LEARNING ACTIONABLE REPRESENTATIONS WITH GOAL-CONDITIONED POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Representation learning is a central challenge across a range of machine learning areas. In reinforcement learning, effective and functional representations have the potential to tremendously accelerate learning progress and solve more challenging problems. Most prior work on representation learning has focused on generative approaches, learning representations that capture all the underlying factors of variation in the observation space in a more disentangled or well-ordered manner. In this paper, we instead aim to learn functionally salient representations: representations that are not necessarily complete in terms of capturing all factors of variation in the observation space, but rather aim to capture those factors of variation that are important for decision making – that are "actionable." These representations are aware of the dynamics of the environment, and capture only the elements of the observation that are necessary for decision making rather than all factors of variation, eliminating the need for explicit reconstruction. We show how these learned representations can be useful to improve exploration for sparse reward problems, to enable long horizon hierarchical reinforcement learning, and as a state representation for learning policies for downstream tasks. We evaluate our method on a number of simulated environments, and compare it to prior methods for representation learning, exploration, and hierarchical reinforcement learning.

1 INTRODUCTION

Representation learning refers to a transformation of an observation, such as a camera image or state observation, into a form that is easier to manipulate to deduce a desired output or perform a downstream task, such as prediction or control. In reinforcement learning (RL) in particular, effective representations are ones that enable generalizable controllers to be learned quickly for challenging and temporally extended tasks. While end-to-end representation learning with full supervision has proven effective in many scenarios, from supervised image recognition (Krizhevsky et al., 2012) to vision-based robotic control (Levine et al., 2015), devising representation learning methods that can use unlabeled data or experience effectively remains an open problem.

Much of the prior work on representation learning in RL has focused on generative approaches. Learning these models is often challenging because of the need to model the interactions of *all* elements of the state. We instead aim to learn functionally salient representations: representations that are not necessarily complete in capturing all factors of variation in the observation space, but rather aim to capture factors of variation that are relevant for decision making – that are actionable.

How can we learn a representation that is aware of the dynamical structure of the environment? We propose that a basic understanding of the world can be obtained from a *goal-conditioned policy*, a policy that can knows how to reach arbitrary goal states from a given state. Learning how to execute shortest paths between all pairs of states suggests a deep understanding of the environment dynamics, and we hypothesize that a representation incorporating the knowledge of a goal-conditioned policy can be readily used to accomplish more complex tasks. However, such a policy does not provide a readily usable state representation, and it remains to choose *how* an effective state representation should be extracted. We want to extract those factors of the state observation that are critical for deciding which action to take. We can do this by comparing which actions a goal-conditioned policy takes for two different goal states. Intuitively, if two goal states require different actions, then they are functionally different and vice-versa. This principle is illustrated in the diagram in Figure 1.

Based on this principle, we propose actionable representations for control (ARC), representations in which Euclidean distances between states correspond to expected differences between actions taken to reach them. Such representations emphasize factors in the state that induce significant differences in the corresponding actions, and de-emphasize those features that are irrelevant for control.

While learning a goal-conditioned policy to extract such a representation might itself represent a daunting task, it is worth noting that such a policy can be learned without any knowledge of downstream tasks, simply through unsupervised exploration of the environment. It is reasonable to postulate that, without active exploration, *no* representation learning method can possibly acquire a dynamics-aware representation, since understanding the dynamics requires experiencing transitions and interactions, rather than just observations of valid states. As we demonstrate in our experiments, representations extracted from goal-conditioned policies can be used to better learn more challenging tasks than simple goal reaching, which cannot be easily contextualized by goal states. The process of learning goal-conditioned policies can also be made recursive, so that the actionable representations learned from one goal-conditioned policy can be used to quickly learn a better one.

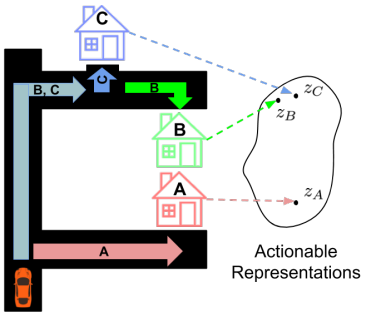


Figure 1: Actionable representations: 3 houses A, B, C can only be reached by indicated roads. The actions taken to reach A, B, C are shown by arrows. Although A, B are very close in space, they are functionally different. The car has to take a completely different road to reach A, compared to B and C. Representations z_A, z_B, z_C learn these functional differences to differentiate A from B and C, while keeping B and C close.

