Robotics: Science and Systems 2022
New York City, NY, USA, June 27-July 1, 2022

Invariance Through Latent Alignment

Takuma Yoneda[*,§], Ge Yang[*,†,‡] Matthew R. Walter[§], Bradly C. Stadie[§]

[§]Toyota Technological Institute at Chicago (TTIC)
[†]Institute of Artificial Intelligence and Fundamental Interactions (IAIFI)
[‡]Computer Science and Artificial Intelligence Laboratory (CSAIL), MIT
takuma@ttic.edu,geyang@csail.mit.edu

***Abstract*—A robot's deployment environment often involves perceptual changes that differ from what it has experienced during training. Standard practices such as data augmentation attempt to bridge this gap by augmenting source images in an effort to extend the support of the training distribution to better cover what the agent might experience at test time. In many cases, however, it is impossible to know test-time distribution-shift *a priori*, making these schemes infeasible. In this paper, we introduce a general approach, called Invariance through Latent Alignment (ILA), that improves the test-time performance of a visuomotor control policy in deployment environments with unknown perceptual variations. ILA performs unsupervised adaptation at deployment-time by matching the distribution of latent features on the target domain to the agent's prior experience, without relying on paired data. Although simple, we show that this idea leads to surprising improvements on a variety of challenging adaptation scenarios, including changes in lighting conditions, the content in the scene, and camera poses. We present results on calibrated control benchmarks in simulation—the distractor control suite—and a physical robot under a sim-to-real setup. Video and code available at: https://takuma.yoneda.xyz/projects/ila**

I. Introduction

Reinforcement learning for control has achieved great success in a wide variety of challenging sensory-motor control tasks, including agile drone flight [20, 21, 26], deformable object manipulation [41], and quadruped locomotion [19, 24, 28, 30]. In comparison to their classical model-predictive control counterparts, reinforcement learning-based approaches enables the use of more realistic forward dynamics model in the form of a physics simulator. Improvements in rigid-body simulator technologies [27, 38] allows reinforcement learning algorithms to overcome their prohibitively-high sample complexity by first training in simulation and then deploying directly on the physical robot. Differences, however, still exist between what the controller experiences in the simulator and in the physical environment in the form of a *sim-to-real gap*. In particular, the ability to produce visuomotor control policies that remain robust when perceptual conditions change during deployment, remains an open problem.

Consider the illustration in Figure 1. A common approach to battle domain shift is to expose the agent to a large variety of data during training with the hope that the training distribution provides adequate coverage over what the agent will experience in the wild. When the simulator is extensible, one can use domain randomization to generate more diverse training

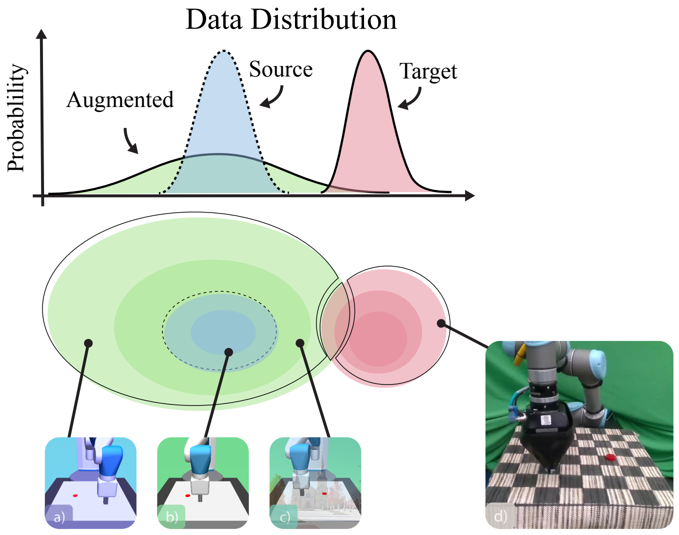


Fig. 1: Data augmentation improves the coverage of the training distribution at the expense of learning complexity and performance. Insets: (a, c) augmented training data; (b) the original input image from the source domain; and (d) the input image in the target domain. Data augmentation fails to provide sufficient coverage of the unknown deployment condition, so the learned controller fails to accomplish the task.

data [29, 37]. Alternatively, one can use data augmentation mechanisms to decorate existing data [12, 42]. Both approaches aim to produce visual features that are invariant to perceptual changes orthogonal to the task. Such invariance does not come for free, however. Additional training slows down the wall-clock speed of the training process, while domain randomization requires manual tuning [2] and relies on the assumption that the policy network has sufficient capacity to handle the increased support of the input distribution. The added complexity can negatively affect model and policy performance [12, 22]. Further more, hidden beneath is the assumption that one needs to know roughly what types of perceptual shift would occur during deployment. Failure can happen when the target domain is not known *a priori* and falls *out-of-distribution*, resulting in a fumbling robot that is unable to self-correct (see Figure 1).

Different from these prior approaches that produce invariance by memorizing what is irrelevant to the task during training, we consider a more challenging, but also more realistic scenario in which the specifics of the deployment is not known in advance.

[*]Equal contribution.

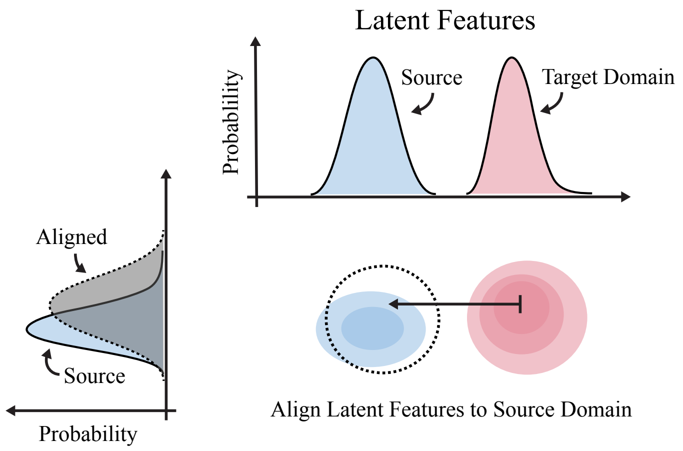


Fig. 2: Latent features of the network experience a covariant shift in the target domain. Our proposed method, **I**nvariance through **L**atent **A**lignment (**ILA**), counteracts this domain shift by projecting the features back to the known distribution of latent features on the source domain. We formulate this as a distribution-matching objective in Section IV.

This requires the agent to truly generalize *out-of-distribution*, without prior knowledge of the target environment. We further assume that reward supervision is unavailable, so fine-tuning via reinforcement learning is out of the question. This might seem to be an impossible task, but it does implicitly make the assumption that the *same task* that the agent was optimized for during training remains well-defined in the target domain. This means that the agent has *some* notion of *what it knows* despite of the sudden appearance of many unknowns that are not required for the task of interest. Without further assumptions or loss of generality, this *out-of-distribution* generalization problem can be formulated as *unsupervised policy adaptation* between two MDPs that share the same latent dynamics and reward structure, but with distinct pixel observations (See Figure 3).

