Unsupervised Visual Attention and Invariance for Reinforcement Learning

Xudong Wang*   Long Lian*   Stella X. Yu
UC Berkeley / ICSI

{xdwang,longlian,stellayu}@berkeley.edu

Abstract

Vision-based reinforcement learning (RL) is successful, but how to generalize it to unknown test environments remains challenging. Existing methods focus on training an RL policy that is universal to changing visual domains, whereas we focus on extracting visual foreground that is universal, feeding clean invariant vision to the RL policy learner. Our method is completely unsupervised, without manual annotations or access to environment internals.

Given videos of actions in a training environment, we learn how to extract foregrounds with unsupervised keypoint detection, followed by unsupervised visual attention to automatically generate a foreground mask per video frame. We can then introduce artificial distractors and train a model to reconstruct the clean foreground mask from noisy observations. Only this learned model is needed during test to provide distraction-free visual input to the RL policy learner.

Our Visual Attention and Invariance (VAI) method significantly outperforms the state-of-the-art on visual domain generalization, gaining 15~49% (61~229%) more cumulative rewards per episode on DeepMind Control (our DrawerWorld Manipulation) benchmarks. Our results demonstrate that it is not only possible to learn domain-invariant vision without any supervision, but freeing RL from visual distractions also makes the policy more focused and thus far better.

1. Introduction

Vision-based deep reinforcement learning (RL) has achieved considerable success on robot control and manipulation. Visual inputs provide rich information that are easy and cheap to obtain with cameras [31, 32, 27, 8, 7]. However, vision-based RL remains challenging: It not only needs to process high-dimensional visual inputs, but it is also required to deal with significant variations in new test scenarios (Fig. 1), e.g. color/texture changes or moving distractors [34, 1].

One solution is to learn an ensemble of policies, each handling one type of variations [46]. However, anticipating
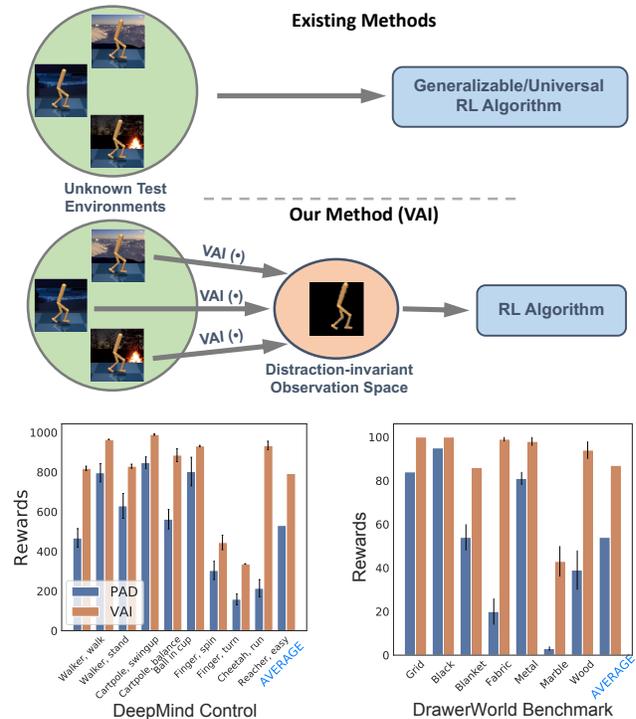
---
*Equal contribution.

Figure 1: **Top)** Two ways to make vision-based reinforcement learning generalizable to unknown environments at the test time: Existing methods focus on learning an RL policy that is universal to varying domains, whereas our proposed Visual Attention and Invariance (VAI) extracts visual foreground that is universal, feeding clean and invariant vision to RL. **Bottom)** VAI significantly outperforms PAD (SOTA), increasing cumulative rewards by 49% and 61% respectively in random color tests on DeepMind control and random texture tests on our DrawWorld manipulation benchmarks.

all possible variations quickly becomes infeasible; domain randomization methods [50, 55, 44, 45, 59] apply augmentations in a simulated environment and train a *domain-agnostic universal policy* conditioned on estimated discrepancies between testing and training scenarios.

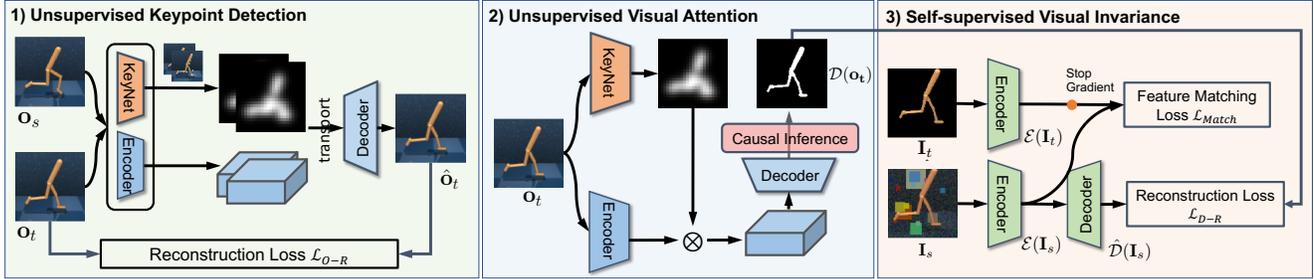Two caveats limit the appeal of a universal RL policy. **1)**

Figure 2: Our VAI method has three components. **1) Unsupervised keypoint detection**: Given two adjacent video frames, we learn to predict keypoints and visual features from each image so that foreground features and target keypoints can be used to reconstruct the target frame, without any manual annotations. **2) Unsupervised visual attention**: We apply causal inference to remove the model bias from the foreground mask derived from detected keypoints. **3) Self-supervised visual invariance**: We are then able to add artificial distractors and train a model to reconstruct the clean foreground observations. Keypoint and attention modules are only used during training to extract foregrounds from videos without supervision, whereas only the last encoder/decoder (colored in green) trained for visual invariance is used to remove distractors automatically at the test time.

Model complexity. The policy learner must have enough complexity to fit a large variety of environments. While there are universal visual recognition and object detection models that can adapt to multiple domains [47, 56], it would be hard to accomplish the same with a RL policy network often containing only a few convolutional layers. **2) Training instability.** RL training could be brittle, as gradients for (often non-differentiable) dynamic environments can only be approximated with a high variance through sampling. Adding strong augmentations adds variance and further instability, causing inability to converge. [13] handles instability with weaker augmentations, in turn reducing generalization.

The state-of-the-art (SOTA) approach, PAD [13], performs unsupervised policy adaption with a test-time auxiliary task (e.g. inverse dynamic prediction) to fine-tune the visual encoder of the policy network on the fly. However, there is no guarantee that intermediate representations would fit the control part of the policy network. Drastic environment changes such as background texture change from *grid* to *marble* can cause feature mismatches between adapted layers and frozen layers, resulting in high failure rates.

Instead of pursuing a policy that is universal to changing visual domains, we propose to extract visual foreground that is universal, and then feed clean invariant vision to a standard RL policy learner (Fig. 2). As the visual observation varies little between training and testing, the RL policy can be simplified and focused, delivering far better results.

Our technical challenge is to deliver such clean visuals with a completely unsupervised learning approach, without mannual annotations or access to environment internals.

Given videos of actions in a training environment, we first learn how to extract visual foreground with unsupervised keypoint detection followed by unsupervised visual attention to automatically generate a foreground mask per video frame. We can then introduce artificial distractors

and train a model to reconstruct the clean foreground mask from noisy observations. Only this learned model, not the keypoint or attention model, is needed during test to provide distraction-free visual input to the RL policy learner.

