Abstract
Learning data representations that are useful for various downstream tasks is a cornerstone of artificial intelligence. While existing methods are typically evaluated on downstream tasks such as classification or generative image quality, we propose to assess representations through their usefulness in downstream control tasks, such as reaching or pushing objects. By training over 10,000 reinforcement learning policies, we extensively evaluate to what extent different representation properties affect out-of-distribution (OOD) generalization. Finally, we demonstrate zero-shot transfer of these policies from simulation to the real world, without any domain randomization or fine-tuning. This paper aims to establish the first systematic characterization of the usefulness of learned representations for real-world OOD downstream tasks.

1 Introduction
Robust out-of-distribution (OOD) generalization is one of the key open challenges in machine learning. This is particularly relevant for the deployment of ML models to the real world, where we need systems that generalize well beyond the i.i.d. (independent and identically distributed) data setting (Schölkopf et al., 2021; Djolonga et al., 2020; Koh et al., 2020; Barbu et al., 2019; Azulay and Weiss, 2019; Roy et al., 2018; Gulrajani and Lopez-Paz, 2020; Hendrycks and Dietterich, 2019; Michaelis et al., 2019). One instance of such models are agents that learn by interacting with a training environment but cannot generalize and transfer their learned skill to other environments with different statistics (Zhang et al., 2018; Cobbe et al., 2019; Ahmed et al., 2021).

Consider the example of a robot with the task of moving a cube to a target position: Such an agent can easily fail as soon as some aspects of the environment differ with respect to the training setup, e.g. the shape, color, and other object properties, or when transferring from simulation to real world. In particular, some of the main issues in deep reinforcement learning are data inefficiency, brittleness with respect to changes in the input data distribution, and poor interpretability (Garnelo et al., 2016; Lake et al., 2017; Kaiser et al., 2019; Li, 2018; Zambaldi et al., 2019; Lyu et al., 2019; Zhang et al., 2019; Heuillet et al., 2021). Humans seem to not suffer from these pitfalls when transferring learned skills beyond a narrow training domain. In fact, one of the fundamental cognitive capabilities in humans is to represent visual sensory data in a useful and concise manner (Marr, 1982; Gordon and Irwin, 1996; Lake et al., 2017; Anand et al., 2019; Spelke, 1990). Therefore, a particularly promising path is to base decisions and predictions on such structured and meaningful lower-dimensional representations of our world (Bengio et al., 2013). The learned representation should facilitate efficient downstream learning (Eslami et al., 2018; Anand et al., 2019) and exhibit better generalization (Zhang et al., 2020; Srinivas et al., 2020).

While previous work shows that good internal representations of raw observations are important for domain adaptation (Littman et al., 2001; Pan and Yang, 2009; Finn et al., 2016a; Barreto et al., 2017), so far representations are typically evaluated on downstream tasks such as classification or generative image quality which often serve as proxies for intended use cases. To move closer to realistic settings, we present a large-scale study (with 11,520 trained policies) investigating the relevance of learning representations for real-world reinforcement learning and OOD generalization. This study is based on the practically relevant setting of robotics and empirically analyzes key principles for representations and downstream policies in simulation and real world. See Fig. 1 for an overview of the setup.

We summarize our contributions as follows:

- We conduct a large-scale study training 11,520 policies and empirically investigate the role of pre-trained
2 Background

In this section, we provide relevant background on the methods for representation learning, reinforcement learning, and evaluation of out-of-distribution generalization.

Variational autoencoders. VAEs (Kingma and Welling, 2014; Rezende et al., 2014) are a framework for optimizing a latent variable model \( p_\theta(x) = \int \phi(z) p(z|x) dz \) with parameters \( \theta \), typically with a fixed prior \( p(z) = \mathcal{N}(z; \mathbf{0}, \mathbf{I}) \), using amortized stochastic variational inference. A variational distribution \( q_\phi(z|x) \) with parameters \( \phi \) approximates the intractable posterior \( p_\theta(z|x) \). The approximate posterior and generative model, typically called encoder and decoder and parameterized by neural networks, are jointly optimized by maximizing the ELBO (Evidence Lower BOund) which is a lower bound to the log likelihood:

\[
\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z))
\]

In \( \beta \)-VAEs, the KL term is modulated by a factor \( \beta \) to enforce a more structured latent space (Higgins et al., 2017a; Burgess et al., 2018). While VAEs are typically trained without supervision, in this work we will also employ a form of weak supervision proposed by Locatello et al. (2020) to learn disentangled representations.

Deep reinforcement learning. The problem setting in Reinforcement Learning (RL) is modeled via a Partially Observable Markov Decision Process (POMDP) defined by the tuple \((S, A, T, R, \Omega, O, \gamma, \rho_0, H)\) with states \( s \in S \), actions \( a \in A \) and observations \( o \in \Omega \) determined by the state and action of the environment \( O(o|s, a) \). \( T(s_{t+1}|s_t, a_t) \) is the transition probability distribution function, \( R(s_t, a_t) \) is the reward function, \( \gamma \) is the discount factor, \( \rho_0(s) \) is the initial state distribution at the beginning of each episode, and \( H \) is the time horizon per episode. The objective in RL is to learn a policy \( \pi : S \times A \rightarrow [0, 1] \), typically parameterized by a neural network, that maximizes the total discounted expected reward \( J(\pi) = \mathbb{E} \left[ \sum_{t=0}^{H} \gamma^t R(s_t, a_t) \right] \). There is a broad range of proposed model-free learning algorithms to find \( \pi^* \) by policy gradient optimization or learning value functions while trading off exploration and exploitation (Haarnoja et al., 2018; Schulman et al., 2017; Sutton et al., 1999; Schulman et al., 2015b; Silver et al., 2014; Fujimoto et al., 2018). Here, we optimize the objective above with Soft Actor Critic (SAC), a widely used off-policy method in control tasks due to its sample efficiency (Haarnoja et al., 2018). SAC aims to improve sample-inefficiency and convergence in RL by simultaneously maximizing the expected reward and the entropy \( H(\pi(\cdot|s_t)) \).