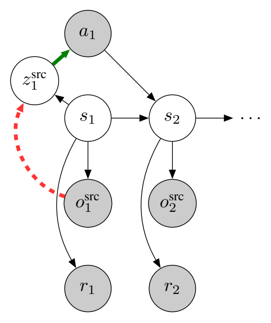
In this paper, we investigate ways to improve generalization under this challenging scenario. Rather than battling domain shift by baking perceptual invariance explicitly into the network during training, we demonstrate a way to produce feature invariance at the time of deployment by taking advantage of the fact that the task of interest remains the same, therefore what the agent experiences internally should remain the same as well. We collect latent features collected during training as examples of what the agent *knows* about the task. On the target domain (see Figure 2), the new latents are shifted from these prior distributions. Our goal in unsupervised policy adaptation is to match the distribution of these latent features on the target domain with those that appeared during training. This unsupervised learning objective, which we refer to as *latent alignment*, does not require paired image data between the source and the target, and can be applied to any agent without imposing specific requirements of how it is trained. To distinguish our approach from prior works that attempt to produce generalization by baking invariances into the policy at training time, we refer to our method as **I**nvariance through **L**atent **A**lignment (**ILA**).

II. Related Work

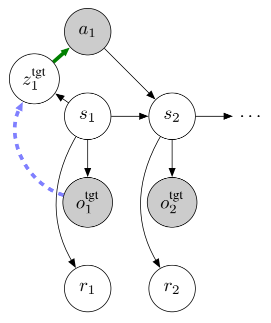
A large body of work is dedicated to improving the ability for neural networks to generalize. These works can largely be placed under two categories—the first category, including domain randomization [37], data augmentation [11], invariant risk minimization [4, 43], and meta-learning [7], all make the assumption that one has a rough idea of what type of perceptual change is going to occur during deployment. For example, to get the best result with data augmentation, one needs to fine-tune the weights between different augmentation mechanisms because each produces a different type of invariance [23]. Variations in the camera pose, for instance, is a common problem in robotics. Yet it can not be fixed by augmenting images alone [33]. Invariance to the projective geometry requires randomizing camera extrinsics during rendering [11].

Similarly, meta-learning makes the assumption that one has access to a meta-distribution of task-environment pairs. This is an even stronger assumption than those typically made by supervised learning, which merely requires that training data is sampled from an *i.i.d.* that covers the test distribution. In many cases and especially in reinforcement learning, generalization comes from exposure to a large amount of diverse data [6]. Meta-learning offers little gain procedural-wise, because a meta-learning reinforcement learning algorithm is identical to multi-task training from a task distribution plus fine-tuning that relies on rewards being available at test time.

The second category of methods, which this proposal is also a member of, makes no explicit assumptions of what type of distribution shift occurs in the target domain. These methods include approaches such as *unsupervised domain adaptation* [16, 18], *train-on-test* [13, 35, 40], and *tailoring* [1]. These methods all tackle adaptation face-on, as an out-of-distribution generalization problem. Under this view, what happens in the target domain can not be known in advance when training the model. These approaches differentiate the adaptation phase from the training phase by what types of information is privileged, i.e., being only available during training. Examples include ground-truth labels under supervised learning, segmentation masks for dense predictions in computer vision, and instrumented rewards during reinforcement learning. Without these forms of privileged information available at test time, these approaches cast adaptation as an unsupervised, or self-supervised, learning process, with the main differences between methods being the learning objective, optimization details, and ways that they augment the data. In particular, test-time adaptation by entropy minimization [40] shows that fine-tuning just the two parameters in layernorm gives better performance than fine-tuning the entire network. CycADA [17] and FCN in-the-wild [16] use a cycle-consistent, adversarial loss for matching pixel-wise dense features. Some of these methods [32] produce feature alignment by synthesizing image pixels, whereas our proposal directly enforces distributional alignment in a compact latent space using an adversarial objective, without reconstructing image patches. This long line



(a) Training on $\mathcal{M}_{\text{src}}$



(b) Adaptation on $\mathcal{M}_{\text{tgt}}$

Fig. 3: A Markov decision process gives rise to two different observations o_t^{src} and o_t^{tgt}. o_t^{src} is accessible during training, whereas o_t^{tgt} is only accessible during deployment. The red dashed arrow indicates the learned inference network F, produced as part of the policy during training. During deployment, the latent features produced by F, z_t^{tgt} (blue dashed arrow), experience a domain-shift. The reward further becomes unobservable. Our goal is to make the representation domain-invariant, such that the policy $\pi(a|o^{\text{tgt}})$ (frozen policy head in green) can succeed in the target domain.

of work derive from classical unsupervised domain adaptation methods that pre-date adversarial generative techniques. [8, 25] and [34], for instance, directly minimizes the measure of Maximum Mean Discrepancy (MMD) to great effect.

III. Unsupervised Policy Adaptation

Unsupervised policy adaptation is a setting that involves two distinct domains — a source domain and a target domain. These two domains share the same underlying MDP and task structure, but has different observation conditions. In the target domain, the agent only has access to observations o_t^{tgt} and its own actions a_t, but not the corresponding rewards or the ground-truth state s_t (see Figure 3). A practical example is a robot that is trained with images in a clean, simulated environment that now has to work in-the-wild, in the presence of visual distractors and changes in the lighting condition or the mounting pose of the video camera. These variations could lead to significantly different image observations. As a result of this shift, deploying an agent trained in the source domain directly in the target domain (i.e., zero-shot transfer) generally results in poor performance.

Formally, we consider an infinite horizon Markov decision process (MDP) [31] $\mathcal{M}$ parameterized via the tuple $\langle S, A, O, R, P, \gamma \rangle$, where S and A are the state and action spaces. $P : S \times A \mapsto S$ is the transition function, $R : S \times A \mapsto \mathbb{R}$ is the scalar reward, and γ is the discount factor. The agent receives a stream of observations $o \in O$. We assume a fully-observable setting where a single observation carries enough information to decide an appropriate action. In the source domain $\mathcal{M}_{\text{src}}$ (see Figure 3a), we can use reinforcement learning to produce an optimal policy $\pi : O \times A \mapsto [0, 1]$ that maximizes the expected discounted return $\mathcal{J} = \mathbb{E}\left[\sum_{\infty} \gamma^t R(s_t, a_t)\right]$. In the target domain (see Figure 3b) however, the reward is not observable therefore we can not rely on reinforcement learning for fine-tuning. Nevertheless the task structure remains identical to that of the source domain. We assume that the policy π consists of an encoder $F : O \mapsto Z$, where Z is a compact latent space, and a policy head $\pi_z : Z \times A \mapsto [0, 1]$ shown as green arrows. The goal of unsupervised policy adaptation is to find ways to battle this distribution shift, so that the resulting, adapted policy can succeed on the task in $\mathcal{M}_{\text{tgt}}$.