Our unsupervised Visual Attention and Invariance (VAI) method has several desirable properties.

1. **Unsupervised task-agnostic visual adaption training.** Our foreground extraction only assumes little background change between adjacent video frames, requiring no manual annotations or knowledge of environment internals (e.g. get samples with altered textures). It does not depend on the task, policy learning, or task-specific rewards associated with RL. That is, for different tasks in the same environment, we only need to collect one set of visual observations and train one visual adapter, which gets us a huge saving in real-world robotic applications.

2. **Stable policy training, no test-time adaptation.** By freeing RL from visual distractions, our policy learning is stable and fast without being subject to strong domain augmentations, and our policy deployment is immediate without test-time fine-tuning.

3. **Clear interpretation and modularization.** We extract keypoints from videos to identify foreground, based on which attentional masks can be formed. This unsupervised foreground parsing allows us to anticipate visual distractions and train a model to restore clean foregrounds. Compared to existing methods that work on intermediate features, our method has clear assumptions at each step, which can be visualized, analyzed, and improved.

We conduct experiments on two challenging benchmarks with diverse simulation environments: DeepMind Control suite [53, 13] and our DrawerWorld robotic manipulation tasks with texture distortions and background distractions during deployment. Our VAI significantly outperforms the state-of-the-art, gaining 15~49% (61~229%) more cumula-

tive rewards per episode on DeepMind Control (our Drawer-World Manipulation) benchmarks.

To summarize, we make the following contributions.

1. We propose a novel domain generalization approach for vision-based RL: Instead of learning a universal policy for varying visual domains, we decouple vision and action, learning to extract universal visual foreground while keeping the RL policy learning intact.

2. We propose a fully unsupervised, task-agnostic visual adaptation method that removes unseen distractions and restores clean foreground visuals. Without manual annotations, strong domain augmentations, or test-time adaptation, our policy training is stable and fast, and our policy deployment is immediate without any latency.

3. We build unsupervised keypoint detection based on KeyNet [21] and Transporter [24]. We develop a novel unsupervised visual attention module with causal inference for counterfactual removal. We achieve visual invariance by unsupervised distraction adaptation based on foreground extraction. Each step is modularized and has clear interpretations and visualizations.

4. We propose DrawerWorld, a pixel-based robotic manipulation benchmark, to test the adaptation capability of vision-based RL to various realistic textures.

5. Our results demonstrate that it is not only possible to learn domain-invariant vision from videos without supervision, but freeing RL from visual distractions also leads to better policies, setting new SOTA by a large margin.

2. Related Works

Unsupervised Learning has made much progress in natural language processing, computer vision, and RL. It aims to learn a feature transferable to downstream tasks [6, 37, 25, 5, 58, 15, 4, 3, 57]. In RL, UNREAL [20] proposes unsupervised reinforcement and auxiliary learning to improve learning efficiency of model-free RL algorithms, by maximizing pseudo-reward functions; CPC [38] learns representations for RL in 3D environments by predicting the future in the latent space with autoregressive models; CURL [51] extracts high-level features from raw pixels using contrastive learning and performs off-policy control on extracted features to improve data-efficiency on pixel-based RL.

Domain Adaptation incorporates an adaptation module to align the feature distribution from the source domain and the target domain without paired data [39, 28, 49, 30, 63, 10]. There are various approaches to this, from using supervised data [?, 28, 49], to assumed correspondences [?], to unsupervised approaches [?, ?, 63].

Multi-domain Learning learns representations for multiple domains known a prior [23, 35, 47, 56]. A combination of shared and domain-specific parameters are adopted. It is also feasible to simply learn multiple visual domains with residual domain adapters [47, 56].

Our work is different from these works, since we do not have prior knowledge of test data distributions and the model needs to generalize to unknown test environments.

Robustness to Distribution Shifts studies the effect of corruptions, perturbations, out-of-distribution examples, and real-world distribution shifts [36, 29, 18, 33, 45, 17]. Recent deep RL approaches model such uncertainties explicitly. [16] uses recurrent neural networks for direct adaptive control and determines dynamic model parameters on-the-fly. UP-OSI [61] applies indirect adaptive control for online parameter identification. EPOpt [46] uses simulated source domains and adversarial training to learn policies that are robust and generalizable to a range of possible target domains. PAD [13] uses self-supervision to continue policy training during deployment without any rewards, achieving SOTA in several environments. SODA [14], a concurrent work to ours, alternates strong augmentations associated with self-supervised learning and weak augmentations associated with RL for obtaining both generalizability and stability.

Instead of demanding a universal *policy* that is invariant to distribution shifts or transferable to novel environments, we achieve generalizability by demanding universal *visuals* that can be fed into the subsequent RL policy learner, freeing it from visual distractions and making it more effective.

3. Unsupervised Visual Attention & Invariance

Our goal is to extract universal visual foreground and then feed clean invariant vision to an RL policy learner (Fig. 2). Our technical challenge is to deliver such clean visuals with a completely unsupervised learning approach, without manual annotations or access to environment internals.

Our VAI method has three components: Unsupervised keypoint detection, unsupervised visual attention, and self-supervised visual invariance. The first two are only used during training to extract foregrounds from videos without supervision, whereas only the last trained model is deployed to automatically remove distractors from a test video.

3.1. Unsupervised Keypoint Detection

We assume that training videos contain moving foregrounds against a relatively still background. Our idea for unsupervised foreground extraction is the following: Given two such source and target frames, we can learn to predict keypoints and visual features from each image so that foreground features and target keypoints can be used to reconstruct the target frame, without requiring manual annotations.

For a particular image pair, the moving foreground may have a still part (*upper body*), or the background may have a moving part (*flickering flames*) . However, when the keypoint predictor and the visual feature extractor have to work consistently across all the videos in the same environment, they would have to focus on the entire moving foreground and disregard the random minor background motion.
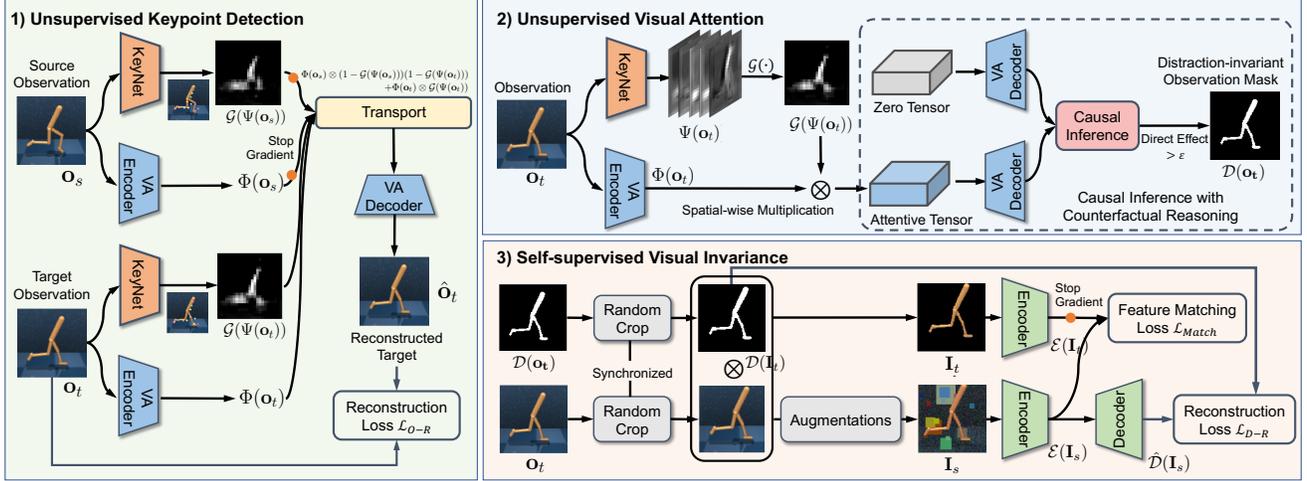
Figure 3: Technical implementation of our three components. **1) Unsupervised keypoint detection**: We build unsupervised keypoint detection and visual feature extraction based on KeyNet [21] and Transporter [24]. The goal is to reconstruct the target frame from the target foreground appearance and the source-transported background appearance, capturing a moving foreground on a relatively still background. **2) Unsupervised visual attention**: We remove the model bias in the foreground mask derived from detected keypoints with novel causal inference for counterfactual removal. **3) Self-supervised visual invariance**: We train a model to restore an invariant foreground visual image by adding artificial distractors to extracted foreground and perform self-supervised distraction removal.

