POLICY-INDUCED SELF-SUPERVISION IMPROVES REPRESENTATION FINETUNING IN VISUAL RL

Anonymous authors
Paper under double-blind review

ABSTRACT

We study how to transfer representations pretrained on source tasks to target tasks in visual percept based RL. We analyze two popular approaches: freezing or finetuning the pretrained representations. Empirical studies on a set of popular tasks reveal several properties of pretrained representations. First, finetuning is required even when pretrained representations perfectly capture the information required to solve the target task. Second, finetuned representations improve learnability and are more robust to noise. Third, pretrained bottom layers are task-agnostic and readily transferable to new tasks, while top layers encode task-specific information and require adaptation. Building on these insights, we propose a self-supervised objective that *clusters representations according to the policy they induce*, as opposed to traditional representation similarity measures which are policy-agnostic (*e.g.* Euclidean norm, cosine similarity). Together with freezing the bottom layers, this objective results in significantly better representation than frozen, finetuned, and self-supervised alternatives on a wide range of benchmarks.

1 INTRODUCTION

Learning representations via pretraining is a staple of modern transfer learning. Typically, a feature encoder is pretrained on one or a few source task(s). Then it is either frozen (*i.e.*, the encoder stays fixed) or finetuned (*i.e.*, the parameters of the encoders are to be updated) when solving a new downstream task (Yosinski et al., 2014). While whether to freeze or finetune is application-specific, finetuning outperforms in general freezing when there are sufficient (labeled) data and compute. This pretrain-then-transfer recipe has led to many success stories in vision (Razavian et al., 2014; Chen et al., 2021), speech (Amodei et al., 2016; Akbari et al., 2021), and NLP (Brown et al., 2020; Chowdhery et al., 2022).

For reinforcement learning (RL), however, finetuning is a costly option as the learning agent needs to collect its own data specific to the downstream task. Moreover, when the source tasks are very different from the downstream task, the first few updates from finetuning destroy the representations learned on the source tasks, cancelling all potential benefits of transferring from pretraining. For those reasons, practitioners often choose to freeze representations, thus completely preventing finetuning.

But representation freezing has its own shortcomings, especially pronounced in visual RL, where the (visual) feature encoder can be pretrained on existing image datasets such as ImageNet (Deng et al., 2009) and even collections of web images. Such generic but easier-to-annotate datasets are not constructured with downstream (control) tasks in mind and the pretraining does not necesarily capture important attributes used to solve those tasks. For example, the downstream embodied AI task of navigating around household items (Savva et al., 2019; Kolve et al., 2017) requires knowing the precise size of the objects in the scene. Yet this information is not required when pretraining on visual objection categorization tasks, resulting in what is called *negative transfer* where a frozen representation hurts downstream performance.

More seriously, even when the (visual) representation needed for the downstream task is known *a priori*, it is unclear that learning it from the source tasks then freezing it should be preferred to finetuning, as shown in Figure 1. On the left two plots, freezing representations (`Frozen`) underperforms learning representations using only downstream data (`De Novo`). On the right two plots, we observe the opposite outcome. Finetuning representations (`Finetuned`) performs well overall, but fails to unequivocally outperform freezing on the rightmost plots.

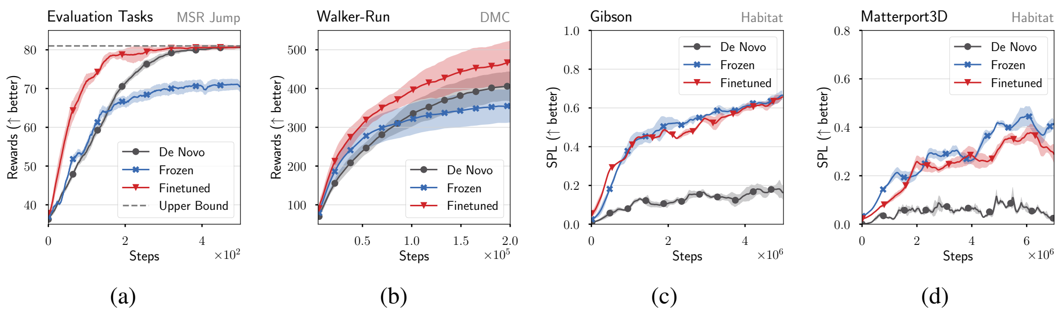


Figure 1: **When should we freeze or finetune pretrained representations in visual RL?** Reward and success weighted by path length (SPL) transfer curves on `MSR Jump`, `DeepMind Control`, and `Habitat` tasks. Freezing pretrained representations can underperform no pretraining at all (Figures 1a and 1b) or outperform it (Figures 1c and 1d). Finetuning representations is always competitive, but fails to significantly outperform freezing on visually complex domains (Figures 1c and 1d). Solid lines indicate mean over 5 random seeds, shades denote 95% confidence interval. See Section 3.2 for details.

Contributions *When should we freeze representations, when do they require finetuning, and why?* This paper answers those questions through several empirical studies on visual RL tasks ranging from simple game and robotic tasks (Tachet des Combes et al., 2018; Tassa et al., 2018) to photo-realistic `Habitat` domains (Savva et al., 2019).

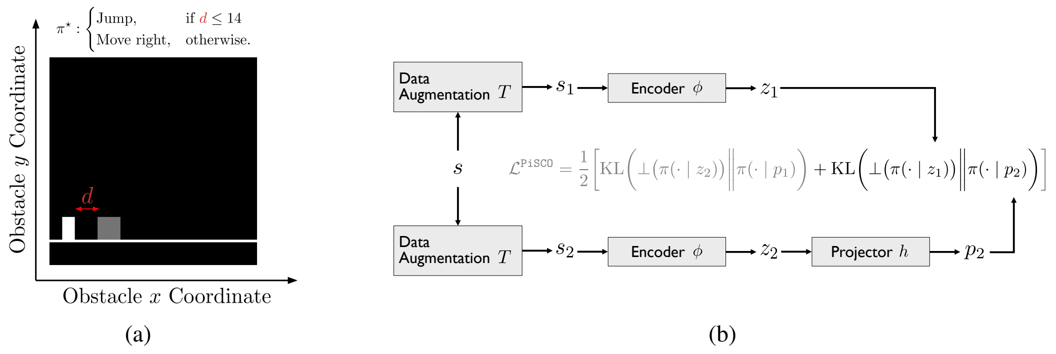
Our studies highlight properties of finetuned representations which improve learnability. First, they are more consistent in clustering states according to the actions they induce on the downstream task; second, they are more robust to noisy state observations. Inspired by these empirical findings, we propose `PiSCO`, **a representation finetuning objective which encourages robustness and consistency with respect to *the actions they induce*.**

We also show that visual percept feature encoders first compute task-agnostic information and then refine this information to be task-specific (*i.e.*, predictive of rewards and / or dynamics) — a well-known lesson from computer vision (Yosinski et al., 2014) but, to the best of our knowledge, never demonstrated for RL so far. We suspect that finetuning with RL destroys the task-agnostic (and readily transferrable) information found in lower layers of the feature encoder, thus cancelling the benefits of transfer. To retain this information, we show how to identify transferrable layers, and propose to **freeze those layers while adapting the remaining ones** with `PiSCO`. This combination yields excellent results on all testbeds, outperforming both representation freezing and finetuning.

2 Related works and background

Learning representations for visual reinforcement learning (RL) has come a long way since its early days. Testbeds have evolved from simple video games (Koutník et al., 2013; Mnih et al., 2015) to self-driving simulators (Shah et al., 2018; Dosovitskiy et al., 2017), realistic robotics engines (Tassa et al., 2018; Makoviychuk et al., 2021), and embodied AI platforms (Savva et al., 2019; Kolve et al., 2017). Visual RL algorithms have similarly progressed, and can now match human efficiency and performance on the simpler video games (Hafner et al., 2021; Ye et al., 2021), and control complex simulated robots in a handful of hours (Yarats et al., 2022; Laskin et al., 2020; Hansen et al., 2022).

In spite of this success, visual representations remain challenging to transfer in RL (Lazaric, 2012). Prior work shows that learned representations can be surprisingly brittle, and fail to generalize to minor changes in pixel observations (Witty et al., 2021). This perspective is often studied under the umbrella of generalization (Zhang et al., 2018; Cobbe et al., 2020; Packer et al., 2018) or adversarial RL (Pinto et al., 2017; Khalifa et al., 2020). In part, those issues arise due to our lack of understanding in what can be transferred and how to learn it in RL. Others have argued for a plethora of representation pretraining objectives, ranging from capturing policy values (Lehnert et al., 2020; Liu et al., 2021) and summarizing states (Mazoure et al., 2020; Schwarzer et al., 2021; Abel et al., 2016; Littman & Sutton, 2001) to disentangled (Higgins et al., 2017) and bi-simulation metrics (Zhang et al., 2021; Castro & Precup, 2010). Those objectives can also be aggregated in the hope of learning more generic representations (Gelada et al., 2019; Yang & Nachum, 2021). Nonetheless, it remains unclear which of these methods should be preferred to learn generic transferrable representations.