IV. Invariance Through Latent Alignment

When we ask if the agent can perform in the target domain, we are effectively making the assumption that the task, and the underlying MDP has not changed. One way to factorize the problem is to divide the policy into two modules (see Figure 3a). A *policy head* $\pi(a|z)$ that we keep *frozen* during the adaptation process, and an *encoder* $F(z|o)$ that we adapt. Ideally, the latent feature z captures what the policy needs to know to accomplish the task. Hence we can formulate unsupervised policy adaptation as a distribution-matching objective that "aligns" the distribution $P_\pi(z_{\text{tgt}})$, with the one in the source domain, $P_\pi(z_{\text{src}})$ (see Figure 2).

This overall *sim-and-adaptation* pipeline starts in the source domain with collecting latent features z_{src} into a buffer. The agent carries these data into the deployment environment. Then during adaptation, it optimizes two objectives. The first is a minimax objective that focuses on individual latent features (D in Figure 4). The second is an cooperative dynamics consistency objective that consists of both the forward and the inverse kinematics prediction error (C_{inv} and C_{fwd} in Figure 4). The training procedure partially resembles a generative adversarial network (GAN) [9] with two key distinctions. First, we do not reconstruct raw pixel observations but instead directly match the distribution of the latent features z. This has the benefit that we do not require a generator that incur additional space and optimization overhead. Second, we find it helpful to pretrain the dynamics consistency module at the beginning of the adaptation process because it reduces the wall-clock time of the procedure.

We cover details of the procedure below.

A. Collecting Latent Features in The Source Domain

In the source domain we collect latent vectors z_t^{src} and actions a_t into a buffer

$$\mathcal{B}_{\text{src}} = \{z_0^{\text{src}}, a_0; z_1^{\text{src}}, a_1, \dots\} \quad \text{where} \quad z = F(o_t^{\text{src}}). \tag{1}$$

The trajectories, $\tau_{\text{src}} = \{o_0, a_0; o_1, a_1, \dots\}$ are sampled from $\mathcal{M}_{\text{src}}$ with an exploration policy π that can be different from the pretraind policy for the task. In our experiment we found that using a random policy $\bar{\pi}$ is unexpectedly effective, with the added benefit that the same policy can be used in the target domain.

B. Dynamics Consistency

To match the joint distribution $P(z_t^{\text{tgt}}, a_t, z_{t+1}^{\text{tgt}})$ with those on the source domain, we introduce a dynamics consistency

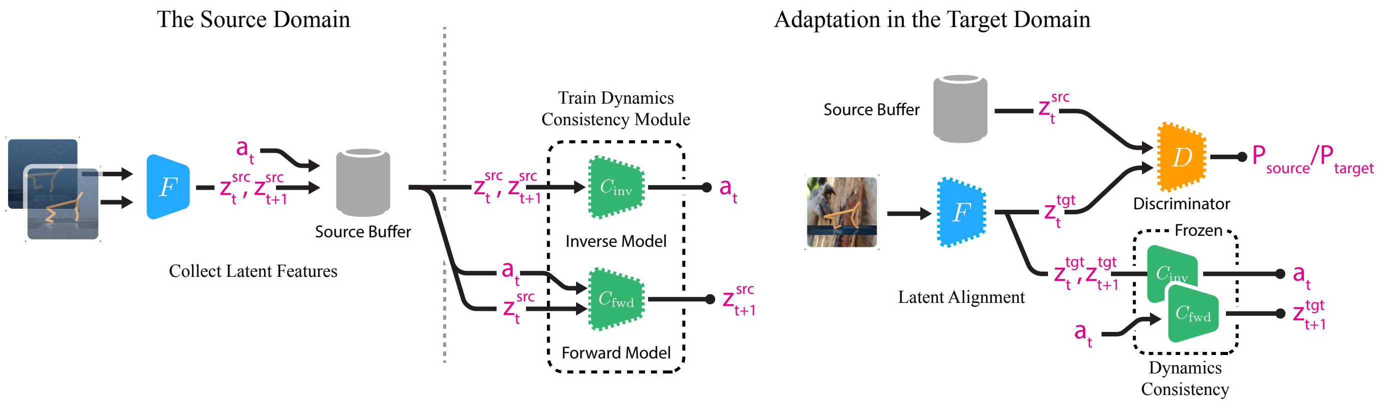


Fig. 4: Training and adaptation phases of invariance through latent alignment. The encoder F takes an observation of target domain, and learns to fool the discriminator, while the discriminator D predicts whether the input is an encoded target observation or a latent sample from source buffer. This adversarial training encourages the distribution of encoder outputs to be similar to the latent embedding sampled from the source buffer. C_{fwd} and C_{inv} are the forward and inverse dynamics networks that guides the encoder adaptation.

Algorithm 1 Populating The Source Buffer

```
Require: Encoder $F$, empty buffer $\mathcal{B}_{\text{src}}$, random policy $\bar{\pi}$
1: for step in $1, \ldots, N$ do                          ▷ Collect latent features
2:     Sample $o_t, a_t, o_{t+1} \sim P_{\bar{\pi}}(\mathcal{M}_{\text{src}})$
3:     Encode $z_t, z_{t+1} \leftarrow F(o_t), F(o_{t+1})$
4:     $\mathcal{B}_{\text{src}} \leftarrow \mathcal{B}_{\text{src}} \cup (z_t, a_t, z_{t+1})$
```

loss which is the sum of the ℓ^2 error in the forward and inverse dynamics predictions

$$\mathcal{L}_{\text{dyn}}(z_t, z_{t+1}, a_t) = \|C_{\text{fwd}}(z_t, a_t) - z_{t+1}\|^2 + \|C_{\text{inv}}(z_t, z_{t+1}) - a_t\|^2. \tag{2}$$

$C_{\text{fwd}}(z_{t+1}|z_t, a_t)$ is the forward kinematics model that predicts the next latent z_{t+1} given the previous latent z_t and a_t. $C_{\text{inv}}(a_t|z_t, z_{t+1})$ is the inverse model that predicts the action a_t associated with the transition from z_t to z_{t+1}. We found it was not necessary to scale the two terms separately as it worked well enough.

We optimize the parameters of C_{fwd}, C_{inv} and F in a *cooperative* manner as opposed to an adversarial one. C_{fwd} and C_{inv} are optimized using latent transitions sampled from the source buffer $\mathcal{B}_{\text{src}}$

$$C_{\text{fwd}}, C_{\text{inv}} = \underset{C_{\text{fwd}}, C_{\text{inv}}}{\arg\min}\ \mathbb{E}_{z_t, a_t, z_{t+1} \sim \mathcal{B}_{\text{src}}}\left[\mathcal{L}_{\text{dyn}}(z_t^{\text{src}}, z_{t+1}^{\text{src}}, a_t^{\text{src}})\right].$$

To update the encoder F we sample transitions using a random policy $\bar{\pi}$ from the target domain. We freeze the parameters of the two dynamics model when updating F.

$$\mathcal{J}_{\text{dyn}} = \mathbb{E}_{o_t, a_t, o_{t+1} \sim P_{\bar{\pi}}(\mathcal{M}_{\text{tgt}})}\left[\mathcal{L}_{\text{dyn}}(F(o_t^{\text{tgt}}), F(o_{t+1}^{\text{tgt}}), a_t^{\text{tgt}})\right]. \tag{3}$$

We found that a learning rate of $1e^{-6}$ worked sufficiently well for both, and we did not find it necessary to scale the two loss terms separately.

