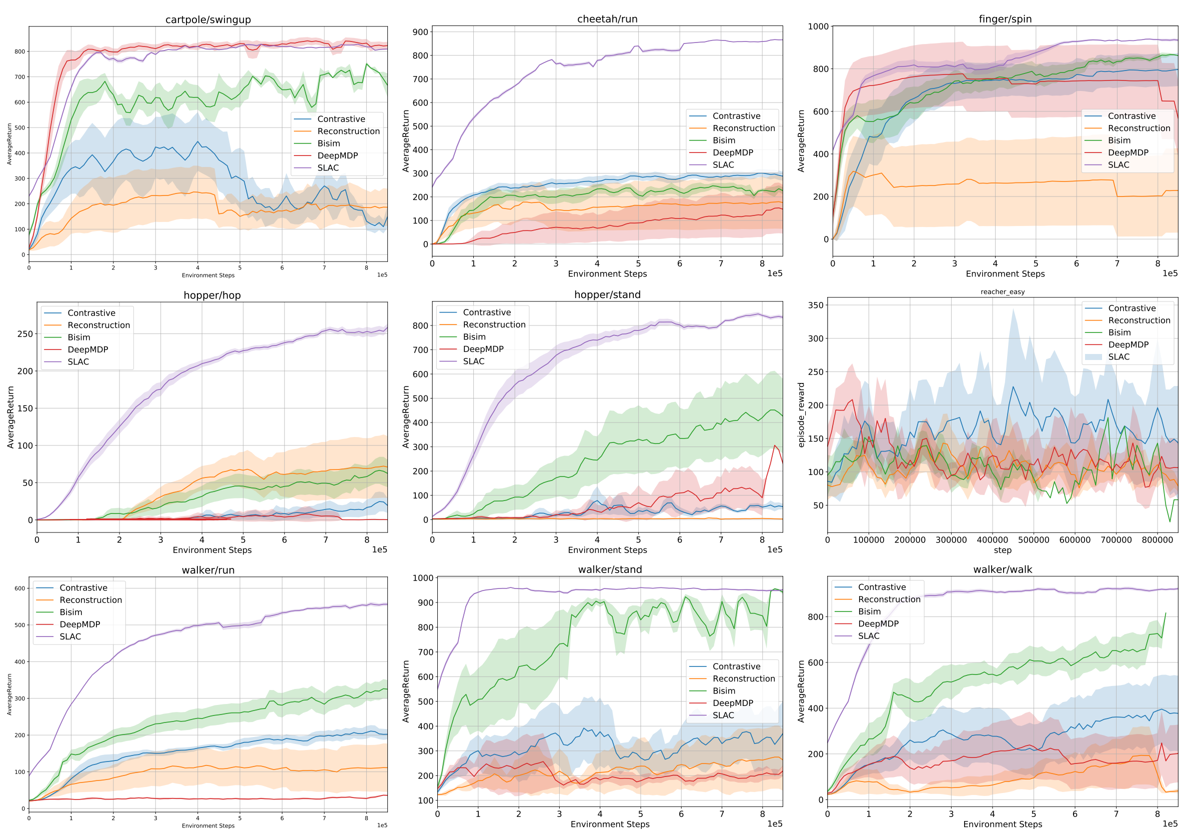

Figure 10. Results for DBC in the default setting, in comparison to baselines with reconstruction loss, contrastive loss, and SLAC on 10 seeds with 1 standard error shaded.

Figure 11. Results for DBC in the simple distractors setting, in comparison to baselines with reconstruction loss, contrastive loss, DeepMDP, and SLAC on 10 seeds with 1 standard error shaded.