Actionable representations for control are useful for a number of downstream tasks: as representations for task-specific policies, as representations for hierarchical RL, and to construct well-shaped reward functions. We show that ARCs enable these applications better than representations that are learned using unsupervised generative models, predictive models, and other prior representation learning methods. We analyze structure of the learned representation, and compare the performance of ARC with a number of prior methods on downstream tasks in simulated robotic domains such as wheeled locomotion, legged locomotion, and robotic manipulation.

2 Preliminaries

Our work builds on the framework of goal-conditioned RL to extract representations.

Goal-conditioned reinforcement learning. In RL, the goal is to learn a policy $\pi_\theta(a_t|s_t)$ that maximizes the expected return $R_t = \mathbb{E}_{\pi_\theta}[\sum_t r_t]$. Typically, RL learns a single task that optimizes for a particular reward function. If we instead would like to train a policy that can accomplish a variety of tasks, we might instead train a policy that is conditioned on another input – a goal. When the different tasks directly correspond to different states, this amounts to conditioning the policy π on both the current and goal state. The policy $\pi_\theta(a_t|s_t, g)$ is trained to reach goals from the state space $g \sim \mathcal{S}$, by optimizing $\mathbb{E}_{g\sim S}[\mathbb{E}_{\pi_\theta(a|s,g)}(R_g))]$, where R_g is a reward for reaching the goal g.

Maximum entropy RL. Maximum entropy RL algorithms modify the RL objective, and instead learns a policy to maximize the reward as well as the entropy of the policy (Haarnoja et al., 2017; Todorov, 2006), according to $\pi^\star = \arg\max_\pi E_\pi[r(s,a)] + \mathcal{H}(\pi)$. In contrast to standard RL, where optimal policies in fully observed environments are deterministic, the solution in maximum entropy RL is a stochastic policy, where the entropy reflects the sensitivity of the rewards to the action: when the choice of action has minimal effect on future rewards, actions are more random, and when the choice of action is critical, the actions are more deterministic. In this way, the action distributions for a maximum entropy policy carry more information about the dynamics of the task.

3 Learning Actionable Representations

In this work, we extract a representation that can distinguish states based on actions required to reach them, which we term an actionable representation for control (ARC). In order to learn state

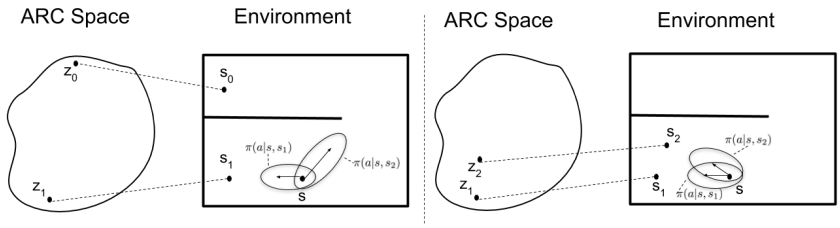


Figure 2: An illustration of actionable representations. For a pair of states s_1, s_2, the divergence between the goal-conditioned action distributions they induce defines the actionable distance D_{Act}, which in turn is used to learn representation ϕ.

representations ϕ that can capture the elements of the state which are important for decision making, we first consider defining actionable distances $D_{\text{Act}}(s_1, s_2)$ between states. Actionable distances are distances between states that capture the differences between the actions required to reach the different states, thereby implicitly capturing dynamics. If actions required for reaching state s_1 are very different from the actions needed for reaching state s_2, then these states are *functionally* different, and should have large actionable distances. This subsequently allows us to extract a feature representation($\phi(s)$) of state, which captures elements that are important for decision making.

To formally define actionable distances, we build on the framework of goal-conditioned RL. We assume that we have already trained a maximum entropy goal-conditioned policy $\pi_\theta(a|s, g)$ that can start at an arbitrary state $s_0 \in S$ in the environment, and reach a goal state $s_g \in \mathcal{S}$. Although this is a significant assumption, we will discuss later how this is in fact reasonable in many settings. We can extract actionable distances by examining how varying the goal state affects action distributions for goal-conditioned policies. Formally, consider two different goal states s_1 and s_2. At an intermediate state s, the goal-conditioned policy induces different action distributions $\pi_\theta(a|s, s_1)$ and $\pi_\theta(a|s, s_2)$ to reach s_1 and s_2 respectively. If these distributions are similar over many intermediate states s, this suggests that these states are functionally similar, while if these distributions are different, then the states must be functionally different. This motivates a definition for actionable distances D_{Act} as

$$D_{\text{Act}}(s_1, s_2) = \mathbb{E}_s \left[D_{KL}(\pi(a|s, s_1)||\pi(a|s, s_2)) + D_{KL}(\pi(a|s, s_2)||\pi(a|s, s_1)) \right]. \tag{1}$$

The distance consists of the expected divergence over all initial states s. If we focus on a subset of states, the distance may not capture action differences induced elsewhere, and can miss functional differences between states. Since maximum entropy policies learn *unique* optimal stochastic policies, the actionable distance is well-defined and unambiguous. Furthermore, because maximum entropy policies capture sensitivity of the value function, goals are similar under actionable distances if they require the same action *and* they are equally "easy" to reach.