Algorithm 2 Invariance through Latent Alignment

```
Require: Pretrained encoder $F$, discriminator $D$, populated buffer
    $\mathcal{B}_{\text{src}}$, $C_{\text{fwd}}, C_{\text{inv}}$, random policy $\bar{\pi}$
1: for step in $1, \ldots T_{\text{dyn}}$ do               ▷ Pretrain dynamics networks
2:     Sample $z_t, a_t, z_{t+1} \sim \mathcal{B}_{\text{src}}$
3:     $\Delta_{C_{\text{fwd}}}, \Delta_{C_{\text{inv}}} \leftarrow \nabla_{C_{\text{fwd}}, C_{\text{inv}}} \mathcal{L}_{\text{dyn}}(z_t, z_{t+1}, a_t)$
4:     $C_{\text{fwd}}, C_{\text{inv}} \leftarrow$ Optim.step$(C_{\text{fwd}}, C_{\text{inv}}, \Delta_{C_{\text{fwd}}}, \Delta_{C_{\text{inv}}})$
5: for step in $1, \ldots T$ do                            ▷ Adaptation main loop
6:     Sample $z_t^{\text{src}}, a_t^{\text{src}}, z_{t+1}^{\text{src}} \sim \mathcal{B}_{\text{src}}$
7:     Sample $o_t^{\text{tgt}}, a_t^{\text{tgt}}, o_{t+1}^{\text{tgt}} \sim \mathcal{P}_{\bar{\pi}}(\mathcal{M}_{\text{tgt}})$
8:     Compute gradients:
9:     $\Delta_D \leftarrow \nabla_D \left[D(z_t^{\text{src}}) + (1 - D(F(o_t^{\text{tgt}})))\right]$     ▷ Discriminator
10:    $\Delta_{F_1} \leftarrow \nabla_F \left[D(z_t^{\text{src}}) + (1 - D(F(o_t^{\text{tgt}})))\right]$
11:    $\Delta_{F_2} \leftarrow \nabla_F \mathcal{L}_{\text{dyn}}(F(o_t^{\text{tgt}}), F(o_{t+1}^{\text{tgt}}), a_t^{\text{tgt}})$   ▷ Dyn. consistency
12:    $D \leftarrow$ Optim.step$(D, -\Delta_D)$
13:    $F \leftarrow$ Optim.step$(F, \Delta_{F_1} + \Delta_{F_2})$
```

C. Adversarial Loss

In addition to the dynamics consistency loss, we also introduce an adversarial learning objective (see Figure 4), where a discriminator D tries to distinguish between embeddings from the source domain z_t^{src} and those from the target domain z_t^{tgt}. We update the parameters of the encoder such that latent embeddings on the target domain are indistinguishable from those of the source domain. Using the earth-moving metric from [3], we express this distribution-matching objective as

$$\mathcal{J}_{\text{adv}} = \mathbb{E}_{z \sim \mathcal{B}_{\text{src}}}\left[D\left(z_t^{\text{src}}\right)\right] + \mathbb{E}_{P_{\bar{\pi}}(\mathcal{M}_{\text{tgt}})}\left[1 - D\left(F(o_t^{\text{tgt}})\right)\right]. \tag{4}$$

The encoder tries to minimize this objective while the discriminator acts as an adversary and seeks to maximize it, resulting in a GAN-like minimax game.

D. Putting Things Together

We adapt our encoder by minimizing a loss that combines both the adversarial loss $\mathcal{J}_{\text{adv}}$ (Eqn. 4) and the dynamics consistency loss $\mathcal{J}_{\text{dyn}}$ (Eqn. 3). Specifically, we solve for the

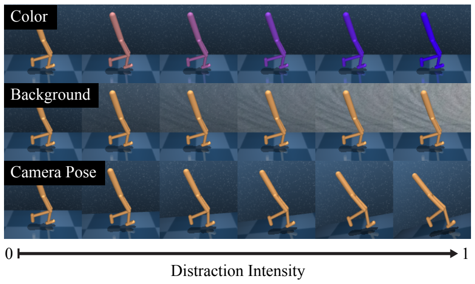


Fig. 5: Samples from the modified *distracting control suite*. Top row: color variations, middle row: background distractions, bottom row: camera pose variations.

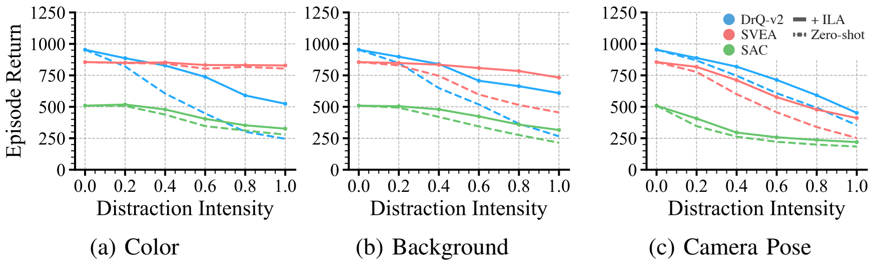


Fig. 6: The gain of applying invariance through latent alignment (ILA) to target domains with various distraction intensities. Dashed lines denote the performance of the baseline agent in the target environment (i.e., zero-shot transfer), while solid lines represent the performance gains of the agents with ILA.

parameters of the encoder through the following objective

$$\min_F \left[\max_D \mathcal{J}_{\text{adv}} + \mathcal{J}_{\text{dyn}}\right]. \tag{5}$$

We did not find it necessary to add additional scaling factors to balance the loss terms. Algorithm 1 and 2 summarizes the whole procedure. We train the dynamics consistency networks to convergence before running the adaptation loss, to improve the wall-time.

The proposed approach, invariance through latent alignment, does not affect the training procedure of the control agent in the source domain, so it can be applied to any pretrained agent. This is also an unsupervised adaptation procedure as it does not require reward supervision at test time, nor does it require paired source and target images.

V. Experiments

We want to understand the impact of test-time adaptation on an agent's ability to generalize out-of-distribution. This section will compare ILA with two state-of-the-art reinforcement learning baselines that uses data-augmentation: SVEA [14] and DrQ-v2 [42]. Recall from the introduction that these methods vie for increased generalization capabilities by expanding the support of the training distribution. We expect the invariance produced this way to be less performant than unsupervised adaptation at test time that only needs to focus on one specific instance of perceptual variation. In our experiments, we will also compare against *policy adaptation during deployment* (PAD, see [13]), a baseline that, like our method, adapts the policy without access to the reward at test time.