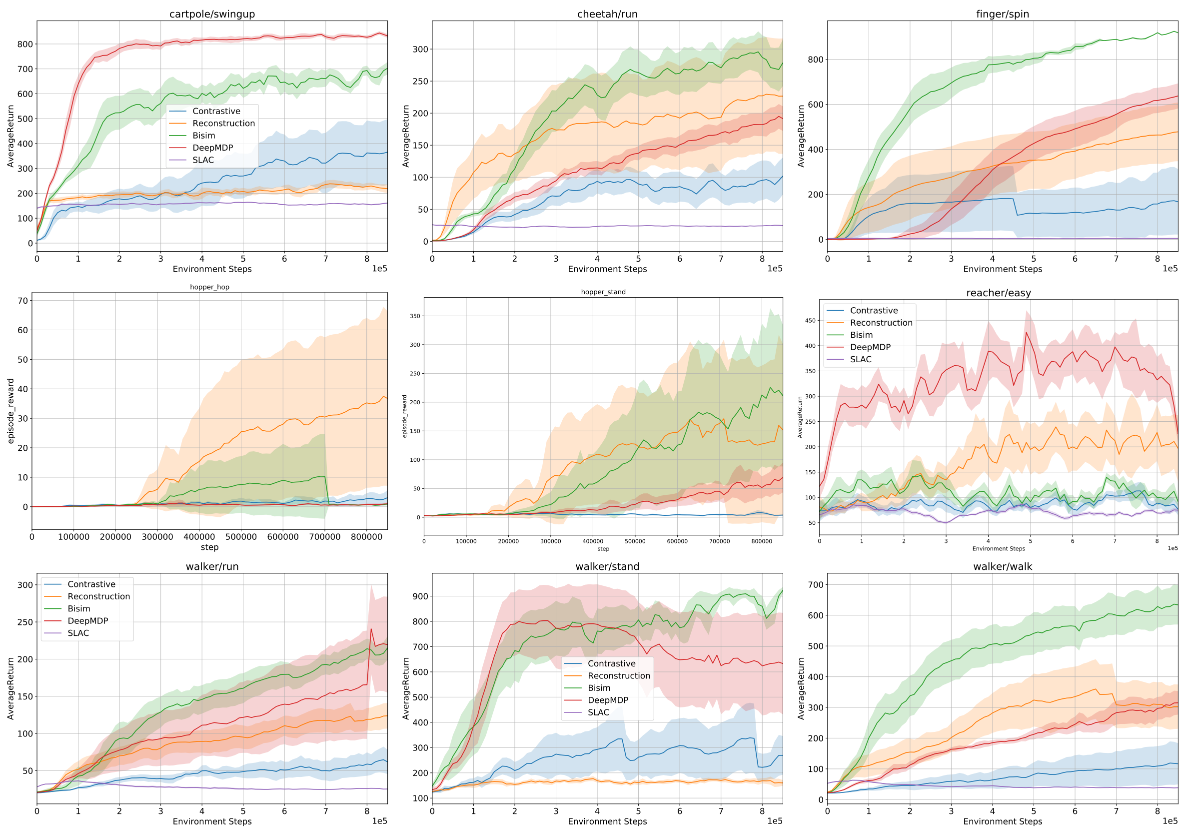

Figure 12. Results for our bisimulation metric method in the natural video setting, in comparison to baselines with reconstruction loss, contrastive loss, DeepMDP, and SLAC on 10 seeds with 1 standard error shaded.

E.1. Comparison with other Bisimulation Encoders

Even though the purpose of bisimulation metrics by Castro (2020) is learning distances d, not representation spaces $\mathcal{Z}$, it nevertheless implements d with function approximation: $d(\mathbf{s}_i, \mathbf{s}_j) = \psi\big(\phi(\mathbf{s}_i), \phi(\mathbf{s}_j)\big)$ by encoding observations with ϕ before computing distances with ψ, trained as:

$$J(\phi, \psi) = \Big(\psi\big(\phi(\mathbf{s}_i), \phi(\mathbf{s}_j)\big) - |\mathcal{R}(\mathbf{s}_i) - \mathcal{R}(\mathbf{s}_j)| - \gamma\hat{\psi}\Big(\hat{\phi}\big(\mathcal{P}(\mathbf{s}_i, \pi(\mathbf{s}_i))\big), \hat{\phi}\big(\mathcal{P}(\mathbf{s}_j, \pi(\mathbf{s}_j))\big)\Big)\Big)^2, \tag{7}$$