$\pi^* : \begin{cases} \text{Jump}, & \text{if } d \leq 14 \\ \text{Move right}, & \text{otherwise.} \end{cases}$
Obstacle y Coordinate
d
Obstacle x Coordinate
(a)
Data Augmentation T
s_1
Encoder ϕ
z_1
s
$\mathcal{L}^{\texttt{PiSCO}} = \frac{1}{2}\left[\mathrm{KL}\left(\perp(\pi(\cdot \mid z_2))\middle\|\pi(\cdot \mid p_1)\right) + \mathrm{KL}\left(\perp(\pi(\cdot \mid z_1))\middle\|\pi(\cdot \mid p_2)\right)\right]$
Data Augmentation T
s_2
Encoder ϕ
z_2
Projector h
p_2
(b)

Figure 2: **(a) Annotated observation from `MSR Jump`.** On this simple game, the optimal policy consists of moving right until the agent (white) is 14 pixels away from the obstacle (gray), at which point it should jump. **(b) Diagram for our proposed `PiSCO` consistency objective.** It encourages the policy's actions to be consistent for perturbation of state s, as measured through the actions induced by the policy. See Section 4 for details.

Those issues are exacerbated when we pretrain representations on generic vision datasets (*e.g.*, ImageNet (Deng et al., 2009)), as in our `Habitat` experiments. Most relevant to this setting are the recent works of Xiao et al. (2022) and Parisi et al. (2022): both pretrain their feature encoder on real-world images and freeze it before transfer to robotics or embodied AI tasks. However, neither reports satisfying results as they both struggle to outperform a non-visual baseline built on top of proprioceptive states. This observation indicates that more work is needed to successfully apply the pretrain-then-transfer pipeline to visual RL. A potentially simple source of gain comes finetuning those generic representations to the downstream task, as suggested by Yamada et al. (2022). While this is computationally expensive, prior work has hinted at the utility of finetuning in visual RL (Schwarzer et al., 2021; Wang et al., 2022).

In this work, we zero-in on representation freezing and finetuning in visual RL. Unlike prior representation learning work, our goal is not to learn generic representations that transfer; rather, we would like to specialize those representations to a specific downstream task. We contribute in Section 3 an extensive empirical analysis highlighting important properties of finetuning, and take the view that finetuning should emphasize features that easily discriminate between good and bad action on the downstream task — in other words, *finetuning should ease decision making*. Building on this analysis, we propose a novel method to simplify finetuning, and demonstrate its effectiveness on challenging visual RL tasks where naive finetuning fails to improve upon representation freezing.

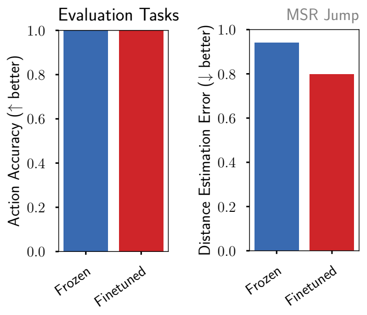
3 Understanding when to freeze and when to finetune

We presents empricial analysis of representation freezing and finetuning. In Sections 3.1 and 3.2 we detail the results from Figure 1 mentioned above. Then, we show in Section 3.3 a surprising result on the `MSR Jump` tasks: although freezing the pretrained representations results in negative transfer, those representations are sufficiently informative to perfectly solve the downstream task. This result indicates that capturing the relevant visual information through pretraining is not always effective for RL transfer. Instead, Section 3.4 shows that representations need to emphasize task-relevant information to improve learnability – an insight we explore in Section 3.5 and build upon in Section 4 to motivate a simple and effective approach to finetuning.

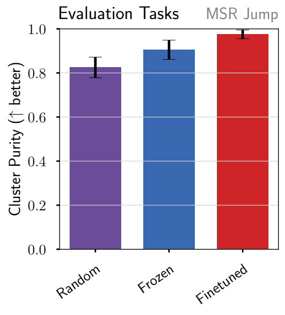
3.1 Experimental testbeds

Our testbeds are standard and thoroughly studied RL tasks in literature. Our description here focuses on using to study RL transfer. We defer details and implementations to Appendix B.

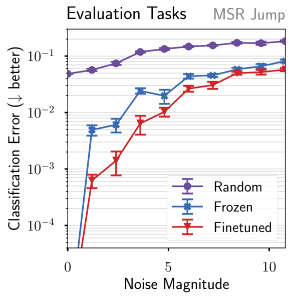
MSR Jump (Tachet des Combes et al., 2018) The agent (a gray block) needs to cross the screen from left to right by jumping over an obstacle (a white block). The agent's observations are video game frames displaying the entire world state (see Figure 2a) based on which it can choose to "move right" or "jump". We generate 140 tasks by changing the position of the obstacle vertically and horizontally; half of them are used for pretraining, half for evaluation. We train the convolutional actor-critic agent from Mnih et al. (2016) with PPO (Schulman et al., 2017) for 500 iterations on all pretraining tasks, and transfer its 4-layer feature encoder to jointly solve all evaluation tasks. Like pretraining, we train the actor and critic heads of the agent with PPO for 500 iterations during transfer.



(a)



(b)



(c)

Figure 3: **(a) Transfer can still fail even though frozen representations are informative enough.** On MSR Jump tasks, we can perfectly regress the optimal actions (accuracy $= 1.0$) and agent-obstacle distance (mean square error < 1.0) with *frozen or finetuned* representations. Combined with Figure 1a, those results indicate that *capturing the right information is not enough for RL transfer*. **(b) Finetuned representations yield purer clusters.** Given a state, we measure the expected purity of the cluster consisting of the 5 closest neighbours of that state in representation space. For Finetuned representations, this metric is significantly higher (98.75%) than for Frozen (91.41%) or Random (82.98%), showing that states which beget the same actions are closer to one another and thus easier to learn. **(c) Finetuned representations are more robust to perturbations.** For source and downstream tasks, the classification error (1 - action accuracy) degrades significantly more slowly for Finetuned representations than for Frozen ones under increasing data augmentation noise, suggesting that robustness to noise improves learnability. See Section 3.3 for details.

DeepMind Control (DMC) (Tassa et al., 2018) This robotics suite is based on the MuJoCo (Todorov et al., 2012) physics engine. We use visual observations as described by Yarats et al. (2021), and closely replicate their training setup. The learning agent consists of action-value and policy heads, together with a shared convolutional feature encoder whose representations are used *in lieu* of the image observations. We pretrain with DrQ-v2 (Yarats et al., 2022) on a single task from a given domain, and transfer only the feature encoder to a different task from the same domain. For example, our `Walker` agent is pretrained on `Walker-Walk` and transfers to `Walker-Run`.

Note: The main text uses `Walker` to illustrate results on `DeepMind Control`, and more tasks are included in Appendix C (including `Cartpole Balance`→`Swingup` and `Hopper Stand`→`Hop`).

Habitat (Savva et al., 2019) In this setting, we pretrain a ConvNeXt (tiny) (Liu et al., 2022) to classify images from the ImageNet (Deng et al., 2009) dataset. We use the "trunk" of the ConvNeXt as a feature encoder (*i.e.*, discard the classification head) and transfer it to either Gibson (Shen et al., 2021) or Matterport3D (Chang et al., 2017) scenes simulated with the `Habitat` embodied AI simulator. The tasks in `Habitat` consist of navigating to a point from an indoor scene (*i.e.*, PointNav) from visual observations, aided with GPS and compass sensing. Our implementation replaces the residual network from "Habitat-Lab" (Szot et al., 2021) with our ConvNeXt[1], and uses the provided DDPPO (Wijmans et al., 2020) implementation for transfer training. We train for 5M steps on Gibson and 7M steps on Matterport3D, include all scenes from the respective datasets, and report reward weighted by path length (SPL).

3.2 When freezing or finetuning works

As a first experiment, we take the pretrained feature encoders from the previous section and transfer them to their respective downstream tasks. We consider three scenarios.

- `De Novo`: where the feature encoder is randomly initialized (*not* pretrained) and its representations are learned on the downstream task only. The feature encoder might use different hyper-parameters (*e.g.*, learning rate, momentum, etc) from the rest of the agent.
- `Frozen`: where the feature encoder is pretrained and its representations are frozen (*i.e.*, we never update the weights of the feature encoder).
- `Finetuned`: where the feature encoder is pretrained and its representations are finetuned (*i.e.*, we update the weights of the feature encoder using the data and the algorithm of the downstream task). As for `De Novo`, the hyper-parameters for the feature encoder are tuned separately from the other hyper-parameters of the model.

[1]We choose a ConvNeXt because it removes batch normalization and dropout layers after pretraining.