To further probe the generalization abilities of test-time adaptation, we conduct an experiment where we vary the intensity of environmental distractions. The results show that test-time adaptaion significantly increases the policy performance during deployment. We will conclude with some general discussion and remarks regarding the design tradeoffs involved in test-time adaptation.

a) Setup: We conduct experiments on nine domains from the DeepMind Control Suite (DMC, see [36]) and treat it as the *source* domain for training the RL agents. We use the Distracting Control Suite [33] as the *target* domain. Distracting control suite adds three types of distractions to DMC, including image background, random color texture, and changes to the camera pose. The intensity of these modes of distraction are calibrated. For details, refer to the accompanying report (see Stone et al. 33)

b) Modifications to Distracting Control Suite: The default configuration of distracting control suite changes distractions at the start of every episode (e.g., different background images are used at every episode). However, we are interested in measuring an agent's ability to perform adaptation across several episodes on the same target environment. Thus, we modify distracting control suite to sample a distraction once in the beginning of learning, and then use the same distraction across all learning epochs. This also ensures consistent evaluation across algorithms. In accordance with this change, we also modify the intensity benchmark from distracting control suite. In our experiments, intensity measures the deviation between an environment distraction and the train environment's default value. For example, intensity may measure how far the distracting color is from the default. Finally, we modify the environments to only apply a single distraction during testing (rather than all three) in order to better understand the impact of each type of distraction on overall performance. Figure 5 shows an example of distractions across intensities on Walker-walk domain.

Table I presents the results for the different distracting control suite domains in the presence of background distractions with an intensity level of 1.0. Specifically, we compare the test-time performance of SAC, SVEA, and DrQ-v2 in each domain with the episode rewards that we achieve when using ILA to adapt the encoder. The baseline algorithms employ image augmentation, which provides some robustness to variations at test time. Even then, however, we find that ILA improves the test-time generalization of all three baseline policies in most domains, often resulting in significant performance gains. In cases where ILA does not improve performance, the resulting reward is comparable to the baseline policy, i.e., ILA does not result in a performance degradation.

Figure 6 visualizes the performance of the different methods, averaged over the set of distracting control suite domains, as a function of the intensity of the distractions. Since the baseline methods are trained with image augmentation, they

TABLE I: Episode return in the target (test) environments (mean and standard deviation) before (zero-shot) and after (+ILA) adaptation for SAC, SVEA, and DrQ-v2 with background distraction at an intensity setting of 1.0. The performance of each baseline in the source (training) environments can be found in the Appendix.

	SAC		SVEA		DrQ-v2	
Domain	Zero-shot	+ILA	Zero-shot	+ILA	Zero-shot	+ILA
`ball_in_cup-catch`	$115^{\pm 50}$	$\mathbf{227^{\pm 222}}$	$490^{\pm 376}$	$\mathbf{987^{\pm 27}}$	$88^{\pm 39}$	$\mathbf{386^{\pm 425}}$
`cartpole-balance`	$434^{\pm 275}$	$\mathbf{585^{\pm 295}}$	$446^{\pm 330}$	$\mathbf{627^{\pm 258}}$	$273^{\pm 107}$	$\mathbf{322^{\pm 117}}$
`cartpole-swingup`	$182^{\pm 147}$	$\mathbf{369^{\pm 243}}$	$269^{\pm 365}$	$\mathbf{612^{\pm 213}}$	$82^{\pm 35}$	$\mathbf{247^{\pm 136}}$
`cheetah-run`	$169^{\pm 65}$	$\mathbf{248^{\pm 53}}$	$317^{\pm 137}$	$\mathbf{378^{\pm 55}}$	$100^{\pm 88}$	$\mathbf{393^{\pm 125}}$
`finger-spin`	$113^{\pm 162}$	$\mathbf{192^{\pm 196}}$	$391^{\pm 467}$	$\mathbf{943^{\pm 54}}$	$207^{\pm 328}$	$\mathbf{769^{\pm 206}}$
`finger-turn_easy`	$\mathbf{163^{\pm 99}}$	$146^{\pm 33}$	$278^{\pm 180}$	$\mathbf{491^{\pm 343}}$	$268^{\pm 241}$	$\mathbf{914^{\pm 44}}$
`reacher-easy`	$179^{\pm 65}$	$\mathbf{381^{\pm 76}}$	$75^{\pm 77}$	$\mathbf{624^{\pm 305}}$	$58^{\pm 32}$	$\mathbf{685^{\pm 211}}$
`walker-stand`	$330^{\pm 118}$	$\mathbf{364^{\pm 115}}$	$917^{\pm 138}$	$\mathbf{999^{\pm 12}}$	$630^{\pm 197}$	$\mathbf{868^{\pm 151}}$
`walker-walk`	$242^{\pm 142}$	$\mathbf{291^{\pm 134}}$	$866^{\pm 45}$	$\mathbf{924^{\pm 45}}$	$326^{\pm 195}$	$\mathbf{770^{\pm 140}}$

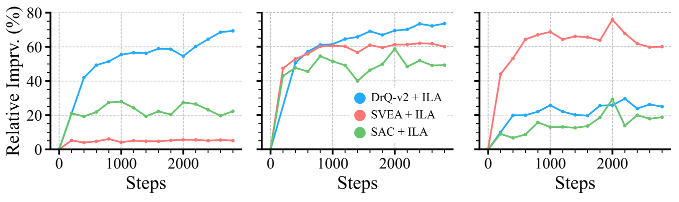
do exhibit some robustness to distraction. However, we see this robustness rapidly diminishes as the distraction intensity increases. In particular, large changes to camera pose or the image background proved challenging for standard augmentation procedures. Comparatively, ILA makes it much smoother and slower degradation of performance. This supports our hypothesis that adaptation powered by unsupervised learning can significantly widen the generalization abilities of learning algorithms.

A. *ILA on DeepMind Control Suite*

This section studies the impact of test-time adaptation on the DeepMind control suite. We begin by pretraining soft actor-critic (SAC) [10], SVEA, and DrQ-v2 in a non-distracting training-time environment. After training, we evaluate the learned policies on test environments with distractions of various intensities. This evaluation is zero-shot, i.e., there is no additional training in the test environment.

B. *Comparisons with PAD*

Similar to our approach, PAD pretrains the agent in a clean environment, and then adapts the agent via unsupervised objectives without assuming access to the target environment's reward function [13]. To evaluate the robustness of PAD to distractions, we consider distracting control suite with a fixed distraction intensity of 1.0. Table III compares the performance



(a) Color (b) Background (c) Camera Pose

Fig. 7: Relative improvement (compared to zero-shot) as a function of adaptation steps when applying ILA to different baseline policies. As in Figure 6, each point represents the mean over nine domains and five random seeds. The results correspond to a distraction intensity value of 1.

as the difference between the episode returns before and after adaptation along with the episode returns in the clean environment. It should be noted that PAD requires the policy to be trained along with an inverse dynamics prediction objective, whereas ILA does not. We include this additional auxiliary objective with soft actor-critic specifically for this experiment.