Robotic setup and related dataset. Recently, Dittadi et al. (2021) introduced a dataset of over 1M simulated and real-world images derived from the Trifinger robot platform introduced by Wüthrich et al. (2020) we will base our study on. The scene comprises a robot finger with three joints that can be controlled to manipulate a cube in a bowl-shaped stage. See Fig. 1 (step 1) for an example. The data was generated from 7 different factors of variation (FoV): angles of the upper, middle, and lower joints, and position \((x, y)\), orientation, and color of the cube. This dataset corresponds to a robotic setup, so that learned representations can be used for control and reinforcement learning in simulation and in the real world. However, the focus of that work was to scale VAE-based learning approaches to this complex dataset and conduct a large-scale empirical study on generalization to various transfer scenarios, with a particular emphasis on disentanglement. For this reason, the role of representations in robotic control downstream tasks was not investigated.

Measuring out-of-distribution generalization. We closely follow the framework for measuring out-of-distribution (OOD) generalization proposed by Dittadi et al. (2021). First, representations are learned on a training set \( D_1 \). Then, we investigate OOD generalization by training downstream models on a subset \( D_2 \subset D \) to predict ground truth factor values from the learned representations. These models are then tested on a set \( D_3 \) that differs distributionally from the training set \( D_1 \), e.g. containing images corresponding to held-out values of a chosen factor of variation (FoV).

Dittadi et al. (2021) consider two flavors of OOD generalization depending on the choice of \( D_2 \): The case when \( D_2 \subset D \), i.e. the OOD test set is a subset of the dataset for representation learning, is denoted by OOD1, while in OOD2 \( D_1 \) and \( D_2 \) are disjoint and distributionally different.

For example, consider the case in which distributional shifts are based on one FoV: the color of the cube in our robotic setup. Then, we could define these datasets such that images in \( D_1 \) always contain a red or blue object, and those in \( D_2 \subset D_1 \) always contain a red object. In the OOD1 scenario, images in \( D_2 \) would always contain a blue object, whereas in the OOD2 case they would always contain an object that is neither red nor blue. Dittadi et al. (2021) consider as regression models Gradient Boosted Trees (GBT) and MLPs.
Representation Learning for Out-of-distribution Generalization in Reinforcement Learning

Figure 1. Overview of our experimental setup for investigating out-of-distribution generalization in downstream tasks. (1) We train 240 $\beta$-VAEs on the robotic dataset from Dittadi et al. (2021). (2) We then train downstream policies to solve ReachCube or Pushing, using multiple random RL seeds per VAE. The input to a policy consists of the output of a pre-trained encoder and additional task-related observable variables. Crucially, the policy is only trained on a subset of the cube colors from the pre-training dataset. (3) Finally, we evaluate these policies on their respective tasks in four different scenarios: (a) in-distribution, i.e. with cube colors used in policy training; (b) OOD1, i.e. with cube colours previously seen by the encoder but OOD for the policy; (c) OOD2-sim, having cube colours also OOD to the encoder; (d) sim2real zero-shot on the real world setup.

with 2 hidden layers. Generalization in $D_2$ is measured by the (normalized) mean absolute prediction error across all FoVs except for the one that is OOD. In this work, we will use the negative error for interpretability, and decompose this metric on a per-factor level as well and refer to this generalization score as GS-OOD1, GS-OOD2-sim and GS-OOD2-real accordingly.

In contrast to Dittadi et al. (2021), who focus on introducing the dataset and evaluating generalization of simple factor prediction tasks, we leverage the broader potential of this robotic setup, which is to evaluate the usefulness of representations for more practically relevant robotic downstream tasks. Additionally, we cover more OOD2 scenarios both in simulation and in the real world, and investigate the relationship between a wide array of metrics and OOD generalization in RL tasks.

3 Related work

Lower-dimensional representations that can be flexibly used for a multitude of downstream tasks are considered an important component of any robust and generalizing machine learning system (Bengio et al., 2013; Schölkopf et al., 2021). In machine learning, these representations are typically learned from labels or rewards, which is often sample-inefficient. More sample-efficient alternatives that leverage the large amount of unstructured information in raw data include unsupervised (Kingma and Welling, 2014; Rezende et al., 2014; Dinh et al., 2016; Dumoulin et al., 2016) and self-supervised learning (Pathak et al., 2016; Doersch and Zisserman, 2017; Kolesnikov et al., 2019; Chen et al., 2020). In particular, disentangled representation learning aims at inferring the (causal) factors of the generative model of the data by enforcing sufficient structure on the latent space (Higgins et al., 2017a; Kim and Mnih, 2018; Burgess et al., 2018; Kumar et al., 2018; Chen et al., 2018; Locatello et al., 2019b, 2020).