Figure 1 displays those results. Comparing `De Novo` and `Frozen` shows that freezing representations works well on `Habitat` (see Figures 1c and 1d), likely because these tasks demand significantly richer visual observations. However, freezing representations completely fails on the simpler `MSR Jump` and `DeepMind Control` tasks (see Figures 1a and 1b). The pretrained feature encoders don't provide any benefit over a randomly initialized ones, and even significantly hurt transfer on `MSR Jump`. The latter is especially surprising because there is very little difference between pretraining and evaluation tasks (only the obstacle position is changed). Those results beg the question: *why does transfer fail so badly on `MSR Jump`?*

On the other hand, `Finetuned` performs reasonably well across all benchmarks but those results hide an important caveat: finetuning can sometimes collapse to representation freezing. This was the case on `Habitat`, where our hyper-parameter search resulted in a tiny learning rate for the feature encoder ($1e^{-6}$, the smallest in our grid-search), effectively preventing the encoder to adapt. We hypothesize that on those large-scale and visually challenging tasks, `Finetuned` offers no improvement over `Frozen` because RL updates are too noisy for finetuning. Thus, we ask: *how can we help stabilize finetuning in RL?*

We answer both questions in the remaining of this paper. We take a closer look at `MSR Jump` tasks and explain why transfer failed. We show that a core role of finetuning is to emphasize task-specific information, and that layers which compute task-agnostic information can be frozen without degrading transfer performance. In turn, this suggests an instinctive strategy to help stabilize RL training: freezing task-agnostic layers, and only finetuning task-specific ones.

3.3 Freezing fails even when learned representations are useful

We now show that, on `MSR Jump`, the pretrained representations are informative enough to perfectly solve all evaluation tasks.

Due to the simplicity of the dynamics, we can manually devise the following optimal policy because the size of the agent, the size of the obstacle, and the jump height are shared across all tasks:

$$\pi^* = \begin{cases} \text{Jump}, & \text{if the agent is 14 pixels to the left of the obstacle.} \\ \text{Move right}, & \text{otherwise.} \end{cases}$$

We empirically confirmed this policy reaches the maximal rewards (81) on all pretraining and evaluation tasks. Immediately, we see that the distance from the agent to the obstacle is the only information that matters to solve `MSR Jump` tasks. This observation suggests the following experiment.

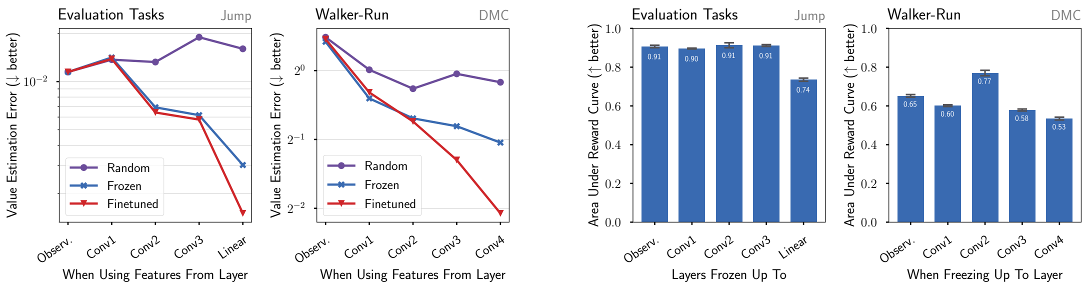
To measure the informativeness of pretrained representations, we regress the distance between the agent and the obstacle from pretrained representations. If the estimation error (*i.e.*, mean square error) is lower than 1.0 pixel, the representations are informative enough to perfectly solve `MSR Jump` tasks. To further validate this scenario, we also regress the optimal actions from the simple policy above given pretrained representations. Again, if the acurracy is perfect the representations are good enough. We repeat this experiment for finetuned representations, and compare the results against one another.

Figure 3a reports results for pretrained and finetuned (*i.e.*, after transfer) representations. We regress the distance and optimal actions with linear models from evaluation task observations, using a mean square error and binary cross-entropy loss. Surprisingly, *both* pretrained and finetuned representations can perfectly regress the distance and optimal actions: they get sub-pixel estimation error (0.94 and 0.79, respectively) and perfect accuracy (both 100%). Combined with the previous section, these results indicate that capturing the right visual information is not enough for successful transfer in RL. They also put forward the following hypothesis: *one role of finetuning is to emphasize information that eases decision making.* This hypothesis is supported by the lower distance error for `Finetuned` in the right panel of Figure 3a.

3.4 Finetuning improves learnability and robustness to noise

How is information refined for decision making? Our next set of experiments answers this question by highlighting differences between pretrained and finetuned representations.

First, we measure the ability of each feature encoder to cluster states assigned to identical actions together. Intuitively, we expect states that require similar actions to have similar representations.



(a) (b)

Figure 4: **Frozen layers that retain the same information as finetuned layers can be kept frozen.** **(a)** Mean squared error when linearly regressing action values from representations computed by a given layer. Some early layers are equally good at predicting action values before or after finetuning. This suggests those layers can be kept frozen, potentially stabilizing training. See Section 3.5 for details. **(b)** Area under the reward curve when freezing up to a given layer and finetuning the rest. Freezing early layers does not degrade performance, and sometimes improves it (*e.g.*, up to `Conv2` on `DeepMind Control`).

To that end, we compute the average purity when clustering a state with its 5 closest neighbors in representation space. In other words, given state s, we find the 5 closest neighbors to x in representation space and count how many of them are assigned the same optimal action as s. Figure 3b the average cluster purity for random, pretrained, and finetuned representations. We find that `Finetuned` representations yield clusters with significantly higher purity (98.75%) than `Frozen` (91.41%) or random (82.98%) ones, which explains why PPO converges more quickly when representations are finetuned: it is easier to discriminate between the states requiring "jump" *v.s.* "move right", if the representations for those states cluster homogenously.

Second, we uncover another property of finetuned representations, namely, that they are more robust to perturbations. We replicate the optimal action classification setup from Figure 3a but perturb states randomly rotating them. For each image, the degree of the rotation is randomly sampled from range $(-x, x)$. Figure 3c shows how classification accuracy degrades as we let x increase from $0°$ to $12°$. Both `Frozen` and `Finetuned` degrade with more noise, but `Finetuned` is more robust and degrades more slowly.

Those results point to finetuned representations that consistently map to similar actions and are robust to perturbations They set the stage for Section 4 where we introduce `PiSCO`, a policy consistency objective which builds on these insights. In the next section, we continue our analysis and further investigate how information gets refined *in each layer* of the feature encoder.

3.5 When and why is representation finetuning required?

As the previous section uncovered important pitfalls of representation freezing, we now turn to finetuning in the hope of understanding why it succeeds when it does. To that end, we dive into the representations of individual layers in `Finetuned` feature encoders and compare them to the `Frozen` and random representations. We take inspiration from the previous section and hypothesize that "purpose drives adaptation"; that is, a layer only needs finetuning if it computes task-specific information that eases decision making on the downstream task. Conversely, layers that compute task-agnostic information can be frozen thus potentially stabilizing RL training. To test this hypothesis, we conduct two layer-by-layer experiments: linear probing and incremental freezing.

With layer-by-layer linear probing, we aim to measure how relevant each layer is when choosing an action (Ding et al., 2021; Zhang et al., 2022). To start, we collect a dataset of $10,000$ state-action pairs and their corresponding action values with the `Finetuned` policy on the downstream task. The action values are computed with an action value function trained to minimize the temporal difference error on the collected state-action pairs (more details in Appendix C). We then proceed as follows for each layer l in the feature encoder. First, we store the representations computed by layer l, and extract their most salient features by projecting them to a 50-dimensional vector with PCA. Second, we regress action values from the projected representations and their corresponding actions using a linear regression model. Finally, we report the mean squared error for this linear model at layer l. Intuitively, the lower the linear probing error the easier it is to predict how good an action is for a given state[2]. We repeat this entire process for `Frozen` and `Finetuned` feature encoders, as well as one with a randomly initialized weights (*i.e.*, `Random`).

Results for this experiment are shown in Figure 4a. On all testbeds, both `Frozen` and `Finetuned` curves trend downwards and lie significantly lower than `Random`, indicating, as expected, that representations specialize for decision making as we move up the feature encoder. We also notice that the first few layers of `Frozen` and `Finetuned` track each other closely before `Frozen` starts to stagnate. On `MSR Jump`, this happens after the last convolutional layer (`Conv3`) and on `DeepMind Control` after the second one (`Conv2`). This evidence further supports our previous insight that early layers compute task-agnostic information, and suggests a new hypothesis: *can we freeze the early layers without degrading finetuning performance?* If so, this strategy might help stabilize finetuning in visually complex tasks like `Habitat`.

We confirm this hypothesis by combining freezing and finetuning in our second layer-by-layer experiment. For each target layer l, we take the pretrained feature encoder and freeze all layers up to and including l; the remaning layers are finetuned using the same setup as Figure 1.

Figure 4b summarizes this experiment with the (normalized) area under the reward curve (AURC). We preferred this metric over "highest reward achieved" since the latter does not consider how quickly the agent learns. On `MSR Jump`, the pretrained layers that have similar value estimation error in Figure 4a can be frozen without degrading adaptation. But, freezing yields lower AURC when value estimation error stagnates (as for `Linear`). Similarly, freezing the last two layers on `DeepMind Control` (`Conv3` and `Conv4`, which did not match `Finetuned`'s value estimation error) also degrades performance. We also see that adapting too many layers (*i.e.*, when `Conv1` is frozen but not `Conv2`) reduces the AURC. The training curves show this is due to slower convergence, suggesting that `Conv2` already computes useful representations which can be readily used for the downstream task.