Across all environments, we see that PAD struggles to adapt to distractions at test time. We suspect this instability is caused by the large deviations in the latent variable distribution as a result of changes in the target environment. In particular, we posit that the signal from PAD's inverse dynamics head does not encourage the latent train and test distributions to match, whereas in ILA, it does.

C. *Sim-to-real Transfer*

We are interested in the ability of ILA to bridge the gap between simulated and real world robotics environments. In the *reaching* task, the target position is given by a red disc placed on a table. The agent's objective is to controls the arm so that the end-effector reaches this target location. Our goal is to train a policy in simulation, and then transfer the policy to a real UR-5 robot at test time. The same as with previous experiments, the test time agent receives no rewards. In both simulation and the real world, the policy's only input is an image from a camera placed in front of the robot and table. The action space is a 2D position controller that drives a small movement $(\delta x, \delta y)$ of the robotic gripper. See Figure 8 for the setup.

We carry out the experiment by first training a policy with SVEA [14] in simulation. For adaptation, we collect random trajectories in the real environment and then use ILA to align the simulated and real world experiences. We evaluated the success rate of zero-shot and the adapted policies over 20 real episodes. We consider the episode to be success if the gripper's tip overlaps with the target in the front-view image.

Due to the challenging domain shift between simulator and real world, the zero-shot policy fails to adapt adequately, repeating the same action of moving the gripper to an edge of the table ad infinitum, regardless of the given goal location. This results in a final success rate of 15%. On the same task,

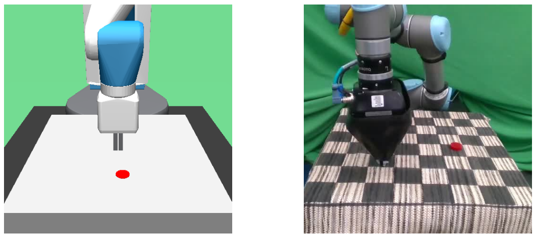


Fig. 8: Simulated and real reach environment. The goal location is denoted as a red disk that the robot must reach. Using ILA, we can transfer a policy trained in simulation (left) onto a real UR-5 robot (right). This adaptation requires no paired data and no rewards on the deployment environment, instead employing ILA for unpaired adaptation.

ILA is able to robustly against this domain shift, achieving a final success rate around 90%.

D. Further Discussion

a) Ablation Studies: In order to better understand the contribution of the different objectives to test-time generalization, we perform a series of ablations in which we omit either the dynamics consistency or the adversarial objectives. In these experiments, we use a pretrained DrQ-v2 network for the algorithm's base policy, and then perform adaptation across all distractions with an intensity value of 1.0. The results in Table II show that the adversarial training is critical to adapt the latent representation in the target domain. Performing adaptation using only the dynamics consistency objective, i.e., $\arg\min_F \mathcal{J}_{\text{dyn}}$ (Eqn. 3) results in a significant decrease in performance. We theorize that the dynamics consistency objective helps to align latent transition manifolds when the latent distributions in source and target domains are reasonably close. If the latent distributions significantly differ, however, the input to the pretrained dynamics networks is largely out-of-distribution, and thus the gradients from dynamics consistency loss may negatively affect convergence.

Compared to the adversarial objective, ablating the dynamics consistency objective has surprisingly little effect on test-time generalization. It may be that the transition manifold in latent spaces are preserved despite the distractions, which then diminishes the net effect of the dynamics consistency objective.

TABLE II: Ablations with variants of ILA that remove inverse/forward dynamics, or the adversarial objectives. DrQ-v2 is used as a pretrained policy. We compute episodic returns from nine domains and five random seeds, and the results correspond to an intensity value of 1.

Distraction	Zero-shot	+ILA	+ILA w/o dyn.	+ILA w/o adv.
Background	$228^{\pm 232}$	$602^{\pm 300}$	$615^{\pm 289}$	$176^{\pm 221}$
Colors	$234^{\pm 245}$	$536^{\pm 320}$	$534^{\pm 327}$	$117^{\pm 96}$
Camera Pose	$345^{\pm 287}$	$417^{\pm 284}$	$407^{\pm 272}$	$208^{\pm 235}$

TABLE III: Comparison with PAD

Distraction	Zero-shot	+PAD	+ILA
None	$835^{\pm 230}$	—	—
Background	$213^{\pm 247}$	$279^{\pm 271}$	$\mathbf{425^{\pm 292}}$
Colors	$230^{\pm 263}$	$271^{\pm 300}$	$\mathbf{402^{\pm 339}}$
Camera Pose	$319^{\pm 265}$	$326^{\pm 259}$	$\mathbf{412^{\pm 275}}$

b) Pre-Filling The Replay Buffer: We implicitly make the assumption that a behavior policy $\bar{\pi}$ is available that can be used to generate trajectory data on both the source and the target domain with similar state visitation, and transition probabilities. To achieve good performance with the adapted policy, such distribution should also cover important states with higher reward. To our surprise, a simple scheme where we pre-fill both the source and target buffer using the random policy works sufficiently well.

VI. Closing Remarks

We introduced *invariance through latent alignment*, an unsupervised approach that matches the distribution of feature vectors in the latent space, to improve the test-time performance of a learned visuomotor control policy. Empirical results show that as discrepancies between the training and deployment environments become more intense, invariance through latent alignmenthas a large competitive edge over alternatives such as data augmentation techniques. The problem of test-time adaptation in visual reinforcement learning using unsupervised test-time trajectories is relatively new, but has thus far shown great relevance and promise in robotics [11, 13], where a sim-to-real pipeline has been at the fore-front of recent progress [20, 28, 41].

A problem that remains is how to sample trajectories for adaptation. Technically, the distribution of trajectories and their latent representation on the source domain depends on the exploration policy to collect data. There are state transitions that are only accessible through a performant policy. To our surprise, simply using the random policy to collect data on both the source and the target domain largely by-passes this issue and achieves good performance post-adaptation. Future work in this area might want to develop better techniques for data generation at test time, so that the collected trajectory data better resembles those collected by the agent in the source buffer. One version towards this direction resembles generative adversarial imitation learning (GAIL, see [15]). Another open problem is that the performance does not improve monotonically upon more unsupervised updates. Finding an improvement scheme that is provably monotonic would be interesting.

Finally, although the adversarial distribution matching objective proposed here produce measurable improvements, better ways to align latent features are still needed. For a recent work in this direction, we refer the reader to *adversarial support alignment* [39].

VII. Acknowledgements

This work was funded in part by the National Science Foundation under grant IIS-1830660. Ge Yang is supported by the National Science Foundation Institute for Artificial Intelligence and Fundamental Interactions (IAIFI, https://iaifi.org/) under the Cooperative Agreement PHY-2019786. The authors would like to thank Adam Bohlander and Greg Shakhnarovich at the Toyota Technological Institute at Chicago (TTIC) for computing resources and additional storage. This paper also benefited from thoughtful discussions with students in the TTIC robotics group. The authors would also like to acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing high performance computing resources.