Evaluating representations. In generative modeling, representations are typically evaluated by log likelihood, ELBO, or perceptual metrics such as FID (Heusel et al., 2017), IS (Salimans et al., 2016), or precision/recall (Sajjadi et al., 2018; Kynkäänniemi et al., 2019). Compression capability can also be evaluated e.g. in the context of bits-back coding, where it is formally related to the ELBO (Honkela and Valpola, 2004; Townsend et al., 2019; Kingma et al., 2019; Ruan et al., 2021). In general, representation quality has also been measured in terms of disentanglement (Higgins et al., 2017a; Kim and Mnih, 2018; Chen et al., 2018; Ridgeway and Mozer, 2018; Kumar et al., 2018; Eastwood and Williams, 2018), robustness (Suter et al., 2019), or the complexity of learning downstream predictors (Whitney et al., 2020). The evaluation framework in this paper is related to recent work that focuses on evaluating generalization in various practically relevant out-of-distributions settings (Gondal et al., 2019; Träuble et al., 2020; Dittadi et al., 2021). To the best of our knowledge, there is no rigorous and systematic study on the role of representations on downstream performance in robotic downstream tasks.
Learning representations for control. Learning low-dimensional representations that are capturing environments’ variations for RL agents in control scenarios is also often being described as state representation learning (Lesort et al., 2018). Methods therein are typically based on autoencoders (Watter et al., 2015; Ha and Schmidhuber, 2018; Higgins et al., 2017b; Van Hoof et al., 2016), video prediction (Oh et al., 2015; Finn et al., 2016b) or contrastive learning (Akkaya et al., 2019; James et al., 2019). Yan et al. tackled by using large-scale domain randomization in simulations and is sometimes unfeasible if the reward structure is typically not very sample-efficient, a well-known problem of deep RL (Lake et al., 2017; Kaiser et al., 2019). It is thus becoming increasingly popular to leverage pre-trained representations and effectively decouple representation learning and policy learning in pixel-based environments (Esliam et al., 2018; Cuccu et al., 2018). For example, Stoeko et al. (2020) propose that decoupling representation learning from RL is more efficient than learning reward structure from pixels on their contrastive method. Similarly, CURL (Srinivas et al., 2020) investigates contrastive representation learning simultaneously with an off-policy RL algorithm, and match the sample-efficiency of policy learning from state-based features. Previous works highlight that for domain adaptation it is important to have good internal representations of raw observations (Littman et al., 2001; Pan and Yang, 2009; Finn et al., 2016a; Barreto et al., 2017). It is argued that these representations should be learned from the source domain only, because it might be difficult or expensive to obtain training data from the target domain (Finn et al., 2017; Rusu et al., 2017). Importantly, by simply training deep RL from scratch, the policies will often overfit to the source distribution (Rusu et al., 2017; Lake et al., 2017).

Closing the (sim2real) generalization gap in real-world RL. A key unsolved challenge in RL is that agents are very brittle to distribution shifts in their environment, even if the underlying structure is largely unchanged (Cobbe et al., 2019; Ahmed et al., 2021). DARLA (Higgins et al., 2017b) focuses on domain adaptation and zero-shot transfer for RL in DeepMind Lab and MuJoCo environments, and claim disentangled representations improve robustness. To obtain better transfer capabilities, Asadi et al. (2020) argue for discretizing the state space in continuous control domains by clustering together states where the optimal policy is similar. Transfer becomes especially challenging from the simulation to the real-world, a phenomenon often referred to as the sim2real gap. This is particularly crucial in RL, as real-world training is expensive, requires sample-efficient methods and is sometimes unfeasible if the reward structure requires accurate ground truth labels (Dulac-Arnold et al., 2019; Kormushev et al., 2013). Typically this issue is tackled by using large-scale domain randomization in simulation (Akkaya et al., 2019; James et al., 2019). Yan et al. (2020) propose using segmentation as a domain-invariant state representation.

4 Study design

Fig. 1 provides an overview of our setup. We study the role of visual representation learning for reinforcement learning in two control tasks:

1. ReachCube: Reach a fixed cube at random positions with a time limit of 2 seconds.

2. Pushing: A more challenging task, where the goal is to push the cube to a random goal pose in the arena within a maximum time of 4 seconds.

We derive both tasks from the CausalWorld benchmark environments (Ahmed et al., 2021). The scene comprises a robot finger with three joints that can be controlled to manipulate a cube in a bowl-shaped stage. The robot is derived from the TriFinger design from Wüthrich et al. (2020). The input variables at time t are the camera observation o_t and a vector of observable variables x_t, which contains the joint angles and velocities in both tasks, as well as the target position for the cube in Pushing. We then feed the camera observation o_t into a given encoder e that was pre-trained on the dataset in Dittadi et al. (2021). The resulting z_t = e(o_t) is concatenated with x_t, yielding a state vector s_t = [x_t, z_t]. For each task we then use SAC to train the policy with s_t as input, implemented with (Hill et al., 2018). The policy, value and Q networks are all implemented as MLPs with 2 hidden layers of size 256. Note that the representation function is fixed when training the policies, i.e. the encoder is not fine-tuned, as our goal is to investigate the link between representation properties and downstream RL performance.

We perform a large-scale empirical study on the setup introduced above by training 11,520 policies across both tasks. The hyperparameter sweep is defined as follows:

- We train 240 β-VAEs, with a subset of the hyperparameter configurations and neural architecture from Dittadi et al. (2021). Specifically, we consider β ∈ {1, 2, 4}, β annealing in {0, 50000} steps, unsupervised and weakly supervised training (Higgins et al., 2017a; Locatello et al., 2020), with and without input noise, and 10 random seeds per configuration. The latent space size is fixed to 10.

- For the ReachCube task, we train 20 downstream policies (with different random seeds) for each of the 240 VAEs. This results in 4,800 policies, which we train with SAC for 400k steps (approximately 2,400 episodes).

- Since the Pushing task takes substantially longer to train, we limit the number of policies to be trained on this task. First, we choose a subset of 96 VAEs corresponding to using only 4 random seeds for the
We evaluate the generalization of policies in three different volumetric overlap of the cube with the goal cube, and the end effector is touching the center of a face of the cube. In Pushing, the task is to select good representations for downstream RL. On the real robot without any fine-tuning in Section 5.3 and how different components of the pre-trained representations and regularization affects in-distribution performance and training reward. Next, we extensively account of predictive factors for out-of-distribution generalization for RL from pre-trained representations is given in Section 5.2 focusing on the simulated environment.

Finally, we extensively evaluate zero-shot sim2real transfer to the real robot without any fine-tuning in Section 5.3 and also discuss predictors for OOD generalization when going to the real world.

5 Results

We split in Section 5.1 by presenting the training results of our large-scale sweep, and how different components of the representations, our policies using the same MLP architecture could in principle internally compute the FoVs relevant for the task. Interestingly, the correlation with the overall MLP accuracy mostly stems from the prediction accuracy of the cube pose FoVs, which are in fact the ones that are not included in the ground-truth robot state $x_t$. These results suggest that these metrics can be used to select good representations for downstream RL. On the more challenging task of Pushing, the correlations are milder but most of them still statistically significant. In general, all correlations discussed in this paper are statistically significant (colored coefficients in figures whenever $p<0.05$).