Merging the observations from our layer-by-layer experiments, we conclude that *pretrained layers which extract task-agnostic information can be frozen*. We show in Section 5 that this conclusion significantly helps stabilize training when finetuning struggles, as in `Habitat`.

4 Finetuning with a policy-induced self-supervised objective

Section 3.4 suggests that an important outcome of finetuning is that states which are assigned the same actions cluster together. This section builds on this insight and introduces `PiSCO`, a self-supervised objective which attempts to accelerate the discovery of representations which cluster together for similar actions. At a high level `PiSCO` works as follows: given a policy π, it ensures that $\pi(a \mid s_1)$ and $\pi(a \mid s_2)$ are similar if states s_1 and s_2 should yield similar actions (*e.g.*, s_1, s_2 are perturbed version of state s). This objective is self-supervised because it applies to *any* policy π (not just the optimal one) and doesn't require knowledge of rewards nor dynamics.

More formally, we assume access to a batch of states $\mathcal{B}$ from which we can sample state s. Further, we compute two embedding representations for state s. The first, $z = \phi(s)$, is obtained by passing s through an encoder $\phi(\cdot)$; the second, $p = h(z)$, applies a projector $h(\cdot)$ on top of representation z. Both represenations z and p have identical dimensionality and can thus be used to condition the policy π. The core contribution behind `PiSCO` is to measure the dissimilarity between z and p in terms of the distribution they induce through π:

$$\mathcal{D}(z, p) = \mathrm{KL}\left(\perp(\pi(\cdot \mid z)) \,||\, \pi(\cdot \mid p)\right),$$

where $\mathrm{KL}(\cdot||\cdot)$ is the Kullback-Leibler divergence, and $\perp(x)$ is the stop-gradient operator, which sets all partial derivatives with respect to x to 0. The choice of the Kullback-Leibler is arbitrary — other statistical divergence are valid alternatives.

The final ingredient in `PiSCO` is a distribution $T(s' \mid s)$ over perturbed states s'. This distribution is typically implemented by randomly applying a set of data augmentation transformations on the state s. Then, the `PiSCO` objective (for **P**olicy-**i**nduced **S**elf-**C**onsistency **O**bjective) is to minimize the

[2]We found action values to be better targets to measure ease of decision making than, say, action accuracy which is upper bounded by 1.0 as in Figure 3a.

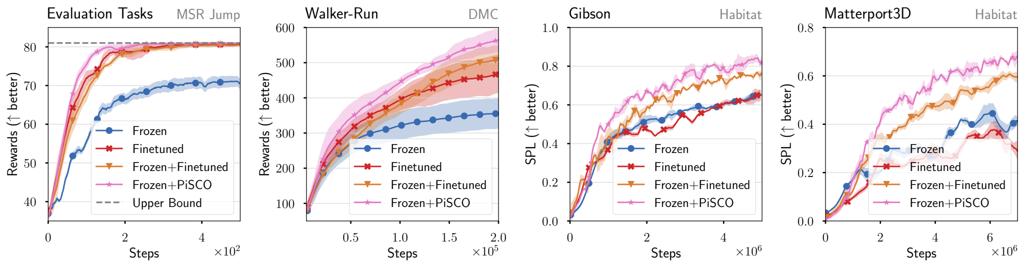


Figure 5: **Partial freezing improves convergence; adding our policy-induced consistency objective improves further.** As suggested by Section 3.5, we freeze the early layers of the feature extractor and finetune the rest of the parameters without (`Frozen+Finetuned`) or with our policy-induced consistency objective (`Frozen+PiSCO`). On challenging tasks (*e.g.*, `Habitat`), partial freezing dramatically boosts downstream performance, while `Frozen+PiSCO` further improves upon `Frozen+Finetuned` across the board.

dissimilarity between the representation of states sampled from T:

$$\mathcal{L}^{\texttt{PiSCO}} = \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{B} \\ s_1,s_2\sim \mathrm{T}(\cdot|s)}} \left[\frac{1}{2}\left(\mathcal{D}(z_1,p_2) + \mathcal{D}(z_2,p_1)\right)\right],$$

where s_1 and s_2 are two different perturbations obtained from state s, and the objective uses a symmetrized version of the dissimilarity measure $\mathcal{D}$. In practice, the `PiSCO` objective is added as an auxiliary term to the underlying RL objective and optimized for the encoder ϕ and the projector h. Pseudocode is available in Appendix D.

Remarks `PiSCO` is reminiscent of SimSiam (Chen & He, 2020) and only differs in how dissimilarity between embeddings is measured. Our proposed policy-induced similarity measure is crucial for best performance, as shown in Section 5.2. In fact, similar representation learning algorithms for RL could be derived by replacing the embedding similarity measure in existing self-supervised algorithms, such as SimCLR (Chen et al., 2020), MoCo (He et al., 2019), or VICReg (Bardes et al., 2022). We chose SimSiam for its simplicity – no target encoder nor negative samples required – which suits the requirements of RL training.

Alone, `PiSCO` is not a useful objective to solve RL tasks; rather, its utility stems from assisting the underlying RL algorithm in learning representations that are robust to perturbations. *Could we obtain the same benefits by learning the policy with augmented data (and thus side-stepping the need for `PiSCO`)?* In principle, yes. However, the policy objective is typically a function of the rewards which is known to be excessively noisy to robustly learn representations. For example, DrQ avoids learning representations through the policy objective, and instead relies on Bellman error minimization. We hypothesize that `PiSCO` succeeds in learning robust representations because its self-supervised objective is less noisy than the policy objective.

5 Experiments

We complete our analysis with a closer look at `PiSCO` and partially frozen feature encoders. First, we check whether partial freezing and policy-induced supervision can help accelerate finetuning with RL; second, we compares policy-induced supervision as an RL finetuning objective against more common similarity measures in self-supervised learning.

5.1 Partial freezing and policy-induced supervision improve RL finetuning

This section shows that partially freezing a pretrained feature encoder can significantly help stabilize downstream training. We revist the experimental setup of Figure 1, this time including two new transfer variations. The first one is `Frozen+Finetuned`, where we freeze early layers and finetune the remaining ones as with `Finetuned`. The second, `Frozen+PiSCO`, additionally includes the `PiSCO` objective as an auxiliary loss to help finetuning representations. `Frozen+PiSCO` involves one extra hyper-parameter, namely, the auxiliary loss weight, which we set to 0.01 for all benchmarks.

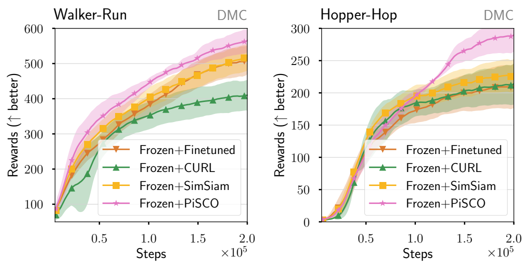


Figure 6: **Policy-induced supervision is a better objective than representation alignment for finetuning.** We compare our policy-induced consistency objective as a measure of representation similarity against popular representation learning objectives (*e.g.* SimSiam, CURL). Policy supervision provides more useful similarities for RL finetuning, which accelerates convergence and reaches higher rewards.

In both cases, we identify which layers to freeze and which to finetune following a similar linear probing setup as for Figure 4a. For each layer, we measure the action value estimation error of pretrained and finetuned representations, and only freeze the first pretrained layers that closely match the finetuned ones. On `MSR Jump`, we freeze up to `Conv3`; on `DeepMind Control` (`Walker`) up to `Conv2`; and on `Habitat`, we freeze upto `Conv8` for Gibson and upto `Conv7` for Matterport3D. (See the Appendix C for results on `Habitat`.)

We report convergence curves in Figure 5. As suggested by our analysis of Figure 4b, freezing those early layers does not degrade performance; in fact, we see significant gains on `Habitat` both in terms of convergence rate and in asymptotic rewards. Those results are particularly noticeable since `Finetuned` struggled to outperform `Frozen` on those tasks. We also note that `Frozen+PiSCO` improves upon `Frozen+Finetuned` across the board: it accelerates convergence on `MSR Jump`, and also improves asymptotic performance on `DeepMind Control` and `Habitat`. These results tie all our analyses together: they show that (a) freezing early layers can help stabilize transfer in visual RL, (b) which layer to freeze is predicted by how well it encodes action value information, and (c) policy-induced supervision (and `PiSCO`) is a useful objective to accelerate RL finetuning.

5.2 Policy-induced supervision improves upon contrastive predictive coding

As a final experiment, we answer how important it is to evaluate similarity through the policy rather than with representation alignment as in contrastive predictive coding. In other words, *could we swap `PiSCO` for SimSiam and obtain similar results in Figure 5?* Our ablation studies focus on `DeepMind Control` as it is more challenging than `MSR Jump` yet computationally tractable, unlike `Habitat`.