References

[1] Ferran Alet, Maria Bauza, Kenji Kawaguchi, Nurullah Giray Kuru, Tomás Lozano-Pérez, and Leslie Kaelbling. Tailoring: encoding inductive biases by optimizing unsupervised objectives at prediction time. *Adv. Neural Inf. Process. Syst.*, 34:29206–29217, December 2021. ISSN 1049-5258.

[2] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *International Journal of Robotics Research*, 39(1):3–20, 2020.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

[4] Martín Arjovsky, L Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[6] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

[8] Bo Geng, Dacheng Tao, and Chao Xu. DAML: Domain adaptation metric learning. *IEEE Transactions on Image Processing*, 20(10):2980–2989, October 2011.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Adv. Neural Inf. Process. Syst.*, 27, 2014. ISSN 1049-5258.

[10] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[11] Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[12] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. *arXiv preprint arXiv:2007.04309*, 2020.

[13] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. *arXiv preprint arXiv:2007.04309*, 2020.

[14] Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep Q-learning with ConvNets and vision transformers under data augmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[15] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[16] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. FCNs in the wild: Pixel-level adversarial and constraint-based adaptation. *arXiv preprint arXiv:1612.02649*, 2016.

[17] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017.

[18] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1989–1998, 2018.

[19] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), January 2019.

[20] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: Learning agile flight in dynamic environments. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2018.

[21] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Deep drone acrobatics. In *Proceedings of Robotics: Science and Systems (RSS)*, July 2020.

[22] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020.

[23] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020.

[24] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47),

October 2020.

[25] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[26] Antonio Loquercio, Elia Kaufmann, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59), October 2021.

[27] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac Gym: High performance GPU-based physics simulation for robot learning. *arXiv Preprint arXiv:2108.10470*, 2021.

[28] Gabriel Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. In *Proceedings of Robotics: Science and Systems (RSS)*, 2022.

[29] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher Pal, and Liam Paull. Active domain randomization. *arXiv preprint arXiv:/1904.04762*, 2019.

[30] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62), January 2022.

[31] Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, August 2014.

[32] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. Learning from synthetic data: Addressing domain shift for semantic segmentation. *arXiv preprint arXiv:1711.06969*, 2017.

[33] Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The distracting control suite—A challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021.

[34] Yu Sun, Eric Tzeng, Trevor Darrell, and Alexei A Efros. Unsupervised domain adaptation through self-supervision. *arXiv preprint arXiv:1909.11825*, 2019.

[35] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[36] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind Control Suite. *arXiv preprint arXiv:1801.00690*, 2018.

[37] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[38] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[39] Shangyuan Tong, Timur Garipov, Yang Zhang, Shiyu Chang, and Tommi S Jaakkola. Adversarial support alignment. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

[40] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

[41] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to manipulate deformable objects without demonstrations. *arXiv preprint arXiv:1910.13439*, 2019.

[42] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[43] Amy Zhang, Clare Lyle, Shagun Sodhani, Angelos Filos, Marta Kwiatkowska, Joelle Pineau, Yarin Gal, and OPTdoina Precup. Invariant causal prediction for block MDPs. *arXiv preprint arXiv:2003.06016*, 2020.

Appendix

A. Architectures

This section describes the architectures of encoder F, discriminator D, inverse dynamics C_{inv} and forward dynamics C_{fwd}. Encoder architectures follow the originally presented design choices of each base policy except for DrQ-v2. We take the part of the original network that produces a latent for the actor, and use it as an encoder F. In all of SAC (of the version used in [14]), SVEA and PAD, this shared latent is set to have dimension 100. For DrQ-v2, we took the entire network architecture from SVEA. The discriminator consists of a linear layer with hidden dimension 100 followed by Layer Normalization (LN) [5] and tanh activation, and a three layer multi-layer perceptron (MLP) with and ReLU activations. Inverse dynamics network C_{inv} is five layer MLP and ReLU activations. It takes the concatenated latents as its input. Forward dynamics network C_{fwd} takes action and latent, and encode them separately followed by concatenation and further layers. The action is fed it to a linear layer followed by LN, another linear layer and ReLU with hidden dimension 100. The latent is fed to three layer MLP where the first activation is LN and others are ReLU. The both encoded inputs are concatenated and fed to 4 layer MLP with ReLU activations followed by LN and tanh activation. In all layers except for those explicitly mentioned, hidden dimension is set to $1,024$.

B. Hyperparameters

This section details the hyperparameter settings that were used for the experimental evaluation. Table IV lists the hyperparameters relevant to pretraining the dynamics networks, while Table V provides those relevant to adaptation. For the base policies, we adopted the hyperparameter settings and architecture choices from the original papers. Details for SAC and SVEA can be found in Hansen et al. [14], while Hansen et al. [12] provides settings for PAD. The dimension of the latent z_t differs depending on the encoder of the base policy, but all of the encoders in our experiments produce a latent with dimension 100.

TABLE IV: Hyperparameters for dynamics pretraining

Hyperparameter	Value
Steps (T_{dyn})	$100,000$
Batch size	256
Optimizer	RMSProp($\alpha = 0.99, \epsilon = 1.0 \times 10^{-8}$)
Learning rate (forward dynamics)	0.001
Learning rate (inverse dynamics)	0.001

TABLE V: Hyperparameters for adaptation

Hyperparameter	Value
Capacity of buffers (N_{buf})	$1,000,000$
Batch size	256
Discriminator updates per step	5
Gradient clipping	0.01
Optimizer	RMSProp($\alpha = 0.99, \epsilon = 1.0 \times 10^{-8}$)
Learning rate (encoder)	1.0×10^{-4} (for DrQ-v2)
	1.0×10^{-5} (otherwise)
Learning rate (discriminator)	1.0×10^{-4} (for DrQ-v2)
	1.0×10^{-5} (otherwise)
Learning rate (inverse dynamics)	1.0×10^{-6}

C. Performance in original domains

Table VI presents the average reward for the baseline SAC, SVEA, and DrQ-v2 on the non-distracted source domains. The method labeled SAC+Inv denotes a soft actor-critic agent that is trained using inverse dynamics as an additional auxiliary objective that is only used in making a comparison against PAD.

TABLE VI: Performance in the source (clean) domains.