Summary. Both tasks can be consistently solved from pixel data using pre-trained representations. In particular, all policies on ReachCube attain almost perfect scores. Pushing is a much more complex tasks, involving learning the non-linear rigid-body interactions. Unsurprisingly, this task requires significantly more training, and the variance of performance across policies is larger. Nonetheless, almost all policies learn to solve the task satisfactorily. To investigate the effect of representations on the training reward, we now compute its Spearman rank correlations with various supervised and unsupervised metrics of the representations (Fig. 2 bottom). On ReachCube, the final reward correlates with ELBO and reconstruction loss. A simple supervised metric to evaluate a representation is how well a small downstream model can predict the ground-truth FoV. Following Dittadi et al. (2021), we use the MLP10000 and GBT10000 metrics, where MLPs and GBTs are trained for predicting the FoVs from 10,000 samples (we will simply call these metrics MLP and GBT in the following). Training reward correlates with these metrics as well, especially with the MLP accuracy. This is not entirely surprising: if an MLP can predict the FoVs from the representations, our policies using the same MLP architecture could in principle internally compute the FoVs relevant for the task. Interestingly, the correlation with the overall MLP accuracy mostly stems from the prediction accuracy of the cube pose FoVs, which are in fact the ones that are not included in the ground-truth robot state $x_t$. These results suggest that these metrics can be used to select good representations for downstream RL. On the more challenging task of Pushing, the correlations are milder but most of them still statistically significant. In general, all correlations discussed in this paper are statistically significant (colored coefficients in figures whenever $p<0.05$).

Summary. Both tasks can be consistently solved from pixel data using pre-trained representations. Unsupervised (ELBO, reconstruction loss) and supervised (ground-truth factor prediction) metrics of the representations are correlated with reward in the training environment.

5.2 Out-of-distribution generalization in simulation

From train time performance to OOD generalization. Fig. 3 shows that in-distribution reward correlates with OOD1 performance on both tasks, especially with L1 regu-
Figure 2. Top: Average training success, aggregated over all policies from the sweep (median, quartiles, 5th/95th percentiles). Bottom: Rank correlations between representation metrics and training reward, in the case without policy regularization.

Unsupervised metrics and informativeness. In Fig. 4 (left) we assess the relation of OOD reward with unsupervised metrics (ELBO and reconstruction loss) and downstream performance on factor prediction (using MLP and GBT). Both ELBO and reconstruction loss exhibit a correlation with OOD1 reward, but not with OOD2 reward. These unsupervised metrics can thus be useful for selecting representations that will lead to more robust downstream RL tasks, as long as the representation function is in-distribution. While the GBT score is not correlated with reward under distribution shift, we observe a significant correlation between OOD1 reward and the MLP score, which measures downstream factor prediction accuracy of an MLP with the same architecture as the one parameterizing the policies. As in Section 5.1, we further investigate the source of this correlation, and find it in the pose parameters of the cube. Correlations in the OOD2 setting are much weaker, thus we conclude that these metrics do not appear helpful for model selection in this case. Our results on Pushing confirm these conclusions although correlations are generally weaker, presumably due to the more complicated nature of this task. (see Appendix B.2).

Disentangled representations. Almost perfect disentanglement has been shown to be helpful for downstream performance and OOD1 generalization even with MLP downstream tasks. However, in ReachCube without regularization, we only observe a weak correlation with some disentanglement metrics (Fig. 5). In agreement with Dittadi et al. (2021), disentanglement does not seem to correlate with OOD2 generalization. Dittadi et al. (2021) observed that disentanglement correlates with the informativeness of a representation. To understand if these weak correlations originate from this common confounder, we investigate whether disentanglement is still correlated with a higher OOD1 reward if we compare representations with similar MLP FoV prediction accuracy. Given two representations with the same MLP accuracy, does the more disentangled one exhibit better OOD1 generalization? To measure this we predict success from the MLP accuracy using kNN (k=5).
and compute the residual reward by subtracting the amount of reward explained by the MLP accuracy. In Fig. 5, we see that this resolves the remaining correlations with disentanglement. Thus, for the downstream tasks considered here, disentanglement does not seem to be useful for downstream OOD generalization. We present similar results on Pushing in Appendix B.2.

Policy regularization and observation noise. It might seem unsurprising that disentanglement is not useful for generalization in RL, as MLP policies do not have any explicit inductive bias to exploit it. Thus, we attempt to introduce such inductive bias by repeating all experiments with L1 regularization on the first layer of the policy. As discussed in Appendix B.2, although regularization has a positive effect on OOD1 and OOD2 generalization in general (Fig. 5 right), we see no link with disentanglement. In agreement with (Dittadi et al., 2021), we also find that observation noise when training representations is beneficial for OOD2 generalization (see Appendix B.2).

Strong OOD shifts: evaluating on a novel shape. On the ReachCube task, we also tested generalization w.r.t. a novel object shape by replacing the cube with an unmovable sphere. This corresponds to a strong OOD2-type shift, since shape was never varied when training the representations. We then evaluated the trained policies as before, with the same color splits. Surprisingly, the policies appear to handle the novel shape. In fact, when the sphere has the same colors that the cube had during policy training, all policies get closer than 5cm to the sphere on average, with a mean success metric of about 95%. On sphere colors from the OOD1 split, more than 98.5% move the finger closer than this threshold, and on the strongest distribution shift (OOD2-sim colors and cube replaced by sphere) almost 70% surpass that threshold with an average success metric above 80%.

Summary. Reward from the training environment is significantly correlated with OOD generalization reward, as long as the encoder remains in its training distribution (OOD1 generalization). The OOD1 reward is significantly correlated with ELBO, reconstruction loss, and the MLP accuracy. This however does not hold for the OOD2-sim reward, hence these metrics cannot be used to predict OOD2 generalization in our experiments. The generalization metrics from (Dittadi et al., 2021), which measure robustness to distribution shifts, are significantly correlated with RL performance under similar distribution shifts. These metrics are thus useful for selecting representations that will yield robust downstream policies. Disentanglement does not seem to be beneficial for generalization in this setting, while input noise when training representations is beneficial for OOD2 generalization.