We include `Frozen+SimSiam` and `Frozen+CURL` in addition to `Frozen+Finetuned` and `Frozen+PiSCO`. `Frozen+SimSiam` is akin to `Frozen+PiSCO` but uses the negative cosine similarity to measure the dissimilarity between embeddings z and p. For `Frozen+CURL`, we implement the contrastive auxiliary loss of CURL (Laskin et al., 2020), a self-supervised method specifically designed for RL. Both of these alternatives use DrQ as the underlying RL algorithm.

Figure 6 reports convergence curves on `Walker Walk`→`Run` and `Hopper Stand`→`Hop`[3]. Including the SimSiam objective typically improves upon vanilla finetuning, but including CURL does not. We hypothesize that most of the benefits from CURL are already captured by DrQ, as shown in prior work (Yarats et al., 2021). `PiSCO` significantly outperforms the standard self-supervised methods thus demonstrating the efficacy of using the policy when measuring similarity between embeddings. We hypothesize that representation alignment objectives naturally map two similarly looking states to similar representations, even when they shouldn't (*e.g.*, when the agent is 15 and 14 pixels away from the obstacle in `MSR Jump`). Instead, `PiSCO` is free to assign different representations since those two states might induce different actions ("move right" and "jump", respectively).

6 Conclusion

Our analysis of representation learning for transfer reveals several new insights on the roles of finetuning: similarity between representations should reflect whether they induce the same or similar distribution of actions; not all layers in encoders of states need to be adapted. Building on those insights, we develop a hybrid approach which partially freezes the bottom (and readily transferrable) layers while finetuning the top layers with a policy-induced self-supervised objective. This approach is especially effective for hard downstream tasks, as it alleviates the challenges of finetuning rich visual representations with reinforcement learning.

[3] `Hopper-Hop` is especially difficult; *e.g.*, DrQ-v2 fails to solve it after 3M iterations (Yarats et al., 2022).

References

David Abel, David Hershkowitz, and Michael Littman. Near optimal behavior via approximate state abstraction. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2915–2923, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/abel16.html. Cited on p. 2.

Hassan Akbari, Liangzhe Yuan, Rui Qian, Wei-Hong Chuang, Shih-Fu Chang, Yin Cui, and Boqing Gong. VATT: Transformers for multimodal self-supervised learning from raw video, audio and text. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=RzYrn625bu8. Cited on p. 1.

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, Jie Chen, Jingdong Chen, Zhijie Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Ke Ding, Niandong Du, Erich Elsen, Jesse Engel, Weiwei Fang, Linxi Fan, Christopher Fougner, Liang Gao, Caixia Gong, Awni Hannun, Tony Han, Lappi Vaino Johannes, Bing Jiang, Cai Ju, Billy Jun, Patrick LeGresley, Libby Lin, Junjie Liu, Yang Liu, Weigao Li, Xiangang Li, Dongpeng Ma, Sharan Narang, Andrew Ng, Sherjil Ozair, Yiping Peng, Ryan Prenger, Sheng Qian, Zongfeng Quan, Jonathan Raiman, Vinay Rao, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Kavya Srinet, Anuroop Sriram, Haiyuan Tang, Liliang Tang, Chong Wang, Jidong Wang, Kaifu Wang, Yi Wang, Zhijian Wang, Zhiqian Wang, Shuang Wu, Likai Wei, Bo Xiao, Wen Xie, Yan Xie, Dani Yogatama, Bin Yuan, Jun Zhan, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 173–182. JMLR.org, 2016. Cited on p. 1.

Ba, Kiros, and Hinton. Layer normalization. *arXiv preprint arXiv:1607. 06450*, 2016. URL http://arxiv.org/pdf/1607.06450. Cited on p. 17.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xm6YD62D1Ub. Cited on p. 8.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf. Cited on p. 1.

Pablo Samuel Castro and Doina Precup. Using bisimulation for policy transfer in MDPs. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, July 2010. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/viewPaper/1907. Cited on p. 2.

A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niebner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *2017 International Conference on 3D Vision (3DV)*, pp. 667–676, Los Alamitos, CA, USA, oct 2017. IEEE Computer Society. doi: 10.1109/3DV.2017.00081. URL https://doi.ieeecomputersociety.org/10.1109/3DV.2017.00081. Cited on p. 4, 18.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé Iii and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 2020. URL https://proceedings.mlr.press/v119/chen20j.html. Cited on p. 8.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. November 2020. URL http://arxiv.org/abs/2011.10566. Cited on p. 8, 21, 22.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9640–9649, 2021. URL https://github.com/facebookresearch/moco-v3. Cited on p. 1.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling language modeling with pathways. April 2022. URL http://arxiv.org/abs/2204.02311. Cited on p. 1.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2048–2056. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/cobbe20a.html. Cited on p. 2.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, June 2009. URL http://dx.doi.org/10.1109/CVPR.2009.5206848. Cited on p. 1, 3, 4, 17.

Frances Ding, Jean-Stanislas Denain, and Jacob Steinhardt. Grounding representation similarity through statistical testing. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=_kwj6V53ZqB. Cited on p. 6.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg (eds.), *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pp. 1–16. PMLR, 13–15 Nov 2017. URL https://proceedings.mlr.press/v78/dosovitskiy17a.html. Cited on p. 2.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2170–2179. PMLR, 2019. URL http://proceedings.mlr.press/v97/gelada19a.html. Cited on p. 2.

Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0oabwyZbOu. Cited on p. 2.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv [cs.LG]*, March 2022. URL http://arxiv.org/abs/2203.04955. Cited on p. 2.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. November 2019. URL http://arxiv.org/abs/1911.05722. Cited on p. 8.

D Hendrycks, K Gimpel arXiv preprint arXiv:1606.08415, and 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arxiv.org*, 2016. URL https://arxiv.org/abs/1606.08415. Cited on p. 16, 17.

Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1480–1490. JMLR.org, 2017. Cited on p. 2.

Ahmed Khalifa, Philip Bontrager, Sam Earle, and Julian Togelius. Pcgrl: Procedural content generation via reinforcement learning. In *Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'20. AAAI Press, 2020. ISBN 978-1-57735-849-7. Cited on p. 2.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. December 2014. URL http://arxiv.org/abs/1412.6980. Cited on p. 17.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017. Cited on p. 1, 2.

Jan Koutník, Giuseppe Cuccu, Jürgen Schmidhuber, and Faustino Gomez. Evolving large-scale neural networks for vision-based reinforcement learning. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, GECCO '13, pp. 1061–1068, New York, NY, USA, July 2013. Association for Computing Machinery. URL https://doi.org/10.1145/2463372.2463509. Cited on p. 2.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5639–5650. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/laskin20a.html. Cited on p. 2, 9.

Alessandro Lazaric. Transfer in reinforcement learning: A framework and a survey. In Marco Wiering and Martijn van Otterlo (eds.), *Reinforcement Learning: State-of-the-Art*, pp. 143–173. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. URL https://doi.org/10.1007/978-3-642-27645-3_5. Cited on p. 2.

Lucas Lehnert, Michael L Littman, and Michael J Frank. Reward-predictive representations generalize across tasks in reinforcement learning. *PLoS Comput. Biol.*, 16(10):e1008317, October 2020. URL http://dx.doi.org/10.1371/journal.pcbi.1008317. Cited on p. 2.

Michael Littman and Richard S Sutton. Predictive representations of state. *Adv. Neural Inf. Process. Syst.*, 14, 2001. URL https://web.eecs.umich.edu/~baveja/Papers/psr.pdf. Cited on p. 2.

Guoqing Liu, Chuheng Zhang, Li Zhao, Tao Qin, Jinhua Zhu, Li Jian, Nenghai Yu, and Tie-Yan Liu. Return-based contrastive representation learning for reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=_TM6rT7tXke. Cited on p. 2.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. Cited on p. 4, 17.

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. August 2021. URL http://arxiv.org/abs/2108.10470. Cited on p. 2.

Bogdan Mazoure, Remi Tachet des Combes, Thang Long Doan, Philip Bachman, and Devon Hjelm. Deep reinforcement and infomax learning. In *NeurIPS 2020*. ACM, December 2020. URL https://www.microsoft.com/en-us/research/publication/deep-reinforcement-and-infomax-learning/. Cited on p. 2.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, February 2015. URL http://dx.doi.org/10.1038/nature14236. Cited on p. 2, 16.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016. Cited on p. 3.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, V Koltun, and D Song. Assessing generalization in deep reinforcement learning. *ArXiv*, 2018. URL https://www.semanticscholar.org/paper/caea502325b6a82b1b437c62585992609b5aa542. Cited on p. 2.

Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. The unsurprising effectiveness of Pre-Trained vision models for control. March 2022. URL http://arxiv.org/abs/2203.03580. Cited on p. 3.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, High-Performance deep learning library. In H Wallach, H Larochelle, A Beygelzimer, F d'Alche Buc, E Fox, and R Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8026–8037. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. Cited on p. 24.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2817–2826. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/pinto17a.html. Cited on p. 2.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 512–519, 2014. doi: 10.1109/CVPRW.2014.131. Cited on p. 1.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. Cited on p. 1, 2, 4, 17.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. July 2017. URL http://arxiv.org/abs/1707.06347. Cited on p. 3, 16, 23.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uCQfPZwRaUu. Cited on p. 2, 3, 22.

Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In Marco Hutter and Roland Siegwart (eds.), *Field and Service Robotics*, pp. 621–635, Cham, 2018. Springer International Publishing. ISBN 978-3-319-67361-5. Cited on p. 2.

Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne Tchapmi, Micael Tchapmi, Kent Vainio, Josiah Wong, Li Fei-Fei, and Silvio Savarese. igibson 1.0: A simulation environment for interactive tasks in large realistic scenes. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7520–7527, 2021. doi: 10.1109/IROS51168.2021.9636667. Cited on p. 4.

Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Cited on p. 4, 17.

Remi Tachet des Combes, Philip Bachman, and Harm van Seijen. Learning invariances for policy generalization, 2018. URL `https://openreview.net/forum?id=BJHRaK1PG`. Cited on p. 2, 3, 16.

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. January 2018. URL `http://arxiv.org/abs/1801.00690`. Cited on p. 2, 4, 17.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, October 2012. URL `http://dx.doi.org/10.1109/IROS.2012.6386109`. Cited on p. 4, 17.

Che Wang, Xufang Luo, Keith Ross, and Dongsheng Li. VRL3: A data-driven framework for visual deep reinforcement learning. *arXiv [cs.CV]*, February 2022. URL `http://arxiv.org/abs/2202.10324`. Cited on p. 3.

Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=H1gX8C4YPr`. Cited on p. 4, 18.

Sam Witty, Jun K. Lee, Emma Tosch, Akanksha Atrey, Kaleigh Clary, Michael L. Littman, and David Jensen. Measuring and characterizing generalization in deep reinforcement learning. *Applied AI Letters*, 2(4):e45, 2021. doi: https://doi.org/10.1002/ail2.45. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/ail2.45`. Cited on p. 2.

Fei Xia, Amir R. Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018. Cited on p. 18.

Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv [cs.CV]*, March 2022. URL `http://arxiv.org/abs/2203.06173`. Cited on p. 3.

Jun Yamada, Karl Pertsch, Anisha Gunjal, and Joseph J Lim. Task-induced representation learning. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=OzyXtIZAzFv`. Cited on p. 3.

Mengjiao Yang and Ofir Nachum. Representation matters: Offline pretraining for sequential decision making. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11784–11794. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/yang21h.html`. Cited on p. 2.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=GY6-6sTvGaf`. Cited on p. 4, 9, 17.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=_SJ-_yyes8`. Cited on p. 2, 4, 9, 17, 19, 23.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *Adv. Neural Inf. Process. Syst.*, 34, 2021. URL `https://proceedings.neurips.cc/paper/2021/file/d5eca8dc3820cad9fe56a3bafda65ca1-Paper.pdf`. Cited on p. 2.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems*, 27:3320–3328, 2014. Cited on p. 1, 2.

Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. June 2018. URL `http://arxiv.org/abs/1806.07937`. Cited on p. 2.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=-2FCwDKRREu`. Cited on p. 2.

Wancong Zhang, Anthony GX-Chen, Vlad Sobal, Yann LeCun, and Nicolas Carion. Light-weight probing of unsupervised representations for reinforcement learning. *arXiv [cs.LG]*, August 2022. URL `http://arxiv.org/abs/2208.12345`. Cited on p. 6.

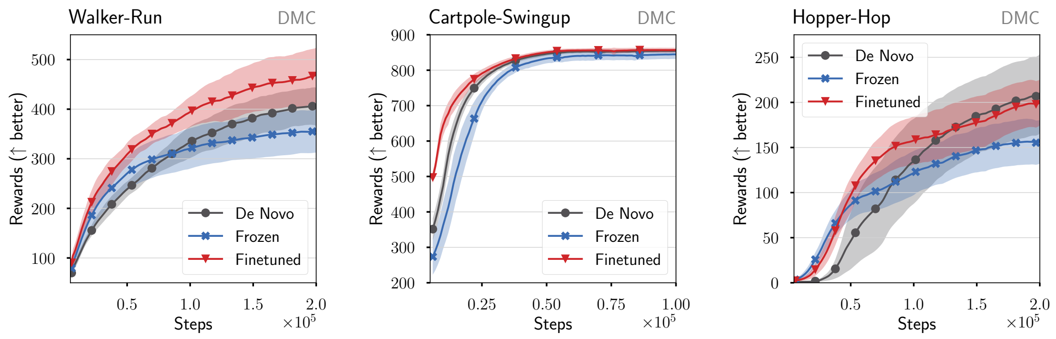


Figure 7: **When should we freeze or finetune pretrained representations in visual RL?** Reward transfer curves on `Cartpole` and `Hopper` `DeepMind Control` domains; replicates Figure 1. Freezing pretrained representations can underperform no pretraining at all (Figures 1a and 1b) or outperform it (Figures 1c and 1d). Finetuning representations is always competitive, but fails to significantly outperform freezing on visually complex domains (Figures 1c and 1d). Solid lines indicate mean over 5 random seeds, shades denote 95% confidence interval. See Section 3.2 for details.

A Appendix

B Testbed details

This section details the pretraining and transfer setups for the `MSR Jump`, `DeepMind Control`, and `Habitat` domains.

B.1 `MSR Jump` (Tachet des Combes et al., 2018)

As discussed in Section 3, the goal of `MSR Jump` tasks is for the agent to reach the right-hand side of the screen. The agent always starts on the left-hand side and has to jump over a single gray obstacle. The agent observes the entire screen from pixel values, and the only factors of variation are the x and y coordinates of the obstacle (*i.e.*, its horizontal position and the floor height). Its only possible actions are to move right or jump. The jump dynamics are shared across tasks and mid-air actions don't affect the subsequent states until the agent has landed (*i.e.*, all jumps go up by 15 pixels diagonally, and down by 15 pixels diagonally). Our experiments carefully replicate the reward and dynamics of Tachet des Combes et al. (2018), save for one aspect: we increase the screen size from 64 pixels to 84 pixels, as this significantly accelerated training with the DQN (Mnih et al., 2015) feature encoder. We provide a fast GPU-accelerated implementation of `MSR Jump` tasks in our code release (see Appendix E).

B.1.1 Pretraining setup

Source tasks We pretrain on 70 source tasks, varying both the obstacle x (from 15 to 50 pixels) and y (from 5 to 50) coordinates, every 5 pixel. We don't use frame stacking nor action repeats.

Learning agent We train with PPO (Schulman et al., 2017) for 500 iterations, where each iteration collects 80 episodes from randomly sampled source tasks. We use those episodes to update the agent 320 times with a batch size of 64, a discount factor of 0.99, and policy and value clip of 0.2. The learning rate starts at 0.0005 and is decayed by 0.99 once every 320 update. The agent consists of the 4-layer DQN feature encoder and linear policy and value heads. All models use GELU (Hendrycks et al., 2016) activations.

B.1.2 Transfer setup

Downstream tasks At transfer time, we replace the 70 source tasks with 70 unseen downstream tasks. Those downstream tasks are obtained by shifting the obstacle position (x and y coordinates) of each source task by 2 pixels. For example, if a source task has an obstacle at $x = 20$ and $y = 30$ there is a corresponding downstream task at $x = 22$ and $y = 32$.

Learning agent The learning agent is identical to the source tasks, but rather than randomly initializing the weights of the feature encoder we use the final weights from pretraining. As mentioned in the main text, we also introduce one additional hyper-parameter as we decouple the learning rate for the feature encoder and policy / value heads. For `Finetuned`, these learning rates are 0.001 for the heads and 0.0001 for the feature encoder; both learning rates are decayed during finetuning.

B.2 `DeepMind Control` (Tassa et al., 2018)

The `DeepMind Control` tasks build on top of the MuJoCo physiscs engine (Todorov et al., 2012) and implement a suite of robotic domains. Our experiments focus on `Walker`, `Cartpole`, and `Hopper` (see Appendix C.2 for the latter two). `Walker` and `Hopper` are robot locomotion domains, while `Cartpole` is the classic cartpole environment. On all domains, the agent receives an agent-centric visual 84×84 RGB observation, and controls its body with continuous actions. Our task implementation exactly replicate the one from DrQ (Yarats et al., 2021).

B.2.1 Pretraining setup

Source task On `Walker`, the source task is `Walk`, where the agent is tasked to walk in the forward direction. On `Cartpole` it is `Balance`, where the pole starts in a standing position and the goal is to balance it. And on `Hopper` it is `Stand`, where the agent needs to reach a (still) standing position. For all tasks, we keep a stack of the last 3 images and treat those as observations.