Domain	SAC	SAC+Inv	SVEA	DrQ-v2
`ball_in_cup-catch`	$452^{\pm 303}$	$999^{\pm 7}$	$1007^{\pm 4}$	$1007^{\pm 3}$
`cartpole-balance`	$1022^{\pm 8}$	$988^{\pm 26}$	$996^{\pm 21}$	$969^{\pm 123}$
`cartpole-swingup`	$735^{\pm 167}$	$885^{\pm 24}$	$892^{\pm 15}$	$874^{\pm 21}$
`cheetah-run`	$309^{\pm 26}$	$415^{\pm 59}$	$448^{\pm 105}$	$897^{\pm 45}$
`finger-spin`	$615^{\pm 63}$	$971^{\pm 64}$	$1000^{\pm 37}$	$997^{\pm 36}$
`finger-turn_easy`	$138^{\pm 28}$	$658^{\pm 141}$	$539^{\pm 317}$	$945^{\pm 46}$
`reacher-easy`	$381^{\pm 39}$	$723^{\pm 383}$	$812^{\pm 293}$	$988^{\pm 28}$
`walker-stand`	$438^{\pm 111}$	$997^{\pm 6}$	$1006^{\pm 4}$	$996^{\pm 31}$
`walker-walk`	$393^{\pm 117}$	$915^{\pm 22}$	$964^{\pm 34}$	$980^{\pm 16}$

D. Ablation Studies

Tables VII, VIII, and IX provide a per-domain ablation summary for background, color, and camera pose distractions, respectively. As with the results in Table II, we use DrQ-v2 as the pretrained policy and present the mean reward and standard deviation for five random seeds.

TABLE VII: Ablation with background distraction

Domain	Zero-shot	+ILA	+ILA w/o inv., fwd.	+ILA w/o inv.	+ILA w/o fwd.	+ILA w/o adv.
`walker-walk`	$326^{\pm 196}$	$749^{\pm 133}$	$778^{\pm 142}$	$777^{\pm 145}$	$768^{\pm 146}$	$268^{\pm 367}$
`walker-stand`	$623^{\pm 233}$	$866^{\pm 153}$	$883^{\pm 110}$	$859^{\pm 144}$	$873^{\pm 129}$	$402^{\pm 347}$
`cartpole-swingup`	$82^{\pm 35}$	$231^{\pm 128}$	$395^{\pm 218}$	$288^{\pm 163}$	$246^{\pm 152}$	$117^{\pm 43}$
`ball_in_cup-catch`	$88^{\pm 39}$	$394^{\pm 387}$	$381^{\pm 416}$	$400^{\pm 414}$	$383^{\pm 381}$	$93^{\pm 42}$
`finger-spin`	$208^{\pm 327}$	$783^{\pm 214}$	$742^{\pm 190}$	$772^{\pm 225}$	$769^{\pm 223}$	$74^{\pm 160}$
`reacher-easy`	$98^{\pm 93}$	$726^{\pm 149}$	$713^{\pm 111}$	$732^{\pm 104}$	$694^{\pm 169}$	$91^{\pm 54}$
`cheetah-run`	$98^{\pm 90}$	$411^{\pm 191}$	$397^{\pm 152}$	$398^{\pm 147}$	$419^{\pm 157}$	$12^{\pm 19}$
`cartpole-balance`	$271^{\pm 101}$	$336^{\pm 126}$	$315^{\pm 98}$	$367^{\pm 84}$	$297^{\pm 121}$	$264^{\pm 80}$
`finger-turn_easy`	$261^{\pm 244}$	$920^{\pm 41}$	$929^{\pm 57}$	$899^{\pm 52}$	$909^{\pm 74}$	$256^{\pm 264}$

TABLE VIII: Ablation with color distraction

Domain	Zero-shot	+ILA	+ILA w/o inv., fwd.	+ILA w/o inv.	+ILA w/o fwd.	+ILA w/o adv.
`walker-walk`	$80^{\pm 43}$	$481^{\pm 313}$	$468^{\pm 305}$	$495^{\pm 330}$	$472^{\pm 319}$	$26^{\pm 5}$
`walker-stand`	$278^{\pm 150}$	$543^{\pm 231}$	$603^{\pm 283}$	$571^{\pm 259}$	$548^{\pm 268}$	$145^{\pm 31}$
`cartpole-swingup`	$152^{\pm 84}$	$552^{\pm 377}$	$520^{\pm 359}$	$507^{\pm 339}$	$434^{\pm 395}$	$94^{\pm 56}$
`ball_in_cup-catch`	$239^{\pm 374}$	$812^{\pm 225}$	$857^{\pm 182}$	$840^{\pm 251}$	$843^{\pm 196}$	$131^{\pm 46}$
`finger-spin`	$349^{\pm 354}$	$612^{\pm 345}$	$561^{\pm 376}$	$591^{\pm 351}$	$603^{\pm 358}$	$143^{\pm 172}$
`reacher-easy`	$138^{\pm 135}$	$490^{\pm 416}$	$486^{\pm 392}$	$486^{\pm 362}$	$445^{\pm 405}$	$140^{\pm 90}$
`cheetah-run`	$193^{\pm 188}$	$422^{\pm 282}$	$416^{\pm 293}$	$421^{\pm 277}$	$406^{\pm 285}$	$4^{\pm 2}$
`cartpole-balance`	$481^{\pm 351}$	$602^{\pm 366}$	$576^{\pm 359}$	$593^{\pm 351}$	$583^{\pm 349}$	$216^{\pm 94}$
`finger-turn_easy`	$194^{\pm 200}$	$313^{\pm 290}$	$323^{\pm 337}$	$297^{\pm 316}$	$304^{\pm 323}$	$170^{\pm 57}$

TABLE IX: Ablation with camera pose distraction

Domain	Zero-shot	+ILA	+ILA w/o inv., fwd.	+ILA w/o inv.	+ILA w/o fwd.	+ILA w/o adv.
walker-walk	$293^{\pm 195}$	$375^{\pm 154}$	$369^{\pm 168}$	$389^{\pm 161}$	$366^{\pm 164}$	$63^{\pm 71}$
walker-stand	$621^{\pm 202}$	$704^{\pm 105}$	$617^{\pm 155}$	$661^{\pm 146}$	$679^{\pm 93}$	$330^{\pm 252}$
cartpole-swingup	$286^{\pm 51}$	$234^{\pm 123}$	$211^{\pm 122}$	$241^{\pm 140}$	$281^{\pm 42}$	$172^{\pm 132}$
ball_in_cup-catch	$327^{\pm 227}$	$462^{\pm 307}$	$429^{\pm 321}$	$566^{\pm 314}$	$400^{\pm 337}$	$199^{\pm 169}$
finger-spin	$29^{\pm 25}$	$252^{\pm 216}$	$222^{\pm 209}$	$275^{\pm 194}$	$251^{\pm 206}$	$32^{\pm 63}$
reacher-easy	$917^{\pm 101}$	$933^{\pm 47}$	$925^{\pm 60}$	$950^{\pm 72}$	$940^{\pm 90}$	$732^{\pm 159}$
cheetah-run	$55^{\pm 20}$	$142^{\pm 57}$	$154^{\pm 90}$	$141^{\pm 74}$	$144^{\pm 61}$	$24^{\pm 28}$
cartpole-balance	$288^{\pm 46}$	$307^{\pm 100}$	$375^{\pm 83}$	$379^{\pm 116}$	$380^{\pm 64}$	$217^{\pm 47}$
finger-turn_easy	$287^{\pm 89}$	$346^{\pm 219}$	$359^{\pm 122}$	$377^{\pm 200}$	$421^{\pm 180}$	$160^{\pm 97}$