5.3 Deploying policies to the real world

We now evaluate a large subset of the trained models sim2real on the equivalent real robot (Wüthrich et al., 2020) without any additional fine-tuning. We are interested in quantifying if our models are able to generalize zero-shot on the real robot and attempt to uncover relevant metrics for predicting real world performance.

Reaching. We chose 960 policies trained in simulation, based on 96 representations and 10 random seeds, and evaluate them on two (randomly chosen, but significantly far apart) goal positions using a red cube. Note that although a
Figure 6. Zero-shot sim2real on ReachBlock. **Top:** Statistically significant rank-correlations on the real platform with a red cube. **Bottom left:** Training encoders with additive noise improves sim2real generalization. **Bottom right:** Histogram of fractional success in the more challenging OOD2-real-{green,blue} scenario from 50 policies across 4 goal positions.

red cube was in the training distribution in simulation, we consider this to be OOD2 because real world images already represent a strong distribution shift for the encoder (Dittadi et al., 2021; Djolonga et al., 2020). Although sim2real in robotics is considered to be very challenging without domain randomization or fine-tuning (Tobin et al., 2017; Finn et al., 2017; Rusu et al., 2017), many of our trained policies obtain a high fractional success score without resorting to these methods. In addition, in Fig. 6 (top) we observe consistent correlations between zero-shot real-world performance and some relevant quantities discussed previously. First, there is a positive correlation with the OOD2-sim reward: Policies that generalize well on unseen cube colors in simulation seem to generalize well to the real world, too. Second, representations with high GS-OOD2-sim and (especially) GS-OOD2-real scores are promising candidates for good sim2real transfer. Third, if no FoV labels are available, the weak statistically significant correlation with the reconstruction loss on the simulated images can be exploited for representation selection. Finally, as observed in (Dittadi et al., 2021) for significantly easier downstream tasks, input noise while learning representations is beneficial for sim2real generalization (see also Fig. 6, bottom left).

Based on these findings, we select 50 policies with high GS-OOD2-real, and evaluate them on the real world with a green and a blue cube, which is an even stronger OOD2 distribution shift than the one considered before. In Fig. 6 (bottom right), where performance metrics are averaged over 4 cube positions per policy, we observe that most policies can still reliably solve the task: approximately 80% of them position the finger less than 5 cm from the cube. Lastly, we mirrored the evaluations in simulation on an unseen green sphere object, and saw a surprisingly consistent finger movement to even such a new unseen object. We refer to Appendix B.3 in the supplementary material.

**Pushing.** We now test whether our real-world findings on ReachCube also hold for Pushing. We again selected a few policies with encoders being trained with added noise on the input and a high GS-OOD2-real score. We recorded episodes on diverse goal positions and cube colors to support our finding that it was also possible to obtain generalizing pushing policies on the real robot (Wüthrich et al., 2020) purely trained in simulation. In Fig. 7 we depict three representative episodes with successful task completions.

**Summary.** Policies trained solely within simulation can zero-shot solve the task on the real robot equivalent without any domain randomization or fine-tuning. We observe that OOD2 robustness of the underlying image encoder is a good predictor for real world performance as is the reconstruction loss of the VAE on simulated images and RL reward measured in a simulated OOD2 setting. For real-world application, we recommend using GS-OOD2-sim and GS-OOD2-real for model selection, and training the image encoder with additive noise.

6 Conclusion

Robust out-of-distribution (OOD) generalization is still one of the key open challenges in machine learning. With this work we presented a principled investigation of OOD generalization in the context of two practical downstream control tasks using RL from vision in simulation and the real world and how this is being driven by pre-trained representations. We worked out key predictors for various OOD generalization scenarios, whose statistical significance is supported by the over 10,000 control policies trained in this study. Ideally, our extensive investigation of representation learning for out-of-distribution generalization in reinforcement learning should encourage further work in this direction.
References


Representation Learning for Out-of-distribution Generalization in Reinforcement Learning


Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Pratul Dharwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John


A Implementation details

Task definitions and rewards. We derived both tasks, ReachCube and Pushing, from the CausalWorld environments introduced by Ahmed et al. (2021). We pre-train representations on the dataset introduced by Dittadi et al. (2021), and allow only one finger to move in our RL experiments. We propose the ReachCube environment as an intermediate simpler RL environment that involves a fixed cube that cannot be moved. We used reward structures similar to those in Ahmed et al. (2021):

- **ReachCube**: \( r_t = -750 \left[ d(g_t, e_t) - d(g_{t-1}, e_{t-1}) \right] \)
- **Pushing**: \( r_t = -750 \left[ d(o_t, e_t) - d(o_{t-1}, e_{t-1}) \right] - 250 \left[ d(o_t, g_t) - d(o_{t-1}, g_{t-1}) \right] + \rho_t \)

where \( t \) denotes the time step, \( \rho_t \in [0, 1] \) is the fractional overlap with the goal cube at time \( t \), \( e_t \in \mathbb{R}^3 \) is the end-effector position, \( o_t \in \mathbb{R}^3 \) the cube position, \( g_t \in \mathbb{R}^3 \) the goal position, and \( d(\cdot, \cdot) \) denotes the Euclidean distance. The cube in ReachCube is fixed, i.e. \( o_t = g_t \) for all \( t \).

Besides the accumulated reward along episodes, that is determined by the reward function, we also report two reward-independent normalized success definitions for better interpretability: In ReachCube, the success metric indicates progress from the initial end effector position to the optimal distance from the center of the cube. It is 0 if the final distance is greater than or equal to the initial distance, and 1 if the end effector is touching the center of a face of the cube. In Pushing, the success metric is defined as the volumetric overlap of the cube with the goal cube, and the task can be visually considered solved with a score around 80%. We observed that accumulated reward and success are highly correlated with each other, thus allowing to use one or the other for measuring performance.