Learning agent Our learning agent is largely inspired by DrQ-v2 (Yarats et al., 2022), and comprises a 4-layer convolutional feature encoder, a policy head, and a twin action value head. Both the policy and action value heads consist of a projection layer, followed by LayerNorm (Ba et al., 2016) normalization, and a 2-layer MLP with 64 hidden units and GELU (Hendrycks et al., 2016) activations. We train with DrQ-v2 for 200k iterations, starting the policy standard deviation at 1.0, decaying it by 0.999954 every iteration until it reaches 0.1. We use a 512 batch size and don't use n-steps bootstrapping nor delayed updates. For pretraining, we always a learning rate of $3e^{-4}$ and default Adam (Kingma & Ba, 2014) hyper-parameters.

B.2.2 Transfer setup

Downstream task On `Walker`, we transfer to `Run`, where the agent needs to run in the forward direction. On `Cartpole`, the downstream task is `Swingup`, where the pole starts in downward position and should be swung and balanced upright. On `Hopper`, we use `Hop`, where the agents move in the forward direction by hopping.

Learning agent We use the same learning agent as for pretraining, but initialize the feature encoder with the weights obtained from pretraining. We also add a learning rate hyper-parameter when finetuning the feature encoder, which is tuned independently for each setting and method.

B.3 `Habitat` (Savva et al., 2019)

On `Habitat`, the goal of the agent is to navigate indoor environments to reach a desired location. To solve this point navigation task, it relies on two sensory inputs: a GPS with compass and visual observations. The visual observations are 256×256 RGB images (*n.b.*, without depth information). Our codebase builds on the task and agent definitions of "Habitat-Lab" (Szot et al., 2021), and only modifies them to implement our proposed methods.

B.3.1 Pretraining setup

Source task We use the ImageNet (Deng et al., 2009) dataset as source task to pretrain our feature encoder.

Learning agent The learning agent is a ConvNeXt-tiny (Liu et al., 2022) classifier, trained to discriminate between ImageNet-1K classes. We directly use the pretrained weights provided by the ConvNeXt authors, available at: `https://github.com/facebookresearch/ConvNeXt`

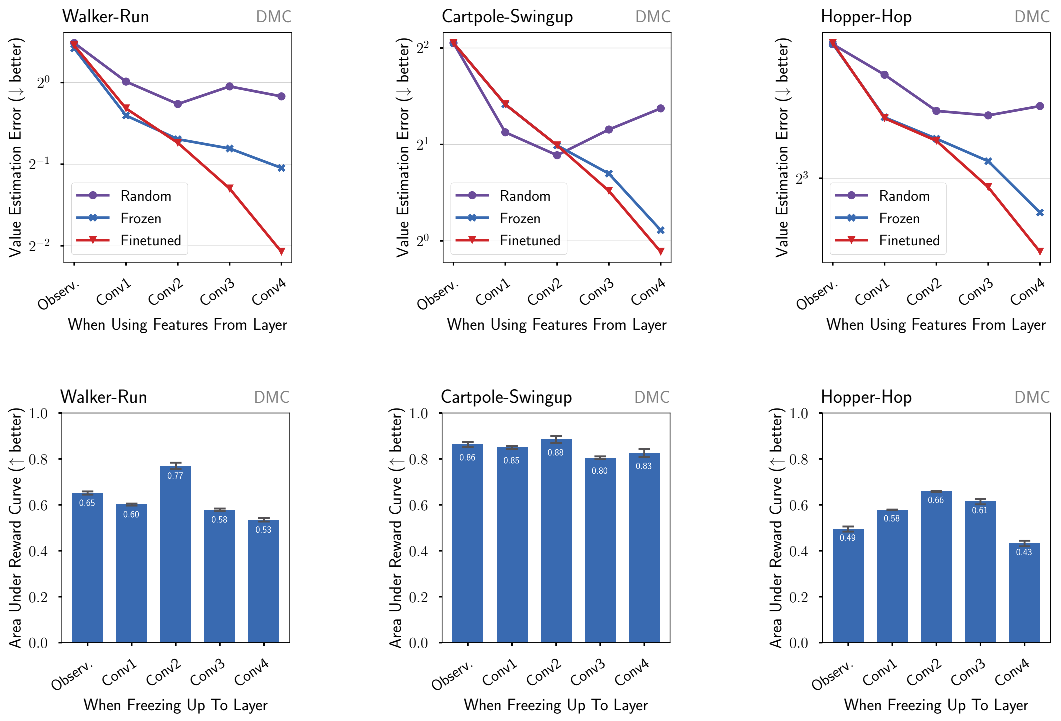


Figure 8: **Frozen layers that retain the same information as finetuned layers can be kept frozen.** Replicates Figure 4a on `Cartpole` and `Hopper` `DeepMind Control` domains. **(top)** Mean squared error when linearly regressing action values from representations computed by a given layer. Some early layers are equally good at predicting action values before or after finetuning. This suggests those layers can be kept frozen, potentially stabilizing training. See Section 3.5 for details. **(bottom)** Area under the reward curve when freezing up to a given layer and finetuning the rest. Freezing early layers does not degrade performance, and sometimes improves it (*e.g.*, up to `Conv2` on `DeepMind Control`).

B.3.2 Transfer setup

Downstream tasks For transfer, we load maps from the Gibson (Xia et al., 2018) and Matterport3D (Chang et al., 2017) datasets. Both are freely available for academic, non-commercial use at:

- Gibson: `https://github.com/StanfordVL/GibsonEnv/`
- Matterport3D: `https://github.com/niessner/Matterport/`

Learning agent We take the pretrained ConvNeXt classifier, discard the linear classification head, and only keep its trunk as our feature encoder. We train a 2-layer LSTM policy head and a linear value head on top of this feature encoder with DDPPO (Wijmans et al., 2020) for 5M iterations on Gibson and 10M on Matterport3D. For both tasks, we collect 128 steps with 4 workers for every update, which consists of 2 PPO epochs with a batch size of 4. We set PPO's entropy coefficient to 0.01, the value loss weight to 0.5, GAE's τ to 0.95, the discount factor to 0.99, and clip the gradient norm to have 0.2 l_2-norm. The learning rate for the policy and value heads is set to $2.4e^{-4}$, and the feature encoder's learning rate to $1e^{-4}$ for `Frozen+Finetuned`. For `Frozen+Varnish`, we use a single learning rate set to $1e^{-4}$ for the feature encoder and policy / value heads (but $2.5e^{-4}$ gave similar results). We never decay learning rates.

C Additional experiments

This section provides additional details on the experiments from the main text, and also includes additional results to further support our conclusions.

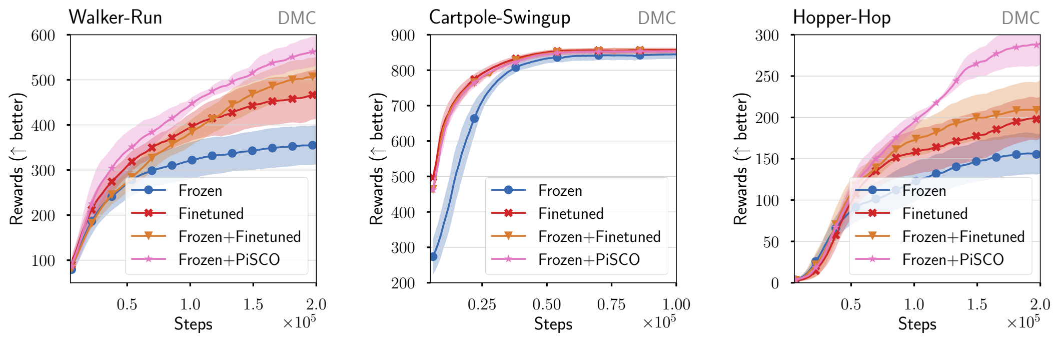


Figure 9: **Partial freezing improves convergence; adding our policy-induced consistency objective improves further.** As suggested by Section 3.5, we freeze the early layers of the feature extractor and finetune the rest of the parameters without (`Frozen+Finetuned`) or with our policy-induced consistency objective (`Frozen+PiSCO`). On challenging tasks (*e.g.*, `Hopper`), partial freezing dramatically boosts downstream performance, while `Frozen+PiSCO` further improves upon `Frozen+Finetuned` across the board.

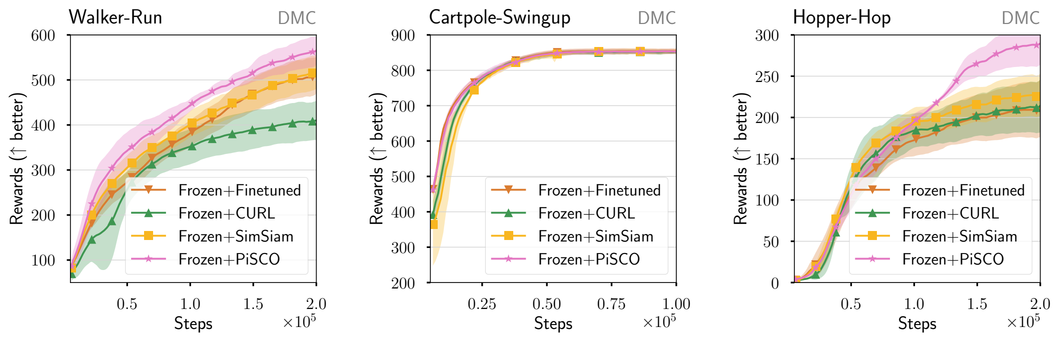


Figure 10: **Policy-induced supervision is a better objective than representation alignment for finetuning.** We compare our policy-induced consistency objective as a measure of representation similarity against popular representation learning objectives (*e.g.* SimSiam, CURL). Policy supervision provides more useful similarities for RL finetuning, which accelerates convergence and reaches higher rewards.