We can use D_{Act} to extract an actionable representation of state. To learn this representation $\phi(s)$, we optimize ϕ such that Euclidean distance between states in representation space corresponds to actionable distances D_{Act} between them. This optimization yields good representations of state because it emphasizes the functionally relevant elements of state, which significantly affect the actionable distance, while suppressing the less functionally relevant elements of state. The optimization problem is:

$$\min_\phi \mathbb{E}_{s_1, s_2} \left[\|\phi(s_1) - \phi(s_2)\|_2 - D_{\text{Act}}(s_1, s_2) \right]^2 \tag{2}$$

This objective yields representations where Euclidean distances are meaningful. This is not necessarily true in the state space or in generative representations (Section 6.3). These representations are meaningful for several reasons. First, since we are leveraging a goal-conditioned policy, they are aware of dynamics and are able to capture local connectivity of the environment. Secondly, the representation is optimized so that it captures only the *functionally* relevant elements of state.

A natural question to ask is whether needing a goal-conditioned policy is too strong of a prerequisite. However, it is worth noting that this goal-conditioned policy can be trained entirely through unsupervised interaction with the environment (using an off-policy RL algorithm), without any downstream

task reward function (Nair et al., 2018). In many realistic situations, unsupervised experience is much cheaper, and the same representation can be reused for many downstream tasks. Furthermore, it is likely not possible to acquire a functionality-aware state representation *without* some sort of active environment interaction, since dynamics can only be understood by observing outcomes of actions, rather than individual states. In our experiments, we show that ARCs can generalize to new parts of the state space beyond the goal-conditioned policy that was used to train them.

4 USING ACTIONABLE REPRESENTATIONS FOR DOWNSTREAM TASKS

4.1 FEATURES FOR LEARNING POLICIES

Many reward functions in the environment cannot be represented as reaching a single goal state. For such tasks, we hypothesize that using the ARC representation as input for a policy or value function can make the learning problem easier. We can learn a policy for a downstream task, of the form $\pi_\theta(a|\phi(s))$, using the representation $\phi(s)$ instead of state s. The implicit understanding of the environment dynamics in the learned representation prioritizes the parts of the state that are most important for learning, and enables quicker learning for these tasks as we see in Section 6.5.

4.2 REWARD SHAPING

We can use ARC to construct better-shaped reward functions. It is common in continuous control to define rewards in terms of some distance to a desired state, oftentimes using Euclidean distance (Schulman et al., 2015; Lillicrap et al., 2015). However, Euclidean distance in state space is not necessarily a meaningful metric of *functional* proximity. ARC provides a better metric, since it directly accounts for reachability. We can use the actionable representation to define better-shaped reward functions for downstream tasks. We define a shaped reward $D(s_1, s_2)$ of this form to be the Euclidean distance between two states in ARC space: $D(s_1, s_2) = ||\phi(s_1) - \phi(s_2)||_2$: for example, on a goal-reaching task $r(s) = -D(s, s_g) = -||\phi(s) - \phi(s_g)||_2$. This allows us to explore and learn policies even in the presence of sparse reward functions.

One may wonder whether, instead of using ARCs for reward shaping, we might directly use the goal-conditioned policy to reach a particular goal. As we will illustrate in Section 6.4, the representation typically generalizes better than the goal-conditioned policy. Training powerful goal-conditioned policies is quite challenging, especially when we need to extrapolate to new parts of the state space. We observe that ARC exhibits better generalization, and can provide for effective reward shaping for goals that are very difficult to reach with the goal-conditioned policy.

4.3 HIERARCHICAL REINFORCEMENT LEARNING

We can use a goal-conditioned policy as a low-level controller for hierarchical tasks that may not be expressible as a single goal-reaching objective. Such tasks attempt to learn a high-level controller $\pi_{\text{meta}}(g|s)$ via RL that produces desired goal states for a goal-conditioned policy to reach sequentially as described in (Nachum et al., 2018). The high-level controller suggests a goal in state space, the goal conditioned policy executes for several time-steps till it achieves the goal, following which the high level controller picks a goal again. For many tasks, naively training such a high-level controller is unlikely to perform well, since such a controller must disentangle the relevant attributes in the goal for long horizon reasoning.