During training, the goal position is randomly sampled at every episode. Similarly, the object color is being sampled from the 4 specified train colors from \( D_1 \) that are corresponding to the OOD1-B split from Dittadi et al. (2021).

For each policy evaluation (in-distribution and out-of-distribution variants), we average the accumulated reward and final success across 200 episodes with randomly sampled cube positions and the respective object color in both tasks.

SAC implementation. Our implementation of SAC builds upon the stable-baselines package (Hill et al., 2018). We used the same SAC hyperparameters used for pushing in Ahmed et al. (2021). When using L1 regularization, we add to the loss function the L1 norm of the first layers of all MLPs, scaled by a coefficient \( \alpha \). We gradually increase regularization by linearly annealing \( \alpha \) from 0 to \( 5 \cdot 10^{-7} \) in 200,000 time steps in ReachCube, and from 0 to \( 6 \cdot 10^{-8} \) in 3,000,000 time steps in Pushing.

B Additional results

B.1 Training environment

Fig. 2 in the main text shows correlations of unsupervised and supervised metrics with in-distribution reward for ReachCube and Pushing, only in the case without regularization. In Fig. 8 we also show these results in the case with regularization, as well as when adjusting for MLP informativeness.

B.2 Out-of-distribution generalization in simulation

In Section 5.2 we discussed rank-correlations of OOD rewards with unsupervised, informativeness and generalization scores on ReachCube without regularization. In Fig. 9 we also show these results for the case with regularization and on Pushing, as well as when adjusting for MLP informativeness. Without regularization, we observe on Pushing very similar correlations along all metrics as we observed on ReachCube, confirming our conclusions on this more complex task. When using regularization, rank correlations are very similar across both tasks. Interestingly, the correlation between GS-OOD2 scores and OOD2 generalization of the policy is even stronger when using the here studied type of regularization. In contrast to our observations without regularization, we find that the correlation between GS-OOD1 and OOD1 generalization of the policy disappears when adjusting for MLP informativeness.

Disentangled representations. As discussed in Section 5.2 for ReachCube without regularization, we observe in Fig. 9 a weak correlation between some disentanglement metrics and OOD1 reward, which however vanishes when adjusting for MLP informativeness. In agreement with Dittadi et al. (2021), we observe no significant correlation between disentanglement and OOD2 generalization, for both tasks, with and without regularization. From Fig. 10 we see that in some cases, especially without regularization, a very high DCI score seems to lead to better performance on average. However, this behavior is not significant (within error bars), as opposed to the results shown in simpler downstream tasks by Dittadi et al. (2021). Furthermore, this trend is likely due to representation informativeness, since the correlations with disentanglement disappear when adjusting for the MLP score, as discussed above.

Regularization. As seen in Fig. 10, regularization generally has a positive effect on OOD1 and OOD2 generalization, especially prominent in the OOD1 setting. On the other hand, it leads to lower training rewards both in ReachCube and in Pushing. In the latter, the performance drop is par-
### Figure 8. Rank correlations between metrics and in-distribution reward, with and without adjusting for informativeness.

<table>
<thead>
<tr>
<th></th>
<th>GT</th>
<th>MLP</th>
<th>MLP (1)</th>
<th>MLP (2)</th>
<th>MLP (3)</th>
<th>MLP (4)</th>
<th>MLP (5)</th>
<th>MLP (6)</th>
<th>MLP (7)</th>
<th>ReachCube, no regularization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reward in distr.</td>
<td>33</td>
<td>45</td>
<td>12</td>
<td>38</td>
<td>36</td>
<td>36</td>
<td>46</td>
<td>46</td>
<td>48</td>
<td>27</td>
</tr>
<tr>
<td>Pushing, no regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward in distr.</td>
<td>17</td>
<td>17</td>
<td>21</td>
<td>37</td>
<td>35</td>
<td>39</td>
<td>45</td>
<td>45</td>
<td>45</td>
<td>41</td>
</tr>
<tr>
<td>L1 regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward in distr.</td>
<td>14</td>
<td>26</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>10</td>
<td>21</td>
<td>21</td>
<td>26</td>
<td>25</td>
</tr>
<tr>
<td>Pushing, L1 regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward in distr.</td>
<td>-11</td>
<td>4</td>
<td>-1</td>
<td>-9</td>
<td>-9</td>
<td>-9</td>
<td>-3</td>
<td>-12</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>L1 regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward in distr.</td>
<td>3</td>
<td>37</td>
<td>10</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>26</td>
<td>26</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>Pushing, L1 regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward in distr.</td>
<td>22</td>
<td>8</td>
<td>13</td>
<td>18</td>
<td>19</td>
<td>18</td>
<td>9</td>
<td>10</td>
<td>6</td>
<td>16</td>
</tr>
</tbody>
</table>