Let $\mathbf{o}_s, \mathbf{o}_t \in \mathbb{R}^{C \times H \times W}$ denote the source and target frames sampled from a trajectory, where C, H, and W are the channel dimension, image height and width respectively. Let $\Phi(\cdot)$ denote the visual feature extractor. Let $\Psi(\cdot)$ denote the keypoint network that predicts K keypoints in terms of 2D spatial locations $\{\mu_k\}$. We render each keypoint as a smaller $H' \times W'$ Gaussian heatmap with fixed variance σ^2, and derive a foreground mask by taking the max of all of them:

$$\mathcal{G}(\mu; x) = \max_{k \in \{1,2,...,K\}} \exp\left(-\frac{\|x - \mu_k\|^2}{2\sigma^2}\right). \quad (1)$$

We follow KeyNet [21, 24] to reconstruct the target observation $\mathbf{o}_t$ from K landmarks $\Psi(\mathbf{o}_s), \Psi(\mathbf{o}_t)$. We follow [24] to transport the source background appearance to the target frame by putting the source feature at common background areas and the target feature at the target keypoints:

$$\hat{\Phi}(\mathbf{o}_t, \mathbf{o}_s) = \Phi(\mathbf{o}_s) \otimes (1 - \mathcal{G}(\Psi(\mathbf{o}_s)))(1 - \mathcal{G}(\Psi(\mathbf{o}_t))) \\ + \Phi(\mathbf{o}_t) \otimes \mathcal{G}(\Psi(\mathbf{o}_t)) \quad (2)$$

where $\otimes$ denotes location-wise multiplication applied to each channel. A visual attention (VA) decoder outputs a reconstruction $\hat{\mathbf{o}}_t$ of target frame $\mathbf{o}_t$ from the transported feature $\hat{\Phi}(\mathbf{o}_t, \mathbf{o}_s)$. Minimizing the reconstruction loss below optimizes the KeyNet and the visual attention encoder/decoder end-to-end:

$$\mathcal{L}_{\text{O-R}}(\mathbf{o}_t, \hat{\mathbf{o}}_t) = \|\mathbf{o}_t - \hat{\mathbf{o}}_t\|_2^2. \quad (3)$$

Note that the original Transporter only focuses on changes between frames in the same episode, whereas we also sample frames from different episodes 50% of the time in reacher environment which has a fixed target throughout each episode, so that our keypoints will be able to capture the target and spread over the entire moving foreground.

3.2. Unsupervised Spatial Attention

Now we already have an unsupervisedly learned keypoint detector. We first explain why we do not use keypoints for control and instead derive a visual foreground mask. We then describe our novel causal inference formulation for obtaining a foreground mask without model bias.

Transporter [24] successfully makes use of keypoints for RL in Atari ALE [2] and Manipulator [53]. Keypoints are geometrical extraction without visual appearance distractions that they could be potentially used to minimize differences between training and testing environments.

However, there are three major issues with keypoints in practice. **1)** It is often hard to track keypoints consistently across frames; even for humans, whether a keypoint is on the left or right foot is unclear in Fig. 4. This implies that using predicted keypoints for control directly would be brittle even in clean images. **2)** While keypoints along with image features and LSTM could work on relatively complicated tasks [24], they add substantial model complexity and computational costs. **3)** While keypoints themselves are free of visual distractions,
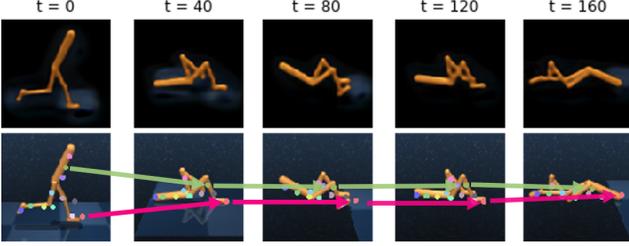
Figure 4: Our VAI foreground reconstruction (Row 1) provides clearer and more robust foreground visual information than detecting keypoints across image frames using Transporter (Row 2). Due to occlusion, symmetry, and lacking visual distinctions, it is often impossible to track keypoints consistently across frames. That is, keypoint locations alone are not suitable as an invariant visual representation.

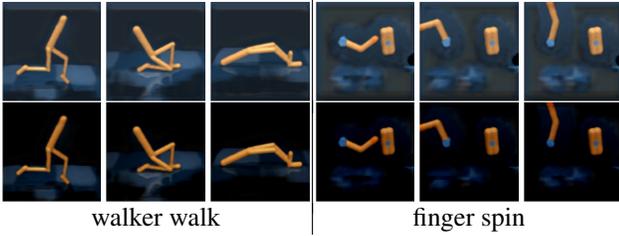walker walk   finger spin

Figure 5: Foreground reconstructions with causal inference are cleaner (Row 2) than those without (Row 1).
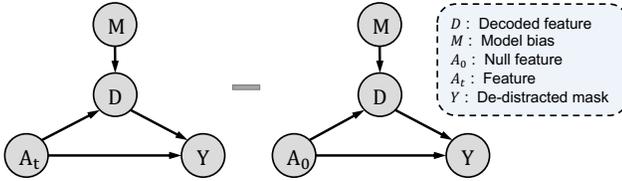

Figure 6: Causal graph of casual inference with counterfactual reasoning for our foreground mask extraction. The Controlled Direct Effect (CDE) is measured by the contrast between two outcomes: the counterfactual outcome given the visual feature A_t and that given the null feature A_0.

their extractor (*KeyNet*) is only trained for the training environment, with no guarantee for robustness against domain shifts.

We thus propose to generate a foreground mask $\mathcal{G}(\Psi(\mathbf{o}_t))$ from the (un-ordered) collection of predicted keypoints instead. We enhance the visual feature in the foreground: $\mathcal{G}(\Psi(\mathbf{o}_t)) \otimes \Phi(\mathbf{o}_t)$ and pass it to the VA decoder to reconstruct a cleaner image $\hat{\mathbf{o}}_t$ (Fig. 5 Row 1). However, it is blurry with common background remnants captured in the bias terms of the decoder. The bias terms are essential for proper reconstruction and cannot be simply set to zero.