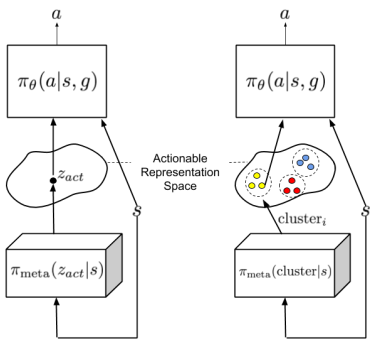


Figure 3: Hierarchical RL with ARC. **Left:** Directly commanding in ARC space **Right:** Commanding a cluster in ARC space

We can use ARC representations as a better goal space for high-level controllers, since it de-emphasizes components of the goal space irrelevant for determining the optimal action. A high-level controller π_{meta} producing outputs in ARC space does not need to find salient features from the goal space, making hierarchical RL easier.

We show that for tasks such as waypoint navigation, using ARC as a hierarchical goal space enables significant improvement.

We can also more directly use the learned structure in the representation space. Since ARC captures the topology of the environment, clusters in ARC space often correspond to semantically meaningful state abstractions. We utilize these clusters, with the intuition that a meta-controller searching in "cluster space" should learn faster. Clustering goals with Euclidean distance in ARC space, we can define a low-level controller which when given a cluster as input, samples a goal from this cluster, and uses it as a goal for the goal-conditioned policy. We train a meta-policy to output clusters, instead of states: $\pi_{meta}(\text{cluster}|s)$. We see that for hierarchical tasks with less granular reward functions such as room navigation, performing RL in "cluster space" induced by ARC outperform cluster spaces induced by other representations.

5 Related Work

The capability to learn effective representations is a major advantage of deep neural network models. These representations can be acquired implicitly, through end-to-end training (Goodfellow et al., 2016), or explicitly, by formulating and optimizing a representation learning objective. A classic approach to representation learning is generative modeling, where a latent variable model is trained to model the data distribution, and the latent variables are then used as a representation (Rasmus et al., 2015; Dumoulin et al., 2016; Kingma & Welling, 2013; Finn et al., 2015; Ghadirzadeh et al., 2017; Curran et al., 2015; Goroshin et al., 2015; Higgins et al., 2017). In the context of control and sequence models, generative models have also been proposed to model transitions (Watter et al., 2015; Assael et al., 2015; Zhang et al., 2018b; Kurutach et al., 2018). While generative models are general and principled, they must not only explain the entirety of the input observation, but must also generate it. Several methods perform representation learning without generation, often based on contrastive losses (Sermanet et al., 2018; van den Oord et al., 2018; Belghazi et al., 2018; Chopra et al., 2005; Weinberger & Saul, 2009). While these methods avoid generation, they either still require modeling of the entire input, or utilize heuristics that encode user-defined information. In contrast, ARCs are directly trained to focus on decision-relevant features of input, providing a broadly applicable objective that is still selective about which aspects of input to represent.

In the context of RL and control, representation learning methods have been used for many downstream applications (Lesort et al., 2018), including representing value functions (Barreto et al., 2016) and building models (Watter et al., 2015; Assael et al., 2015; Zhang et al., 2018b). Our approach is complementary: it can also be applied to these applications. Several works have sought to learn representations that are specifically suited for physical dynamical systems (Jonschkowski & Brock, 2015) and that use interaction to build up dynamics-aware features (Bengio et al., 2017). In contrast to Jonschkowski & Brock (2015), our method does not attempt to encode all physically-relevant features of state, only those relevant for choosing actions. In contrast to Bengio et al. (2017), our approach does not try to determine which features of the state can be independently controlled, but rather which features are relevant for *choosing* controls. Related methods learn features that are predictive of actions based on pairs of sequential states (so-called *inverse models*) (Agrawal et al., 2016; Pathak et al., 2017; Zhang et al., 2018a). Unlike ARC, which is learned from a policy performing long-horizon control, inverse models are not obliged to represent all relevant features for multi-step control, and suffer from greedy reasoning.

6 Experiments

The aim of our experimental evaluation is to study the following research questions:

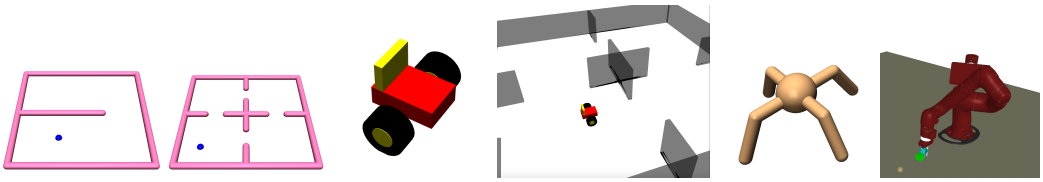
1. Can we learn ARCs for multiple continuous control environments? What are the properties of these learned representations?
2. Can ARCs be used as feature representations for learning policies quickly on new tasks?
3. Can reward shaping with ARCs enable faster learning?
4. Do ARCs provide an effective mechanism for hierarchical RL?