### Figure 9. Rank correlations between metrics and OOD reward, with and without adjusting for informativeness.

<table>
<thead>
<tr>
<th></th>
<th>GT</th>
<th>MLP</th>
<th>MLP (1)</th>
<th>MLP (2)</th>
<th>MLP (3)</th>
<th>MLP (4)</th>
<th>MLP (5)</th>
<th>MLP (6)</th>
<th>MLP (7)</th>
<th>ReachCube, no regularization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reward OOD1</td>
<td>9</td>
<td>32</td>
<td>-6</td>
<td>20</td>
<td>20</td>
<td>26</td>
<td>45</td>
<td>41</td>
<td>51</td>
<td>43</td>
</tr>
<tr>
<td>Pushing, no regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward OOD1</td>
<td>0</td>
<td>9</td>
<td>5</td>
<td>17</td>
<td>21</td>
<td>23</td>
<td>26</td>
<td>21</td>
<td>36</td>
<td>53</td>
</tr>
<tr>
<td>L1 regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward OOD2</td>
<td>-2</td>
<td>-17</td>
<td>10</td>
<td>16</td>
<td>13</td>
<td>14</td>
<td>9</td>
<td>14</td>
<td>14</td>
<td>17</td>
</tr>
<tr>
<td>Pushing, L1 regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward OOD2</td>
<td>0</td>
<td>-4</td>
<td>-4</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>-4</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>GT</th>
<th>MLP</th>
<th>MLP (1)</th>
<th>MLP (2)</th>
<th>MLP (3)</th>
<th>MLP (4)</th>
<th>MLP (5)</th>
<th>MLP (6)</th>
<th>MLP (7)</th>
<th>ReachCube, no regularization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reward OOD1</td>
<td>2</td>
<td>32</td>
<td>-16</td>
<td>7</td>
<td>10</td>
<td>14</td>
<td>35</td>
<td>25</td>
<td>32</td>
<td>37</td>
</tr>
<tr>
<td>Pushing, no regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward OOD1</td>
<td>-18</td>
<td>9</td>
<td>-22</td>
<td>-6</td>
<td>-3</td>
<td>1</td>
<td>13</td>
<td>8</td>
<td>10</td>
<td>18</td>
</tr>
<tr>
<td>L1 regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward OOD2</td>
<td>4</td>
<td>2</td>
<td>12</td>
<td>17</td>
<td>16</td>
<td>14</td>
<td>22</td>
<td>12</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>Pushing, L1 regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward OOD2</td>
<td>-9</td>
<td>-7</td>
<td>-16</td>
<td>-3</td>
<td>3</td>
<td>-2</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>GT</th>
<th>MLP</th>
<th>MLP (1)</th>
<th>MLP (2)</th>
<th>MLP (3)</th>
<th>MLP (4)</th>
<th>MLP (5)</th>
<th>MLP (6)</th>
<th>MLP (7)</th>
<th>ReachCube, no regularization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reward OOD1</td>
<td>11</td>
<td>34</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>16</td>
<td>38</td>
<td>32</td>
<td>32</td>
<td>37</td>
</tr>
<tr>
<td>Pushing, no regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward OOD1</td>
<td>-10</td>
<td>19</td>
<td>-7</td>
<td>-3</td>
<td>-2</td>
<td>-3</td>
<td>-5</td>
<td>-19</td>
<td>-13</td>
<td>10</td>
</tr>
<tr>
<td>L1 regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward OOD2</td>
<td>-4</td>
<td>-11</td>
<td>-4</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>-12</td>
<td>-3</td>
</tr>
<tr>
<td>Pushing, L1 regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward OOD2</td>
<td>-7</td>
<td>-1</td>
<td>-3</td>
<td>-3</td>
<td>-1</td>
<td>-3</td>
<td>-5</td>
<td>-1</td>
<td>4</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>GT</th>
<th>MLP</th>
<th>MLP (1)</th>
<th>MLP (2)</th>
<th>MLP (3)</th>
<th>MLP (4)</th>
<th>MLP (5)</th>
<th>MLP (6)</th>
<th>MLP (7)</th>
<th>ReachCube, no regularization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reward OOD1</td>
<td>20</td>
<td>42</td>
<td>3</td>
<td>12</td>
<td>12</td>
<td>17</td>
<td>39</td>
<td>26</td>
<td>42</td>
<td>51</td>
</tr>
<tr>
<td>Pushing, no regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward OOD1</td>
<td>-15</td>
<td>-11</td>
<td>-1</td>
<td>-7</td>
<td>-1</td>
<td>-7</td>
<td>-8</td>
<td>-6</td>
<td>-2</td>
<td>-9</td>
</tr>
<tr>
<td>L1 regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward OOD2</td>
<td>9</td>
<td>6</td>
<td>-2</td>
<td>16</td>
<td>11</td>
<td>18</td>
<td>10</td>
<td>7</td>
<td>14</td>
<td>-12</td>
</tr>
<tr>
<td>Pushing, L1 regularization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reward OOD2</td>
<td>-6</td>
<td>-0</td>
<td>-6</td>
<td>-6</td>
<td>-2</td>
<td>-2</td>
<td>-4</td>
<td>-4</td>
<td>-3</td>
<td>-1</td>
</tr>
</tbody>
</table>
Figure 10. Fractional success on ReachCube (top) and Pushing (bottom), split according to low (blue), medium-high (orange), and almost perfect (green) disentanglement. Results for ReachCube are also reported in Fig. 5 in Section 5.2.

particularly significant, while in ReachCube it is negligible.

Sample efficiency. In addition to the analysis reported in the main paper, we performed an analysis on representation properties affecting sample efficiency, which we summarize in Fig. 11 for ReachCube and Fig. 12 for Pushing. Specifically, we stored checkpoints of our policies at \( t \in \{20k, 50k, 100k, 400k\} \) for ReachCube and \( t \in \{200k, 500k, 1M, 3M\} \) for Pushing. We then evaluate policies at these time steps on the same three different environments as before: (1) on the cube colors from training; (2) on the OOD1 cube colors; and (3) on the OOD2-sim cube colors.

On ReachCube (Fig. 11), we observe very similar trends with and without regularization: Unsupervised metrics (ELBO and reconstruction loss) display a correlation with the training reward, as do the supervised informativeness metrics (GBT and MLP). This is strongest on early timesteps, meaning these scores could be important for sample efficiency. Similarly, we observe a correlation with the disentanglement scores DCI, MIG and SAP. With the help of the additional evaluation of rewards adjusted for MLP informativeness, we can attribute this correlation again to this common confounder. Lastly, we see that the generalization scores are correlated with generalization of the corresponding policies under OOD1 and OOD2 shifts for all recorded time steps.

On Pushing (Fig. 12), many correlations at early checkpoints are significantly reduced, especially with regularization. This behavior might be due to the more complicated nature of the task, which involves learning to reach the cube first, and then push it to the goal. Correlations are primarily seen towards the end of training, with similar spurious correlations with disentanglement as elaborated above. Importantly, correlations between generalization scores and policy generalization under the same distribution shift remain strong and statistically significant.