We apply causal inference with counterfactual reasoning [41, 48, 43, 11, 52] to remove the model bias (Fig. 6). Intu-
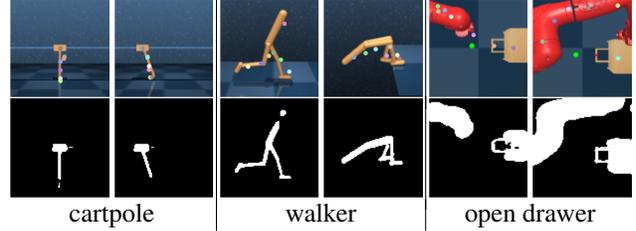

Figure 7: Our detected keypoints (Row 1) and generated foreground masks (Row 2) from DeepMind control and DrawerWorld benchmarks. Note that they could cover multiple moving objects in the foreground.

itively, the predicted foreground mask Y has a direct cause from the visual feature A, and an indirect cause from the model bias M through the decoder D. To pursue the direct causal effect, we perform counterfactual reasoning known as *Controlled Direct Effect* (CDE) [41, 11], which contrasts the counterfactual outcomes (marked by $do(\cdot)$) between visual feature A_t and null visual feature A_0 (set to the zero tensor):

$$\text{CDE}(Y) = [Y|do(A_t), do(M)] - [Y|do(A_0), do(M)]. \quad (4)$$

We further threshold it to obtain the foreground mask $\mathcal{D}(\mathbf{o_t})$:

$$\mathcal{D}(\mathbf{o_t}) = \begin{cases} 0, \text{CDE}(Y) < \epsilon \\ 1, \text{CDE}(Y) \geq \epsilon \end{cases} . \quad (5)$$

Fig. 7 shows our detected keypoints and generated masks: **1)** While the keypoints may be sparse and imprecise, the foreground mask is clean and complete; **2)** Our unsupervisedly learned keypoints do not correspond to semantic joints of articulation, e.g., for the grasper opening a drawer, there are keypoints on both the grasper and the drawer, and our derived foreground mask contains both moving objects.

3.3. Self-supervised Visual Invariance

Our spatial attention module outputs a foreground mask, after seeing samples in the training environment. To make it adaptable to unknown test environments, we augment the clean foreground image with artificial distractors and train a model to reconstruct a mask to retrieve clean foreground observation.

Given image $\mathbf{o}_t$, we generate equally cropped clean target image I_t and noisy source image I_s.

$$I_t = \mathcal{T}_c(\mathbf{o}_t) \otimes \mathcal{T}_c(\mathcal{D}(\mathbf{o}_t)) \quad (6)$$
$$I_s = \mathcal{T}_f(I_t) + \mathcal{T}_b(\mathcal{T}_c(\mathbf{o}_t) \otimes (1 - \mathcal{D}(I_t))) \quad (7)$$

where $\mathcal{D}(I_t) = \mathcal{T}_c(\mathcal{D}(\mathbf{o}_t))$, $\mathcal{T}_c$ denotes synchronized random crop, $\mathcal{T}_f$ adds possible foreground changes such as color jitter and random brightness change, whereas $\mathcal{T}_b$ adds a set of possible background changes such as random colored boxes to the background. We learn a convolutional
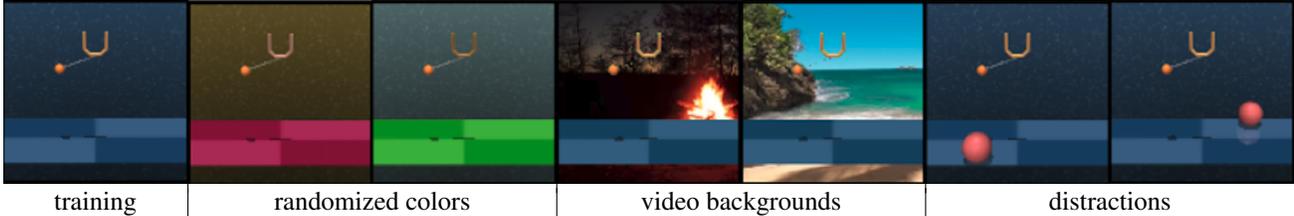
Figure 8: Visualization results of testing environments in DeepMind Control benchmark [13, 53]. The testing environment changes include randomized colors, video backgrounds, and background distractions.

encoder/decoder pair to reconstruct the clean foreground mask from noisy I_s, so that they could focus more on the foreground and ignore background distracters. We impose a feature matching loss at output of encoder $\mathcal{E}$ and an image reconstruction loss at output of decoder $\hat{\mathcal{D}}$:

$$\mathcal{L}_{\text{total}} = \|\hat{\mathcal{D}}(\mathbf{I}_s) - \mathcal{D}(\mathbf{I}_t)\|_2^2 + \lambda \cdot \|\mathcal{E}(\mathbf{I}_s) - \mathcal{E}(\mathbf{I}_t)\|_2^2 \quad (8)$$

where $\mathcal{D}(\mathbf{I}_t)$ is simply the cropped version of $\mathcal{D}(\mathbf{o}_t)$. During RL training and deployment, for any frame I, we feed $I \otimes \hat{\mathcal{D}}(\mathcal{E}(I))$ to the learned RL policy.

What augmentations to use has a big impact on generalization. We propose four additional strong background augmentations on $\mathcal{T}_b$. **1)** The background could randomly assume the training image background, a random color, or the mean foreground color with small perturbations. **2)** Gaussian pixel-wise noise and random boxes are added. Multi-ColorOut, an extension to Cutout-color [26], adds multiple boxes of random sizes, colors, and positions. **3)** Darkened foreground copies are added to the background areas where the foreground mask values are 0, to simulate distractors that look similar to the foreground. **4)** We follow [14] to randomly select images in the Places dataset [62] as background images for augmentation. For fair comparisons, we list our results with and without this option. With such generic augmentations, our model is able to perform well on realistic textures and unknown testing environments even though it has not encountered them during training.

RL policy training with weak augmentations. Our visual invariance model outputs a clean foreground image with background distracters suppressed. The RL policy learner still needs to handle foreground variations in unknown test environments. We train our RL policy with weak foreground augmentations to make it robust to noise and distortions. We add the usual Gaussian random noise and use only a simple MultiColorOut to simulate the inclusion of backgrounds and missing foreground parts. Empirically we find that such weak augmentations do not affect the RL training stability.

4. Experiments

We experiment on two benchmarks, DeepMind and DrawerWorld, and perform ablation studies. The DeepMind Control benchmark contains various background distractions

Random colors	SAC	DR	PAD	SODA+P	VAI	VAI+P	Δ
Walker, walk	414 ±74	594 ±104	468 ±47	692 ±68	819 ±11	**918** **±6**	**+226** (↑ 33%)
Walker, stand	719 ±74	715 ±96	797 ±46	893 ±12	964 ±2	**968** **±3**	**+75** (↑ 8%)
Cartpole, swingup	592 ±50	647 ±48	630 ±63	805 ±28	**830** **±10**	819 ±6	**+14** (↑ 2%)
Cartpole, balance	857 ±60	867 ±37	848 ±29	-	**990** **±4**	957 ±9	**+142** (↑ 17%)
Ball in cup, catch	411 ±183	470 ±252	563 ±50	949 ±19	886 ±33	**960** **±8**	**+11** (↑ 1%)
Finger, spin	626 ±163	465 ±314	803 ±72	793 ±128	932 ±3	**968** **±6**	**+165** (↑ 21%)
Finger, turn_easy	270 ±43	167 ±26	304 ±46	-	**445** **±36**	455 ±48	**+151** (↑ 50%)
Cheetah, run	154 ±41	145 ±29	159 ±28	-	**337** **±1**	334 ±2	**+178** (↑ 112%)
Reacher, easy	163 ±45	105 ±37	214 ±44	-	**934** **±22**	936 ±19	**+722** (↑ 337%)
average	467	464	531	-	793	812	**+281** (↑ 53%)

Table 1: VAI outperforms existing methods on DeepMind randomized color tests by a large margin *without* using the external Places dataset; it is even better than SODA+P, which uses Places as a part of the training set. Soft Actor-Critic (SAC) [12, 32] is used as a base algorithm for DR (domain randomization), PAD [13], SODA [14], and our VAI. SODA+P and VAI+P use Places [62] as overlay or adapter augmentation. The results of SAC and DR are copied from PAD [13]. Listed are the mean and std of cumulative rewards across 10 random seeds and 100 random episode initializations per seed. The absolute and relative improvement of VAI over SOTA method are listed in the Δ column.