Experiment details and videos at `https://sites.google.com/view/arc-reps`.

6.1 DOMAINS

We study six simulated environments (Fig 4): two 2D navigation tasks, two wheeled locomotion tasks, legged locomotion, and object pushing with a robotic gripper. The 2D navigation domains consist of either a room with a central divider wall or four rooms. Wheeled locomotion involves a two-wheeled differential drive robot, either in free space or with four rooms. For legged locomotion, we use a quadrupedal ant robot, where the state space consists of all joint angles, along with the Cartesian position of the center of mass (CoM). The manipulation task uses a simulated Sawyer arm to push an object, and the state consists of the end-effector and object positions.

These environments present interesting representation learning challenges. In 2D navigation, the walls impose structure similar to those in Figure 1: geometrically proximate locations on either side of a wall are far apart in terms of reachability, and we expect a good representation to pick up on this. The locomotion environments present an additional challenge: an effective representation must account for the fact that the internal joints of each robot (legs or wheel orientation) are less salient for long-horizon tasks than CoM. The original state representation does not reflect this structure: joint angles expressed in radians carry as much weight as CoM positions in meters. In the object manipulation task, the representation should be able to distinguish between pushing the block and simply moving the arm in free space.



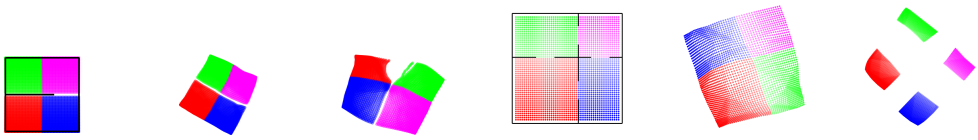
(a) 2D Wall (b) 2D Rooms (c) Wheeled (d) Wheeled Rooms (e) Ant (f) Pushing

Figure 4: The tasks in our evaluation. The 2D navigation tasks allow for easy visualization and analysis, while the more complex tasks allow us to investigate how well ARC and prior methods can discern the most functionally-relevant features of the state.

6.2 COMPARISONS WITH PRIOR WORK

We compare ARC to other general-purpose representation learning methods: variational autoencoders (Kingma & Welling, 2013) (VAE), VAEs trained for feature slowness (Jonschkowski & Brock, 2015) (slowness), features extracted from a predictive model (Oh et al., 2015) (predictive model), and a naïve baseline that uses the full state space as the representation (state). For each downstream task in Section 4, we also compare with methods specifically designed for those tasks. For reward shaping, we compare with VIME (Houthooft et al., 2016), an exploration method based on novelty bonuses. For hierarchical RL, we compare with option critic (Klissarov et al., 2017) and an on-policy adaptation of HIRO, which learns a higher-level policy that sets goals for the low-level policy using the full state space (Nachum et al., 2018).

6.3 ANALYSIS OF LEARNED ACTIONABLE REPRESENTATIONS



(a) Wall (b) VAE Wall (c) ARC Wall (d) 4 Room (e) 4 Room VAE (f) 4 Room ARC

Figure 5: Visualization of ARC for 2D navigation. The states in the environment are colored to help visualize their position in representation space. For the wall task, points on opposite sides of the wall are clearly separated in ARC space (c). For four rooms, we see that ARCs provide a clear decomposition into room clusters (f), while VAEs do not (e).

We analyze the structure of ARC space for the tasks described in Section 6.1, to identify which factors of state ARC chooses to emphasize, and how system dynamics affect the representation.

To examine the 2D navigation tasks, we visualize the original state and learned representations in Figure 5. In both tasks, ARC reflects the dynamics: points close by in Euclidean distance in the original state space are distant in representation space *when they are functionally distinct*. For instance, there is a clear separation in the latent space where the wall should be, and points on opposite sides of the wall are much further apart in ARC space (Figure 5) than in the original environment and in the VAE representation. In the room navigation task, the passages between rooms are clear bottlenecks, and the ARC representation separates the rooms in representation space according to these bottlenecks.