where $\hat{\phi}$ and $\hat{\psi}$ are target networks. A natural question is: how does the encoder ϕ above perform in control tasks? We combine ϕ above with our policy in Algorithm 2 and use the same network ψ (single hidden layer 729 wide). Figure 13 shows representations from Castro (2020) can learn control, but our method learns faster. Further, our method is simpler: by comparing Equation (7) to Equation (1), our method uses the ℓ_1 distance between the encoding instead of introducing an addition network ψ.

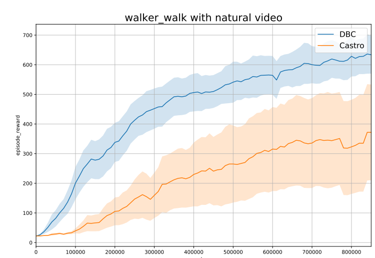

Figure 13. Bisim. results

E.2. Autonomous Driving with Visual Redundancy

Real-world control systems such as robotics and autonomous vehicles must contend with a huge variety of task-irrelevant information, such as irrelevant *objects* (e.g. clouds) and irrelevant *details* (e.g. obstacle color). To evaluate DBC on tasks with more realistic observations, we construct a highway driving scenario with photo-realistic visual observations using the CARLA simulator (Dosovitskiy et al., 2017) shown in Figure 4. The agent's goal is to drive as far as possible down CARLA's Town04's figure-8 the highway in 1000 time-steps without colliding into the 20 other moving vehicles or barriers. Our objective function rewards highway progression an penalises collisions: $r_t = \mathbf{v}_{\text{ego}}^\top \hat{\mathbf{u}}_{\text{highway}} \cdot \Delta t - \lambda_i \cdot \text{impulse} - \lambda_s \cdot |\text{steer}|$, where $\mathbf{v}_{\text{ego}}$ is the velocity vector of the ego vehicle, projected onto the highway's unit vector $\hat{\mathbf{u}}_{\text{highway}}$, and multiplied by time discretization $\Delta t = 0.05$ to measure highway progression in meters. Collisions result in impulses $\in \mathbb{R}^+$, measured in Newton-seconds. We found a steering penalty steer $\in [-1, 1]$ helped, and used weights $\lambda_i = 10^{-4}$ and $\lambda_s = 1$. While more specialized objectives exist like lane-keeping, this experiment's purpose is to compare representations with observations more characteristic of real robotic tasks. We use five cameras on the vehicle's roof, each with 60 degree views. By concatenating the images together, our vehicle has a 300 degree view, observed as 84×420 pixels. Code and install instructions in appendix.

Results in Figure 4 compare the same baselines as before, except for SLAC which is easily distracted (Figure 1). Instead we used SAC, which does not explicitly learn a representation, but performs surprisingly well from raw images. DeepMDP performs well too, perhaps given its similarly to bisimulation. But, Reconstruction and Contrastive methods again perform poorly with complex images. Figure 14 provides insight into the representation space as a t-SNE with corresponding observations. Each run took 12 hours on a GTX 1080 GPU.