[54, 13] as in Fig. 8. We propose a DrawerWorld Robotic Manipulation benchmark, based on MetaWorld [60], in order to test a model's texture adaptability in manipulation tasks.

4.1. DeepMind Control Benchmark

Tasks. There are walking, standing, and reaching objects [54], all in 3D simulation. Our agent receives pixel-based inputs instead of state-based inputs from the underlying dynamics unless otherwise stated.

Testing. We follow PAD [13] and test our method under three types of environments: 1) randomized colors; 2) video backgrounds; and 3) distracting objects. For tasks with video background and distracting objects, we apply a moving average de-noising trick by subtracting a moving average

Video background	SAC	DR	PAD	SODA	SODA+P	VAI	VAI+P	Δ
Walker, walk	616 ±80	655 ±55	717 ±79	635 ±48	768 ±38	870 ±21	**917** ±8	**+149** (↑ 19%)
Walker, stand	899 ±53	869 ±60	935 ±20	903 ±56	955 ±13	**966** ±4	968 ±2	**+13** (↑ 1%)
Cartpole, swingup	375 ±90	485 ±67	521 ±76	474 ±143	**758** ±62	624 ±146	761 ±127	**+3** (↑ 0%)
Cartpole, balance	693 ±109	766 ±92	687 ±58	-	-	**869** ±189	847 ±205	**+182** (↑ 26%)
Ball in cup, catch	393 ±175	271 ±189	436 ±55	539 ±111	**875** ±56	790 ±249	846 ±229	**-29** (↓ 3%)
Finger, spin	447 ±102	338 ±207	691 ±80	363 ±185	695 ±97	569 ±366	**953** ±28	**+258** (↑ 37%)
Finger, turn_easy	355 ±108	223 ±91	362 ±101	-	-	419 ±50	**442** ±33	**+80** (↑ 22%)
Cheetah, run	194 ±30	150 ±34	206 ±34	-	-	322 ±35	**325** ±31	**+119** (↑ 58%)
average	497	470	569	-	-	678	757	**+188** (↑ 33%)

Table 2: VAI+P (VAI) outperforms PAD by more than 33% (19%) on challenging DeepMind video backgrounds. Same settings and conventions as Table 1.

Distracting objects	SAC	DR	PAD	VAI	Δ
Cartpole, swingup	815 ±60	809 ±24	771 ±64	**891** ±0	**+120** (↑ 16%)
Cartpole, balance	969 ±20	938 ±35	960 ±29	**993** ±0	**+24** (↑ 2%)
Ball in cup, catch	177 ±111	331 ±189	545 ±173	**956** ±4	**+411** (↑ 75%)
Finger, spin	652 ±184	564 ±288	**867** ±72	805 ±3	**-62** (↓ 7%)
Finger, turn_easy	302 ±68	165 ±12	347 ±48	**389** ±18	**+42** (↑ 12%)
average	583	561	698	806	**+108** (↑ 15%)

Table 3: VAI outperforms current SOTAs by more than 15% on DeepMind Control distracting objects. Although VAI performs worse than PAD on "Finger, spin" task in terms of mean rewards, the reward variance is greatly reduced from 72 to 3 in std. Same settings and conventions as Table 1.

success %	DrawerOpen				DrawerClose			
	SAC	PAD	VAI	Δ	SAC	PAD	VAI	Δ
Grid	98 ±2	84 ±7	**100** ±0	**+2** (↑ 2%)	**100** ±0	95 ±3	99 ±1	**-1** (↓ 1%)
Black	95 ±2	95 ±3	**100** ±1	**+5** (↑ 5%)	75 ±4	64 ±9	**100** ±0	**+25** (↑ 33%)
Blanket	28 ±8	54 ±6	**86** ±6	**+32** (↑ 59%)	0 ±0	0 ±0	**85** ±8	**+85** (↑ ∞%)
Fabric	2 ±1	20 ±6	**99** ±1	**+79** (↑ 395%)	0 ±0	0 ±0	**74** ±8	**+74** (↑ ∞%)
Metal	35 ±7	81 ±3	**98** ±2	**+17** (↑ 21%)	0 ±0	2 ±2	**98** ±3	**+96** (↑ 4800%)
Marble	3 ±1	3 ±1	**43** ±7	**+40** (↑ 1333%)	0 ±0	0 ±0	**49** ±13	**+49** (↑ ∞%)
Wood	18 ±5	39 ±9	**94** ±4	**+55** (↑ 141%)	0 ±0	12 ±2	**70** ±6	**+58** (↑ 483%)
average	40	54	87	**+33** (↑ 61%)	25	25	82	**+57** (↑ 228%)

Table 4: Our VAI consistently outperforms all the baselines in new texture environments, and on DrawerClose in particular, VAI succeeds 82% vs. SAC/PAD's 25%. Grid is the training environment. Black means a completely dark background without texture. Other textures are shown in Fig. 9. DrawerClose is more challenging than DrawerOpen, as the drawer handle is concealed by the effector in DrawerClose, which would require the agent to infer the handle position from the position and the size of the effector. The success rate is the percentage of successful attempts out of 100 attempts to open or close a drawer. The mean/std are collected over 10 seeds.

of the past observations from the current observation and adding back the mean color of the moving average. We introduced a constant factor α multiplied to past moving average to tune the aggressiveness of the de-noising trick.

Training. For each scenario, we train agents without distractions and evaluate the model across 10 random seeds and 100 random environment initializations. To get observation samples for training, we export 5000 transitions from the replay buffer for the training environment, which are collected with a random policy. We use the same environment settings such as frame skip and data augmentation as in PAD to ensure fair comparisons between VAI, PAD, and others.

Randomized color results. Table 1 shows that our VAI outperforms published SOTA on all the 9 tasks by up-to an astonishing 337% margin in terms of mean cumulative rewards, without seeing samples in the test environment at any time. In contrast, DR is trained with color change to the environment (which requires knowing and changing the internals of the environment), which, to some extent, previews what the test environment would be. Similarly,

although PAD does not use any evaluation samples during training, it does use the samples at the test time to tune the encoder. Since VAI does not change model weights, it has no adaptation delay, better stability, and less compute (see more details in supplementary materials). By suppressing distractions and feeding only the foreground image, the RL algorithm ideally sees the same input no matter what the environment is and is thus not influenced by background distractions or domain shifts in the test environment.

Video background results. Table 2 shows that our VAI outperforms baselines in 7 out of 8 tasks in terms of mean cumulative rewards, often by a large margin.

Distracting object results. Table 3 shows that our VAI surpasses baselines on 4 out of 5 tasks in terms of cumulative rewards. It not only obtains nearly full scores on "Cartpole, balance" and "Ball in cup, catch" tasks, but also greatly decreases the variance of results to a negligible level.

4.2. DrawerWorld Manipulation Benchmark

A New Texture Benchmark for Manipulation. CNNs are sensitive to textures [9]. We propose to evaluate a model's texture adaptability in manipulation tasks, based on the Meta-World [60] benchmark for meta RL and multi-task RL.