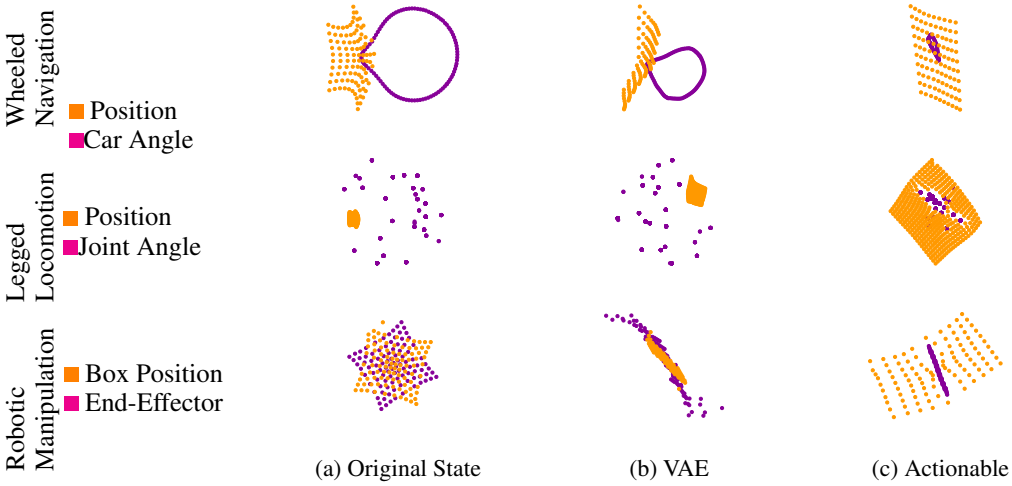


Figure 6: Perturbation analysis: As described in Section 6.3, we analyze the effect of perturbing different elements of state, on the learned representation. If the representation is effective, it will vary significantly with perturbations to elements of state that are functionally relevant (shown in orange), and not vary as much for secondary elements (shown in purple). Refer to Section 6.3 for a detailed explanation. ARC varies significantly while emphasizing functionally relevant elements (spread orange region) - position of robot CoM or object position, and suppressing secondary elements (small blue region). VAE and naive state representations are unable to capture this, and do not emphasize the functionally relevant factors over the secondary ones (small orange region, spread purple region)

The representations learned in more complex domains, such as wheeled or legged locomotion and block manipulation, also show meaningful patterns. We aim to understand which elements of state are being emphasized by the representation, and which are being suppressed. To do so, we analyze how distances in the latent space change as we perturb different elements of state. (Fig 6). We determine two factors in a state: one which we consider important for decision making (in orange), and one which is secondary (in purple). We expect a good representation to have a larger variation in distance as we perturb the important factor than when we perturb the secondary factor. For instance, in the legged locomotion environment, the CoM is the important factor and the joint angles are secondary. As we vary the CoM, the representation should vary significantly, while the effect should be muted as we vary the joints. For the wheeled environment, position of the car should cause large variations while the orientation should be secondary. For the object pushing, we'd expect block position to be the important factor, while end-effector position is secondary. Since distances in high-dimensional representation space are hard to visualize, we project [ARC, VAE, State] representations of perturbed states into 2 dimensions (Fig 6) using multi-dimensional scaling (MDS) (Borg & Groenen, 2005). MDS is a technique that projects points from a high dimensional space into 2-D space such that euclidean distances are preserved. From (Fig 6), we see that ARC captures the factors of interest; as the important factor is perturbed the representation changes significantly (spread out orange points), while when the secondary factor is perturbed the representation changes minimally (close together blue points). This implicates that for Ant, ARC captures CoM while sup-

pressing joint angles; for wheeled, ARC captures position while suppressing orientation; for block pushing, ARC captures block position, suppressing arm movement. Both VAE representations and original state space are unable to capture this as seen from Fig 6 (a), (b), where the variation in orange points is not significantly emphasized over blue points.

6.4 Leveraging Actionable Representations for Reward Shaping

As described in Section 4, ARCs can be used for reward shaping to solve tasks that present a large exploration challenge. We evaluate this on two challenging exploration tasks for wheeled locomotion and legged locomotion. We acquire an ARC from a goal-conditioned policy on a set of states $\mathcal{S}$ where the root position is within a 2m square around the origin. The learned representation is then used to guide learning via reward shaping for tasks on a much larger set of states $\mathcal{S}'$, where the root positions are within a square of 8m. The goal-conditioned policy does not generalize to the larger region, and exploration with standard sparse rewards is very challenging in this space. These tasks involve reaching goals in $\mathcal{S}'$, but with only a *sparse* goal completion reward.