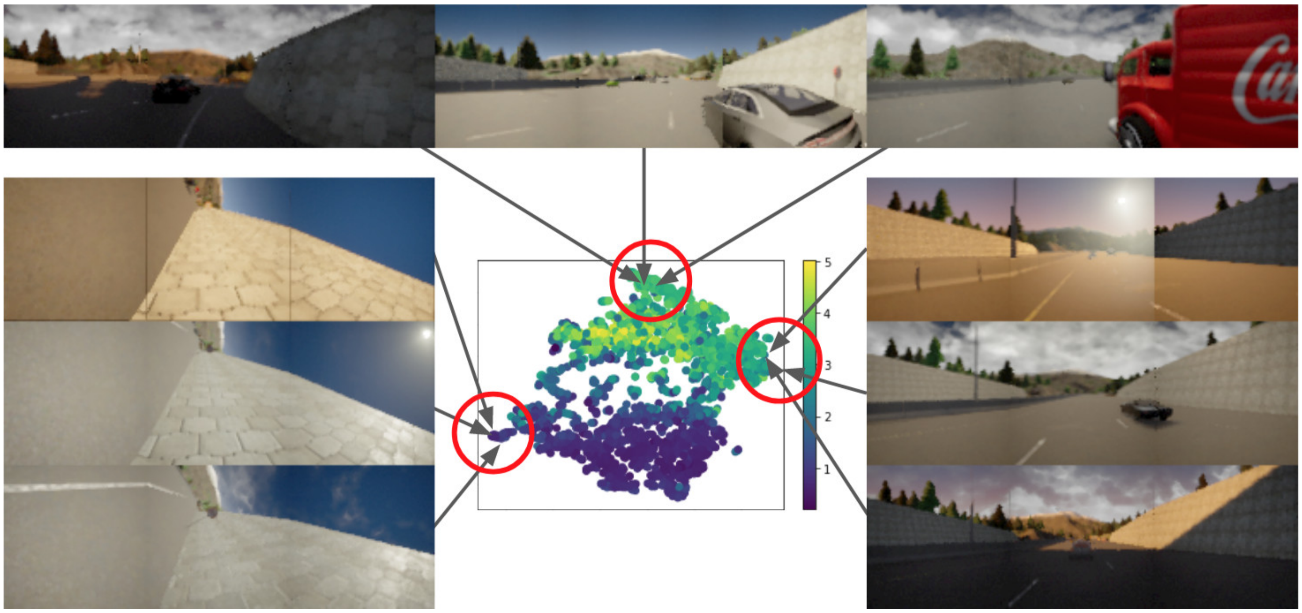

Figure 14. A t-SNE diagram of encoded first-person driving observations after 10k training steps of Algorithm 1, color coded by value (V in Algorithm 2). **Top**: the learned representation identifies an obstacle on the right side. Whether that obstacle is a dark wall, bright car, or truck is task-irrelevant: these states are behaviourally equivalent. **Left**: the ego vehicle has flipped onto its left side. The different wall colors, due to a setting sun, is irrelevant: all states are equally stuck and low-value (purple t-SNE color). **Right**: clear highway driving. Clouds and sun position are irrelevant.

F. Implementation Details

We use the same encoder architecture as in Yarats et al. (2019), which is an almost identical encoder architecture as in Tassa et al. (2018), with two more convolutional layers to the convnet trunk. The encoder has kernels of size 3×3 with 32 channels for all the convolutional layers and set stride to 1 everywhere, except of the first convolutional layer, which has stride 2, and interpolate with `ReLU` activations. Finally, we add `tanh` nonlinearity to the 50 dimensional output of the fully-connected layer.

For the reconstruction method, the decoder consists of a fully-connected layer followed by four deconvolutional layers. We use `ReLU` activations after each layer, except the final deconvolutional layer that produces pixels representation. Each deconvolutional layer has kernels of size 3×3 with 32 channels and stride 1, except of the last layer, where stride is 2.

The dynamics and reward models are both MLPs with two hidden layers with 200 neurons each and `ReLU` activations.