In the original MetaWorld, the observations include 3D Cartesian positions of the robot, the object, and the goal positions collected with sensors on the object and the robot. Accurate object positions and robot keypoints are hard to get
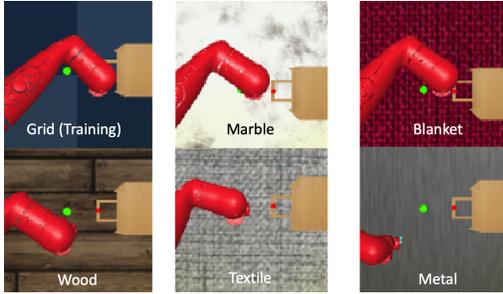
Figure 9: DrawerWorld environments. Grid is the texture used in training. Other 5 evaluation textures are from realistic photos, which makes the task challenging.

success rate (%)	Grid	Wood	Metal	Fabric
SAC	98 ±2	18 ±5	35 ±7	2 ±1
+ RL Augmentation	**100** **±1**	18 ±5	41 ±8	24 ±5
+ Foreground Extraction	**100** **±0**	18 ±4	13 ±4	38 ±4
+ Background Augmentation	**100** **±0**	**94** **±4**	**98** **±2**	**99** **±1**

Table 5: Ablation studies for augmentation and foreground extraction on DrawOpen task. From top to bottom rows, components are added to the method cumulatively. Each method is trained in the grid environment and tested in new texture enviroments of wood, metal, and fabric. Success rates are collected over 500K steps. Only the last method with all augmentations deliver consistent robustness.

by in real-world applications, we thus propose a variant of MetaWorld, *DrawerWorld*, with visual observations instead. We focus on the variety of *realistic textures* (Fig. 9).

Tasks. There are DrawerOpen and DrawerClose tasks, where a Sawyer arm is manipulated to open and close a drawer. The action space contains the end-effector positions in 3D. We adopt MetaWorld reward functions and success metrics. See supplementary materials for details.

Testing. We test the agent on surfaces of different textures which, unlike the grid texture used for training, come from photos instead of from simulations. These tasks are extremely challenging for two reasons: 1) The agent has never seen any realistic textures during training; 2) Each texture also has a different color, so the agent needs to handle both color change and texture change at the same time.

Texture background results. Table 4 shows that our VAI outperforms PAD [13] and SAC [12] significantly in all the test environments. In particular, for 5 out of 6 textures such as blanket, metal, and wood, SAC and PAD have 0% success rate, whereas VAI performs far better at 85%, 98%, and 70% respectively. In the training grid environment, PAD performs worse than SAC, consistent with [13] on the DeepMind Control benchmark, whereas our VAI is on-par or slightly better than SAC, suggesting that we are not gaining texture adaptability at the cost of losing training performance.

CNNs' sensitivity to textures poses a big challenge for visual adaptation. **1)** SAC adapts to unknown test environments with augmentations at the training time. Since textures are not used during training, SAC breaks down during texture testing. **2)** PAD has to change its feature encoder a lot in order to adapt to never-seen textures at the time time, shifting the feature distribution. However, PAD assumes an invariant feature distribution and, therefore, does not fine-tune the control part of the policy network at the test time, which causes the vision-RL pipeline to break down.

4.3. Ablation Studies

We evaluate four ablated variants of our methods on the DrawerOpen task:

1. SAC, a base universal policy learning model
2. Method 1 + RL with image augmentations, equivalent to Domain Randomization;
3. Method 2 + Visual invariance module trained without augmentations: $\mathcal{T}_f, \mathcal{T}_b$ are identity functions;
4. Method 3 + We apply the augmentations in Section 3.3 on $\mathcal{T}_f, \mathcal{T}_b$, for greater adaptability.

Table 5 shows that while all the methods perform well in the training environment, they adapt poorly to realistic textures except the last one. These results suggest that adding visual augmentations during RL or to the entire image as a whole is insufficient; providing a clean observation for RL agents with foreground clues adds significant robustness to vision-based RL.

5. Summary

We propose a fully unsupervised method to make vision-based RL more generalizable to unknown test environments. While existing methods focus on learning a universal policy, we focus on learning universal foreground vision.

We learn to extract foregrounds with unsupervised key-point detection, followed by unsupervised visual attention to remove model bias and generate a foreground mask. We then train a model to reconstruct the clean foreground mask from noise-augmented observations.

We propose an additional challenging DrawerWorld benchmark, which trains manipulation tasks on grid and tests on texture environments. Existing methods fail due to CNN's sensitivity to textures, yet our model with foreground extraction and strong generic augmentation is robust to never-seen textures without sacrificing training performance.

Our method significantly advances the state-of-the-art in vision-based RL, demonstrating that it is not only possible to learn domain-invariant vision without supervision, but freeing RL from visual distractions also improves the policy.

6. Additional Environment Descriptions

6.1. DeepMind Control

We wrote a short description for each environment in DeepMind Control suite [54] in Table 6 to further introduce the environment.

Environment	Descriptions
Walker	A planar walker which encourages an upright torso and minimal torso height in the "stand" task. In "walk" task forward velocity is also encouraged.
Cartpole	A pole tied to a cart at its base, with forces applied to the base. "swingup" task requires the pole to swing up from pointing down while "balance" task requires the pole to balance to be upright.
Ball in cup	A ball attached to a cup, with forces applied to the cup to swing the ball up into the cup in the "catch" task.
Finger	A finger is asked to rotate a rectangular body on a hinge. The top of the body needs to overlap with the object in "turn_easy" task and the body needs to rotate continuously in the "spin" task.
Cheetah	An animal with two feet which is asked to run in the "run" task.
Reacher	A planar reacher with two links connected with a hinge in a plane with a random target location. In the "easy" task, the reacher is asked to reach the object location. The "hard" task is unused in our evaluation since it was not adapted by [13].

Table 6: Descriptions for each environment in DeepMind Control suite.

We also provide samples for the evaluation environments designed by [13] in Fig. 10.

6.2. DrawerWorld

We propose the DrawerWorld, a benchmark with observations in pixels, based on MetaWorld [60] to enable the agent to work in an environment close to real-life scenarios. There are two tasks in DrawerWorld, which are DrawerOpen and DrawerClose. These tasks ask a Sawyer robot to open and close a drawer, respectively.

The multi-component reward function R is a combination of a reaching reward R_{reach} and a push reward R_{push} as follows:

$$R = R_{\text{reach}} + R_{\text{push}}$$
$$= -||h - p||_2 + \mathbb{I}_{||h-p||_2 < \epsilon} \cdot c_1 \cdot \exp\{||p - g||_2^2/c_2\} \quad (9)$$

where ϵ is a small distance threshold and is set as 0.08 by default, $p \in \mathbb{R}^3$ be the object position, $h \in \mathbb{R}^3$ be the position of the robot's gripper, and $g \in \mathbb{R}^3$ be goal position. $c_1 = 1000$ and $c_2 = 0.01$ for all tasks in DrawerWorld benchmark.

The goal of distraction-robust RL is to learn a task-conditioned policy $\pi(a|s, z)$, where z indicates an encoding of the task ID, and in this case, different task IDs have different drawer positions. This policy should maximize the average expected return from the task distribution $p(\mathcal{T})$, given

by $\mathbb{E}_{\mathcal{T} \sim p(\mathcal{T})}[\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t R_t(s_t, a_t)]]$. The success metric, which is evaluate the agent in evaluation time, is described by $\mathbb{I}_{||p-g||_2 < \epsilon}$, where ϵ is set to 8cm. The difference between training and evaluation time is the texture and color of the table cloth. Image samples of textures that we use are provided in the main text.