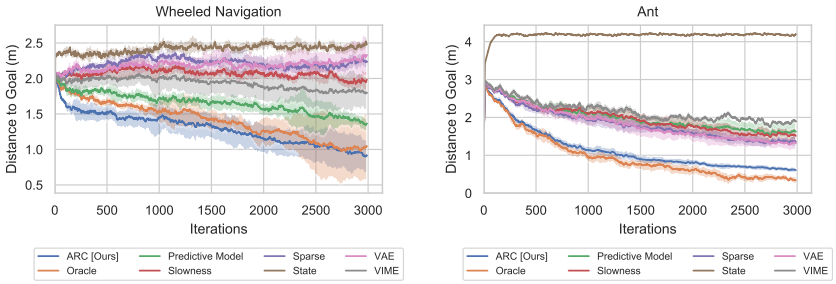


Figure 7: Learning new tasks with reward shaping in representation space. ARC representations are more effective than other methods, and match the performance of a hand-specified oracle. ARC representations generalize beyond the region they are trained on.

The policies use Euclidean distances in the ARC space, or another representation space, as reward shaping to guide learning. As seen in Fig 7, using ARC for reward shaping makes it possible to learn policies for goals to which the original goal-conditioned policy does not generalize effectively. This is likely because the representation simply needs to extract salient elements of the state, and does not need to actually predict optimal actions. ARC substantially outperforms other methods when used for reward shaping, matching performance of the oracle reward shaping distance, which directly uses global position of the root. The knowledge that the root position is important for the task is not available to the non-oracle methods. ARC also outperforms VIME, which is a dedicated exploration algorithm.

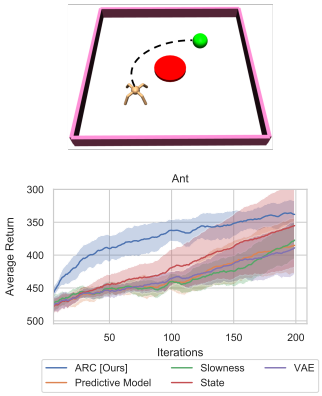


Figure 8: Policy learning using actionable representations as features. **Top**: Reach-while-avoiding task with the legged robot. **Bottom**: Comparison of learning with ARCs compared with other representation learning methods.

6.5 Leveraging Actionable Representations as Features for Learning Policies

We empirically evaluate the effectiveness of using the ARC representation as a feature space for learning policies for new tasks. In particular, we consider a task in which the quadruped robot must walk to a target while avoiding a dangerous region on the path there. Note that this task cannot be performed by the original goal-conditioned policy, since it cannot be represented by a single goal state.

We see from Fig 8 that using ARC as a feature representation allows the policy to quickly learn how to perform this task, which is significantly harder than goal reaching tasks. ARC outperforms the alternative representation learning methods, because ARC emphasizes those elements of the state that are important for control. In the comparison, we see that other representation learning methods actually do not improve over the original state representation.

6.6 Building Hierarchy from Actionable Representations

We learn several temporally extended tasks using ARCs, as described in Section 4. These tasks require reasoning about a sequence of states to reach to maximize a reward function over long horizons. Specifically, we consider tasks in 2D multi-room navigation, wheeled multi-room locomotion and legged locomotion through waypoints. For the multi-room tasks, the goal is to reach a sampled sequence of rooms in a particular order using either a wheeled robot or a 2D agent. We also consider a waypoint navigation task with the ant. The goal is to reach a sequence of waypoints in order.

We consider two different strategies for hierarchical reasoning with ARCs: commanding the goal-conditioned policy with a meta-policy either directly in ARC space, or specifying a target cluster in the representation space. As shown in Fig 9, using a hierarchical meta-policy with ARCs as described in Section 4 performs significantly better than similar techniques applied to alternative representation learning methods (using one of the two methods). These representations are unable to capture abstraction and environment dynamics, and give us slower learning than using actionable representations or their clusterings.

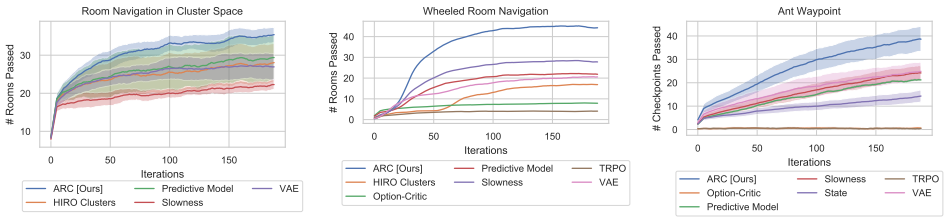


Figure 9: Comparison on hierarchical tasks with actionable representations vs prior representation learning methods. We see that on these tasks hierarchical RL with ARCs does significantly better than other representation methods, option-critic, and commanding goals in state space

As shown in Fig 5, for multi-rooms, ARC clusters very clearly capture different rooms, simplifying hierarchical reasoning. Using the ARCs for hierarchical RL works better than having the meta-policy directly command goals in the state space (Nachum et al., 2018), because state space has redundancies which makes search challenging.