Soft Actor Critic (SAC) (Haarnoja et al., 2018) is an off-policy actor-critic method that uses the maximum entropy framework for soft policy iteration. At each iteration, SAC performs soft policy evaluation and improvement steps. The policy evaluation step fits a parametric soft Q-function $Q(\mathbf{x}_t, \mathbf{a}_t)$ using transitions sampled from the replay buffer $\mathcal{D}$ by minimizing the soft Bellman residual,

$$J(Q) = \mathbb{E}_{(\mathbf{x}_t, \mathbf{x}_t, r_t, \mathbf{x}_{t+1}) \sim \mathcal{D}} \left[\left(Q(\mathbf{x}_t, \mathbf{a}_t) - r_t - \gamma \bar{V}(x_{t+1}) \right)^2 \right].$$

The target value function $\bar{V}$ is approximated via a Monte-Carlo estimate of the following expectation,

$$\bar{V}(x_{t+1}) = \mathbb{E}_{a_{t+1} \sim \pi} \big[\bar{Q}(x_{t+1}, a_{t+1}) - \alpha \log \pi(\mathbf{a}_{t+1} | \mathbf{x}_{t+1}) \big],$$

where $\bar{Q}$ is the target soft Q-function parameterized by a weight vector obtained from an exponentially moving average of the Q-function weights to stabilize training. The policy improvement step then attempts to project a parametric policy $\pi(\mathbf{a}_t | \mathbf{x}_t)$ by minimizing KL divergence between the policy and a Boltzmann distribution induced by the Q-function, producing the following objective,

$$J(\pi) = \mathbb{E}_{\mathbf{x}_t \sim \mathcal{D}} \Big[\mathbb{E}_{\mathbf{a}_t \sim \pi} [\alpha \log(\pi(\mathbf{a}_t | \mathbf{x}_t)) - Q(\mathbf{x}_t, \mathbf{a}_t)] \Big].$$

We modify the Soft Actor-Critic PyTorch implementation by Yarats and Kostrikov (2020) and augment with a shared encoder between the actor and critic, the general model f_s and task-specific models f_η^e. The forward models are multi-layer perceptions with ReLU non-linearities and two hidden layers of 200 neurons each. The encoder is a linear layer that maps to a 50-dim hidden representation. The hyperparameters used for the RL experiments are in Table 1.

Parameter name	Value
Replay buffer capacity	1000000
Batch size	128
Discount γ	0.99
Optimizer	Adam
Critic learning rate	10^{-5}
Critic target update frequency	2
Critic Q-function soft-update rate τ_Q	0.005
Critic encoder soft-update rate τ_ϕ	0.005
Actor learning rate	10^{-5}
Actor update frequency	2
Actor log stddev bounds	$[-5, 2]$
Encoder learning rate	10^{-5}
Decoder learning rate	10^{-5}
Decoder weight decay	10^{-7}
Temperature learning rate	10^{-4}
Temperature Adam's β_1	0.9
Init temperature	0.1

Table 1. A complete overview of used hyper parameters.

G. README for Running Code

1. *Download CARLA 0.9.6*:

```
wget https://carla-releases.s3.eu-west-3.amazonaws.com/Linux/CARLA_0.9.6.tar.gz
tar xvzf CARLA_0.9.6.tar.gz
```

2. *Place inside* `bisim_metric` *directory*:

```
mv -r CARLA_0.9.6 /home/USER/bisim_metric
```

3. *Install Python dependencies*:

```
pip install pygame
pip install networkx
```

4. *Add paths to* `.bashrc`:

```
export PYTHONPATH=$PYTHONPATH:/home/USER/bisim_metric/CARLA_0.9.6/PythonAPI
export PYTHONPATH=$PYTHONPATH:/home/USER/bisim_metric/CARLA_0.9.6/PythonAPI/carla
export PYTHONPATH=$PYTHONPATH:/home/USER/bisim_metric/CARLA_0.9.6/PythonAPI/carla/dist/carla-0.9.6-py3.5-linux-x86_64.egg
```

4. *In one terminal execute*:

```
cd /home/USER/bisim_metric/CARLA_0.9.6
bash CarlaUE4.sh -fps 20
```

5. *In another terminal execute*:

```
cd /home/USER/bisim_metric
./run_local_carla.sh --replay_buffer_capacity 100000 --agent bisim --transition_model_type probabilistic
```

(if you receive ”out of memory” errors, then decrease `replay_buffer_capacity` until the error disappears.