Generalization to a novel shape. As mentioned in Section 5.2 on the ReachCube task, we also tested generalization w.r.t. a novel object shape by replacing the cube with an unmovable sphere. Remember, this corresponds to a strong OOD2-type shift, since shape was never varied when training the representations. We then evaluated a subset of 960 trained policies as before, with the same color splits. Surprisingly, the policies appear to handle the novel shape as we see from the histograms in Fig. 13 in terms of success and final distance. In fact, when the sphere has the same colors that the cube had during policy training, all policies get closer than 5cm to the sphere on average, with a mean
**Figure 11.** Sample efficiency analysis for ReachCube. Rank correlations of rewards with relevant metrics along multiple time steps

<table>
<thead>
<tr>
<th>reward in desired</th>
<th>GBT</th>
<th>MLP</th>
<th>ELBO</th>
<th>recons. loss</th>
<th>GS-OOD1</th>
<th>GS-OOD2-sim</th>
<th>GS-OOD2-real</th>
<th>DCI</th>
<th>MG</th>
<th>SAP</th>
<th>modularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>54</td>
<td>41</td>
<td>28</td>
<td>3</td>
<td>49</td>
<td>52</td>
<td>42</td>
<td>37</td>
<td>50</td>
<td>36</td>
<td>32</td>
<td>-84 -47 -32 -36</td>
</tr>
<tr>
<td>(adj. MLP) reward in desired</td>
<td>7</td>
<td>-7</td>
<td>-14</td>
<td>-27</td>
<td>-1</td>
<td>2</td>
<td>-15</td>
<td>11</td>
<td>-4</td>
<td>-13</td>
<td>-3 -13 0 3</td>
</tr>
<tr>
<td>reward OD1</td>
<td>42</td>
<td>1</td>
<td>11</td>
<td>40</td>
<td>29</td>
<td>24</td>
<td>34</td>
<td>48</td>
<td>39</td>
<td>37</td>
<td>-34 -49 -26 -36</td>
</tr>
<tr>
<td>(adj. MLP) reward OD1</td>
<td>4</td>
<td>-11</td>
<td>-10</td>
<td>-10</td>
<td>-1</td>
<td>2</td>
<td>-10</td>
<td>8</td>
<td>14</td>
<td>11</td>
<td>-3 -18 -10 10</td>
</tr>
<tr>
<td>reward OD2</td>
<td>34</td>
<td>0</td>
<td>4</td>
<td>14</td>
<td>29</td>
<td>26</td>
<td>11</td>
<td>19</td>
<td>-4</td>
<td>-7</td>
<td>-1 -11 6 -1</td>
</tr>
<tr>
<td>(adj. MLP) reward OD2</td>
<td>5</td>
<td>-6</td>
<td>-2</td>
<td>7</td>
<td>11</td>
<td>-2</td>
<td>-12</td>
<td>-12</td>
<td>-11</td>
<td>3</td>
<td>9 -6 -2 -7</td>
</tr>
</tbody>
</table>

**Figure 12.** Sample efficiency analysis for Pushing. Rank correlations of rewards with relevant metrics along multiple time steps

<table>
<thead>
<tr>
<th>reward in desired</th>
<th>GBT</th>
<th>MLP</th>
<th>ELBO</th>
<th>recons. loss</th>
<th>GS-OOD1</th>
<th>GS-OOD2-sim</th>
<th>GS-OOD2-real</th>
<th>DCI</th>
<th>MG</th>
<th>SAP</th>
<th>modularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>42</td>
<td>36</td>
<td>33</td>
<td>50</td>
<td>48</td>
<td>42</td>
<td>45</td>
<td>49</td>
<td>45</td>
<td>49</td>
<td>45 -43 -35 -41</td>
</tr>
<tr>
<td>(adj. MLP) reward in desired</td>
<td>10</td>
<td>-2</td>
<td>1</td>
<td>17</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>17</td>
<td>7</td>
<td>12</td>
<td>-17 -11 -24</td>
</tr>
<tr>
<td>reward OD1</td>
<td>31</td>
<td>5</td>
<td>9</td>
<td>43</td>
<td>25</td>
<td>32</td>
<td>33</td>
<td>45</td>
<td>36</td>
<td>33</td>
<td>-41 -43 -15 -41</td>
</tr>
<tr>
<td>(adj. MLP) reward OD1</td>
<td>12</td>
<td>-10</td>
<td>-9</td>
<td>-0</td>
<td>4</td>
<td>3</td>
<td>9</td>
<td>5</td>
<td>10</td>
<td>12</td>
<td>18 -12 -15 -21</td>
</tr>
<tr>
<td>reward OD2</td>
<td>24</td>
<td>1</td>
<td>-3</td>
<td>11</td>
<td>20</td>
<td>-24</td>
<td>-17</td>
<td>22</td>
<td>-4</td>
<td>-9</td>
<td>-3 -19 4 7</td>
</tr>
<tr>
<td>(adj. MLP) reward OD2</td>
<td>-11</td>
<td>-5</td>
<td>-7</td>
<td>0</td>
<td>2</td>
<td>-2</td>
<td>-4</td>
<td>6</td>
<td>8</td>
<td>4</td>
<td>5 -15 -3 -10</td>
</tr>
</tbody>
</table>

*Note: The tables and figures are part of a document discussing representation learning for out-of-distribution generalization in reinforcement learning.*
success metric of about 95%. On sphere colors from the
OOD1 split, more than 98.5% move the finger closer than
this threshold, and on the strongest distribution shift (OOD2-
sim colors and cube replaced by sphere) almost 70% surpass
that threshold with an average success metric above 80%.

B.3 Deploying policies to the real world

In Fig. [14] we depict three representative episodes of testing
a reach policy on the real robot for the strong OOD shift
with a novel sphere object shape instead of the cube from
training.
Figure 13. Testing ReachCube policies under the same IID, OOD1 and OOD2 evaluation protocols regarding object color in simulation but replacing the cube with a novel shape in the form of a sphere.

Figure 14. Transferring ReachCube models to the real robot setup without any fine-tuning on a green sphere (unseen shape and color).