7 Discussion

In this work, we introduce actionable representations, which capture representations of state which are important for decision making. We build on the framework of goal-conditioned RL, and use a goal-conditioned policy to extract state representations that emphasize features of state that are functionally relevant. The learned state representations are implicitly aware of the environment dynamics, and capture meaningful distances in representation space. Actionable representations are useful for tasks such as learning policies, hierarchical RL and reward shaping.

References

Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *CoRR*, abs/1606.07419, 2016.

John-Alexander M. Assael, Niklas Wahlström, Thomas B. Schön, and Marc Peter Deisenroth. Data-efficient learning of feedback policies from image pixels using deep dynamical models. *CoRR*, abs/1510.02173, 2015.

André Barreto, Rémi Munos, Tom Schaul, and David Silver. Successor features for transfer in reinforcement learning. *CoRR*, abs/1606.05312, 2016.

Ishmael Belghazi, Sai Rajeswar, Aristide Baratin, R. Devon Hjelm, and Aaron C. Courville. MINE: mutual information neural estimation. *CoRR*, abs/1801.04062, 2018.

Emmanuel Bengio, Valentin Thomas, Joelle Pineau, Doina Precup, and Yoshua Bengio. Independently controllable features. *CoRR*, abs/1703.07718, 2017.

I. Borg and P.J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*, pp. 539–546, 2005. doi: 10.1109/CVPR.2005.202.

William Curran, Tim Brys, Matthew E. Taylor, and William D. Smart. Using PCA to efficiently represent state spaces. *CoRR*, abs/1505.00322, 2015.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martín Arjovsky, Olivier Mastropietro, and Aaron C. Courville. Adversarially learned inference. *CoRR*, abs/1606.00704, 2016.

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Learning visual feature spaces for robotic manipulation with deep spatial autoencoders. *CoRR*, abs/1509.06113, 2015.

Ali Ghadirzadeh, Atsuto Maki, Danica Kragic, and Mårten Björkman. Deep predictive policy training using reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, pp. 2351–2358, 2017. doi: 10.1109/IROS.2017.8206046.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Ross Goroshin, Michaël Mathieu, and Yann LeCun. Learning to linearize under uncertainty. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 1234–1242, 2015.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *CoRR*, abs/1702.08165, 2017.

Irina Higgins, Arka Pal, Andrei A. Rusu, Loïc Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. DARLA: improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 1480–1490, 2017.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Curiosity-driven exploration in deep reinforcement learning via bayesian neural networks. *CoRR*, abs/1605.09674, 2016.

Rico Jonschkowski and Oliver Brock. Learning state representations with robotic priors. *Auton. Robots*, 39(3):407–428, 2015. doi: 10.1007/s10514-015-9459-7.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

Martin Klissarov, Pierre-Luc Bacon, Jean Harb, and Doina Precup. Learnings options end-to-end for continuous action tasks. *CoRR*, abs/1712.00004, 2017. URL `http://arxiv.org/abs/1712.00004`.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pp. 1106–1114, 2012.

Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart J. Russell, and Pieter Abbeel. Learning plannable representations with causal infogan. *CoRR*, abs/1807.09341, 2018.

Timothée Lesort, Natalia Díaz Rodríguez, Jean-François Goudou, and David Filliat. State representation learning for control: An overview. *CoRR*, abs/1802.04181, 2018.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *CoRR*, abs/1504.00702, 2015.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *CoRR*, abs/1805.08296, 2018.

Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *CoRR*, abs/1807.04742, 2018.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L. Lewis, and Satinder P. Singh. Action-conditional video prediction using deep networks in atari games. *CoRR*, abs/1507.08750, 2015.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops, Honolulu, HI, USA, July 21-26, 2017*, pp. 488–489, 2017. doi: 10.1109/CVPRW.2017.70.

Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund, and Tapani Raiko. Semi-supervised learning with ladder network. *CoRR*, abs/1507.02672, 2015.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 1889–1897, 2015.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pp. 1134–1141, 2018. doi: 10.1109/ICRA.2018.8462891.

Emanuel Todorov. Linearly-solvable markov decision problems. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pp. 1369–1376, 2006.

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

Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin A. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 2746–2754, 2015.

Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009. doi: 10.1145/1577069.1577078.

Amy Zhang, Harsh Satija, and Joelle Pineau. Decoupling dynamics and reward for transfer learning. *CoRR*, abs/1804.10689, 2018a.

Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew J. Johnson, and Sergey Levine. SOLAR: deep structured latent representations for model-based reinforcement learning. *CoRR*, abs/1808.09105, 2018b.