7. De-noise with Past Averages

Since our adapter model works on each frame separately without any assumption on temporal continuity of consecutive frames, our adapter works exactly the same on videos as on fixed backgrounds and is not affected by drastic changes in the background such as flashes of light. However, in some environments where the assumption of temporal continuity holds, i.e. with a relatively slow-moving background, we may make use of this assumption to better de-noise the observations before passing the them into the adapter.

We exploit the assumption here by keeping a mean of past observations $\mathbf{o}_{\text{mean}} = \frac{1}{t} \sum_{i=1}^{t} \mathbf{o}_i$ and subtract the mean from observation $\mathbf{o}_t$ and compute the observations after de-noise with the formulation: This de-noise step happens before the observation is sent into CNN (adaptor), formulated as:

$$\mathbf{o}_{\text{de-noised}} = \text{filter}(\mathbf{o}_t - \alpha\mathbf{o}_{\text{mean}}, \epsilon) + \alpha\mathbf{o}_{\text{mean\_color}} \quad (10)$$

where $\mathbf{o}_{\text{mean\_color}}$ is the mean of $\mathbf{o}_{\text{mean}}$ in spatial dimensions, filter is a function that sets the part with value less than ϵ to 0 to remove some noise, and $\alpha \in [0, 1]$ is the strength in noise removal.

This is completely optional, and since it makes use of an additional assumption, they are only used in the cartpole and ball in cup in experiments with background videos and experiments with distracting objects. In addition, we observe that with Places dataset as augmentation, the model is robust enough without this trick, so we disable it for all models in the training of which Places dataset is used.

8. Memory Usage and Speed Comparisons

From Fig. 11, it seems that VAI is about 3 times as fast as PAD in terms of the evaluation time in each episode and requires substantially less GPU memory than PAD. This is largely due to the fact that PAD trains the encoder network at evaluation time with back-propagation, which not only requires the intermediate results to be saved in GPU memory but also requires backward computation to update the model parameters, which consumes both time and memory space. Although VAI has an extra adapter module, the computation and memory it takes are much less than the ones required by backward computation and storing intermediate results. According to the requirements of computational resources in terms of speed and memory, our method is more suitable for robots powered by battery and edge inference devices than PAD from this point of view.
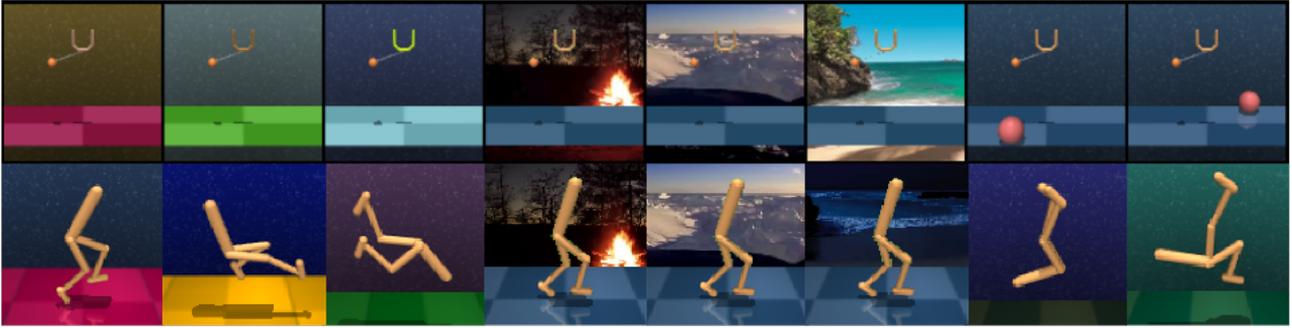
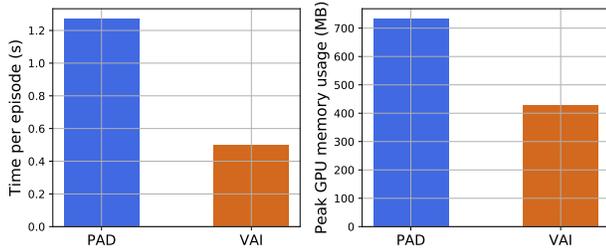Figure 10: Samples in evaluation environments in DeepMind Control. The samples in the first row are from [13].

Figure 11: Comparisons on the mean time per episode and GPU memory occupancy at evaluation time for *DrawerClose* task in DrawerWorld between current state-of-the-art method PAD [13] and the proposed method VAI. VAI is more than 2 times faster than PAD during testing time and requires ~40% less GPU memory usage. Both methods are evaluated with exactly the same backbone network. We take the mean of 10 runs for the latency comparison. Memory usage is obtained with `torch.cuda.max_memory_allocated`.

9. Further Experiments with Raw Keypoints

To investigate the question of whether raw keypoints extracted from image observations by KeyNet are able to contribute to effective learning of useful behaviors, we set up experiments based on the *Walker, walk* task in DeepMind Control and list the outcomes in Table 7.

We first train an agent with the state-based observation provided by the environment, a 24-dimensional tensor, which includes positions of the joint, velocity, and walker's torso height. We do not stack frames for this experiment. This experiment indicates an upper bound that our agent is able to achieve in this environment. However, to make a fair comparison with other experiments, where velocity and torso height information is not directly provided, we also remove these parts from our observation, leaving only the positions of the joint as observation in the second experiment. Thus, in the second experiment, the agent directly reads a 14-dimensional tensor per frame from the environment based on the position of each joint. The third experiment is conducted with the RL agent reading a tensor that contains the (x, y) coordinates of

RL Observations	Cumulative Reward
Joint Positions, Velocity, Torso Height from the Environment	969±2
Joint Positions from the Environment	935±3
Keypoints Extracted with KeyNet	709±3
VAI on Training Environment	889±3

Table 7: Cumulative rewards on *Walker, walk* task with 1) joint positions, velocity, and torso height from the environment as observations; 2) joint positions from the environment as observations; 3) keypoints extracted by KeyNet from images; 4) The proposed method VAI. The first two use the ground truth information, which is not accessible during real-world deployment, and serve as upper bounds. For experiment 2, 3, and 4, we use stack of 3 frames as input for the RL agent to infer the velocity since velocity information is missing. Since walker is a planar environment (the walker will not lean towards to away from the screen), the extracted keypoints should roughly correspond to positions from the significant parts of the walker body. The gap between experiments indicate that a limited number of keypoints from KeyNet on its own is not a sufficiently informative or accurate source for observations for an RL agent, which is in accord with our visualization in the main text about the keypoints' temporal inconsistency.

24 keypoints. The keypoints are from a KeyNet which reads an image input. The KeyNet is pre-trained with transporter. In the last experiment, we run VAI, with the adapter module trained from the same KeyNet that is used in the experiment above, on the training environment, although it is able to adapt to other environments as well and thus is more general. To infer velocity information, we stack the observations for three frames for experiments 2, 3, and 4. We run both experiments for 500k steps. We compare their efficiency by evaluating the agent in training environment 10 times with 10 seeds.

According to the performance of these RL agents in the training environment, keypoints on its own do not capture all the information needed by the RL agent accurately. This
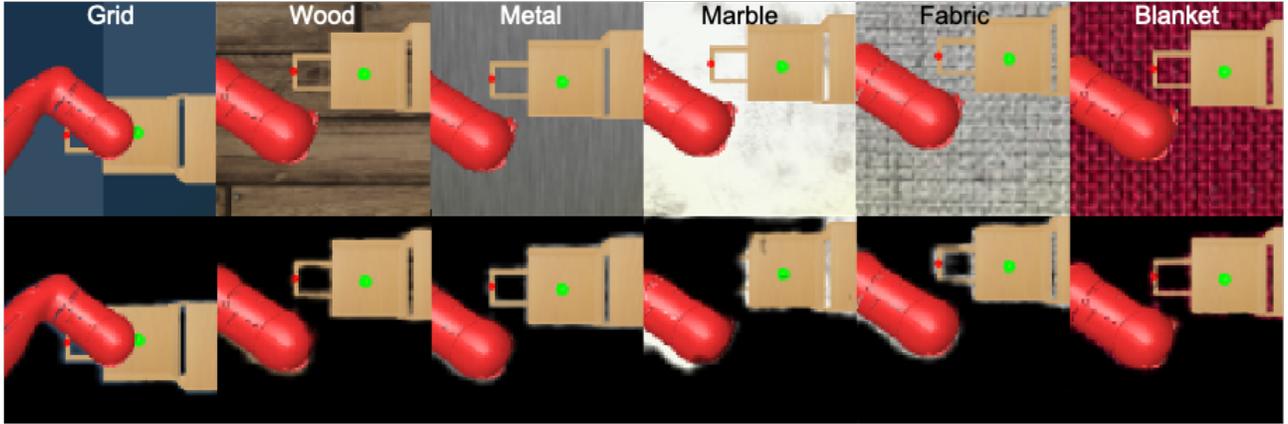
Figure 12: Samples from DrawerWorld, *DrawerClose* task with their corresponding observation processed by the adapter module. The Grid task is the training task for the adapter. All the observations use the same adapter for a fair comparison.

will be even worse if the agent is evaluated in a different environment is has never seen before, since KeyNet itself does not come with the ability to adapt, although the keypoints it generates are not supposed to carry domain-specific or distraction information. Using keypoints information along with image features as well as history observations may help, as illustrated in [24] and described in the main text, but it will add greatly to the complexity of the RL framework. What's more, agents may need information other than what keypoints provide. For example, keypoints do not carry the shape, size, and color information, which may be of paramount importance in certain tasks. Furthermore, since KeyNet allocates an output dimension for each keypoint, the number of parameters as well as computation time scales linearly with the number of keypoints, which prohibits adding a large number of keypoints to compensate the effect of temporal inconsistency or to capture complicated observations. In contrast, since KeyNet is not used in getting adapted observations in our method, the speed and number of parameters of our RL agent, including the adapter, at evaluation time are not affected by the number of keypoints used to generate ground-truth, which allows our method to scale to complicated environments with many moving parts without losing efficiency.

10. Visualizations of the Observation Adapter

How to make sure that RL will adapt to a certain setting that is different from training setting is still an open problem. Our method opts to work on observation-space. In contrast, PAD works on an intermediate encoder feature space. Our method is much easier to visualize and debug since humans are able to directly understand the quality of adapted observations while it is really difficult to understand what happens in the feature space.

To give examples on how to assess whether an adapter

works on a certain environment easily and to illustrate our performance in a visual way in evaluation environments, we gathered 6 pairs of raw samples and samples processed by the adapter in *DrawerClose* task from the same adapter in Fig. 12. As can be seen from the examples, the adapter model differentiates most of the evaluation environments well, with the exception of the marble environment, which the adapter confuses parts of the foreground and background such as the handle and the patches around the actuator, probably due to the fact that the reflected light on the actuator has a similar white color to the color of background. This indicates why our model performances worse in marble environment, as illustrated in the experiment section in the main text, and, in real-life applications, means that the adapter needs to be re-trained or fine-tuned with observations from similar environments, or if this is not applicable, with augmentation specially-designed to handle this case. We leave the question of handling adapter fine-tuning and re-training to later research.

This visualization has a large impact on the real-world applications of our method: with only a few observations from an intended deployment environment, one could easily visualize and assess whether our method will adapt to such environment. This does not require any ability to run the policy in the dynamics, nor does it require reward functions or consecutive observations which may be difficult to obtain from deployment environments in real-world applications. We strongly believe that this simple assessment provides a direction for future research in explainable, adaptable, and generalizable reinforcement learning and will present great benefit to potential applications of reinforcement learning.

References

[1] OpenAI: 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. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[2] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[4] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

[7] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[8] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[9] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[10] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

[11] Sander Greenland, James M Robins, and Judea Pearl. Confounding and collapsibility in causal inference. *Statistical science*, pages 29–46, 1999.

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

[13] Nicklas Hansen, Yu Sun, 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 and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. *arXiv preprint arXiv:2011.13389*, 2020.

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[16] Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*, 2015.

[17] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

[18] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In *Advances in Neural Information Processing Systems*, pages 15663–15674, 2019.

[19] Paul W Holland. Statistics and causal inference. *Journal of the American statistical Association*, 81(396):945–960, 1986.

[20] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[21] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks through conditional image generation. In *Advances in neural information processing systems*, pages 4016–4027, 2018.

[22] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[23] Mahesh Joshi, William W Cohen, Mark Dredze, and Carolyn P Rosé. Multi-domain learning: when do domains matter? In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1302–1312. Association for Computational Linguistics, 2012.

[24] Tejas D Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. In *Advances in neural information processing systems*, pages 10724–10734, 2019.

[25] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *CVPR*, 2017.

[26] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learming with augmented data. arXiv:2004.14990.

[27] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[28] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[29] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[30] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*, pages 67–82, 2018.

[31] Matthew T Mason. *Mechanics of robotic manipulation*. MIT press, 2001.

[32] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[33] Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural computation*, 17(2):335–359, 2005.

[34] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pages 9191–9200, 2018.

[35] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, pages 4293–4302, 2016.

[36] David F Nettleton, Albert Orriols-Puig, and Albert Fornells. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial intelligence review*, 33(4):275–306, 2010.

[37] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

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

[39] Vishal M Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, 32(3):53–69, 2015.

[40] Judea Pearl. *Causality*. Cambridge university press, 2009.

[41] Judea Pearl. Direct and indirect effects. *arXiv preprint arXiv:1301.2300*, 2013.

[42] Judea Pearl et al. Causal inference in statistics: An overview. *Statistics surveys*, 3:96–146, 2009.

[43] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic Books, 2018.

[44] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1–8. IEEE, 2018.

[45] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. *arXiv preprint arXiv:1703.02702*, 2017.

[46] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

[47] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, pages 506–516, 2017.

[48] Lorenzo Richiardi, Rino Bellocco, and Daniela Zugna. Mediation analysis in epidemiology: methods, interpretation and bias. *International journal of epidemiology*, 42(5):1511–1519, 2013.

[49] Amir Rosenfeld and John K Tsotsos. Incremental learning through deep adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[50] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[51] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

[52] Kaihua Tang, Jianqiang Huang, and Hanwang Zhang. Long-tailed classification by keeping the good and removing the bad momentum causal effect. *Advances in Neural Information Processing Systems*, 33, 2020.

[53] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[54] Yuval Tassa, Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, and Nicolas Heess. dm_control: Software and tasks for continuous control, 2020.

[55] 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 *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.

[56] Xudong Wang, Zhaowei Cai, Dashan Gao, and Nuno Vasconcelos. Towards universal object detection by domain attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7289–7298, 2019.

[57] Xudong Wang, Ziwei Liu, and Stella X Yu. Unsupervised feature learning by cross-level discrimination between instances and groups. *arXiv preprint arXiv:2008.03813*, 2020.

[58] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018.

[59] Jiachen Yang, Brenden Petersen, Hongyuan Zha, and Daniel Faissol. Single episode policy transfer in reinforcement learning. *arXiv preprint arXiv:1910.07719*, 2019.

[60] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.

[61] Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017.

[62] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

[63] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.