C.1 Computing action values for the layer-by-layer linear probing experiments

We follow a similar protocol for all tasks to obtain action values for layer-by-layer linear probing experiments. As stated in Section 3.5, we first collect $10,000$ pairs using a policy trained (or finetuned) on the downstream task. We then compute the returns for each state-action pair by discounting the rewards observed in the collected trajectories. Then, we fit a 64-hidden unit 2-layer MLP with GELU activations to predict action values from the collected state-action pairs, and store the predicted action-values. On `MSR Jump` and `Habitat` we use the one-hot encoding of the actions rather than their categorical values. Finally, we use the MLP-predicted action values as regression targets for our linear probing models.

C.2 Full results on `DeepMind Control`

As mentioned in Section 3.1, we also ran our analysis on two `DeepMind Control` domains other than `Walker`: `Cartpole` and `Hopper`. Both tasks use the same pretraining and transfer setups as `Walker`. In `Cartpole`, we pretrain on the `Balance` task, where the pole starts in an upright position and our goal is to keep it blanced. We then transfer to `Swingup`, where the pole starts upside down and the goal is to swing and balance it upright. With `Hopper`, the pretraining task is `Stand` where the agent is asked to stand up still; the downstream task is `Hop` where the agent should move in the forward directino as fast as it can. Note that `Hopper-Hop` is considered more challenging than `Cartpole` and `Walker` tasks, and DrQ-v2 fails to solve it even after 3M iterations (Yarats et al., 2022).

For both domains, all analysis results are qualitatively similar to the `Walker` ones from the main text. Figure 7 shows once more that freezing representations is not sufficient for transfer. Figure 8

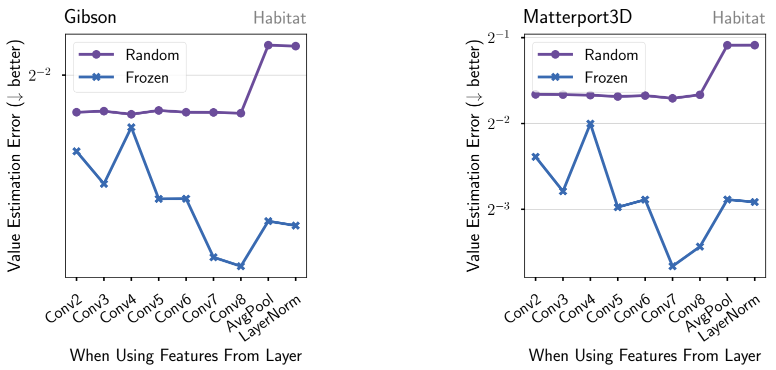


Figure 11: **Identifying which layers to freeze on Habitat tasks.** We replicate the layer-by-layer linear probing experiments on Habitat with the ImageNet pretrained feature extractor. Although the downward trend is less evident than in Figure 4a, we clearly see that layers `Conv8` and `Conv7` yield lowest value prediction error on Gibson and Matterport3D, respectively.

replicates Figure 4a. On both `Cartpole` and `Hopper`, freezing beyond layer `Conv2` increases the value estimation error (top panels); conversely, freezing up to those layers yields the best AURC (bottom panels). Note that on `Cartpole` the degradation in value error is minimal and so is the difference in AURC. We see in Figure 9 that partially freezing the feature encoder yields better transfer results, especially when combined with `PiSCO`. Finally, Figure 10 reconfirms our hypothesis that a policy-driven similarity metric improves RL finetuning.

C.3 Selecting frozen layers in `Habitat`

We decided to freeze up to layer `Conv8` and `Conv7` for the Gibson and Matterport3D experiments in Section 5.1. This section explain how our analysis informs the choice of those layers, without resorting to a finetuned policy nor layer-by-layer freezing experiments as in Figure 4b.

Building on our analysis, we obtained $10,000$ state-action pairs from a policy trained directly on Gibson and Matterport3D. This policy is provided by "Habitat-lab" but in practice a couple of expert demonstrations might suffice. As in Figure 4a, we fit an action value function and save the action values for the observation-action pairs from the collected state-action pairs. Then, we replicate the layer-by-layer linear probing experiment of Section 3.5 with a randomly initialized feature encoder and the one pretrained on ImageNet. Two differences from Section 3.5 stem from the large and visually rich observations on `Habitat`. First, we project each layer's representations to a PCA embedding of dimensions 600 rather than 50[4]; second, we omit the first convolutional layer as PCA is prohibitively slow on such large representations.

Figure 11 displays those results. As expected, the downward trend is less evident than in Figure 4a since the transfer gap between ImageNet and `Habitat` is larger than between tasks in `MSR Jump` or `DeepMind Control`. Nonetheless, we can easily identify the representations in layers `Conv8` and `Conv7` as the most predictive of action value on Gibson and Matterport3D, thus justifying our choice.

C.4 `DeepMind Control` transfer from ImageNet

Given the promising results of large feature extractors pretrained on large amounts of data Figures 1c and 1d, we investigate whether similarly trained feature extractors would also improve transfer on `DeepMind Control` tasks. To that end, we use the same ConvNeXt as on our `Habitat` experiments, freeze its weights, learn policy and action-value heads on each one of the three `DeepMind Control` tasks (replicating the setup of Figure 1).

[4]Dimensions ranging from 500 to 750 work equally well. Smaller dimensions yield underfitting as PCA doesn't retain enough information from the representations. Larger dimensions yield overfitting and pretrained representations don't do much better than random ones.

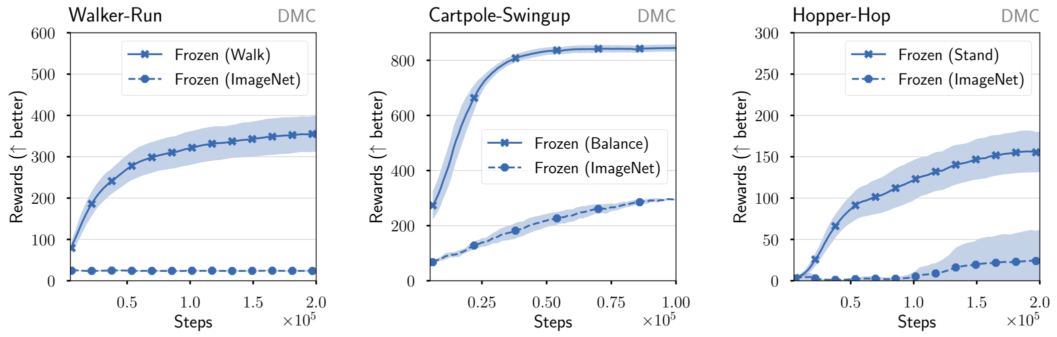


Figure 12: **Pretraining on large and diverse data can hurt transfer when the generalization gap is too large.** When transferring representations that are pretrained on ImageNet to `DeepMind Control` tasks, we see a signficant decrease in convergence rate. We hypothesize this is due to the lack of visual similarity between ImageNet and `DeepMind Control`.

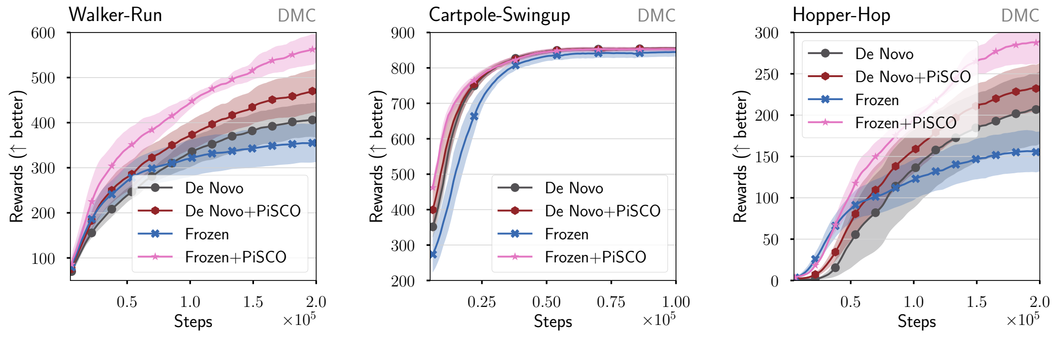


Figure 13: **`PiSCO` can also improves representation learning on source tasks but shines with pretrained representations.** When benchmarking `PiSCO` combined with fully finetunable features (`De Novo+PiSCO`), we observe that it marginally outperforms DrQ-v2 on the downstream tasks (`De Novo`). However, the best performance is obtained when also transferring task-agnostic features (`Frozen+PiSCO`).

Figure 12 provides convergence curves comparing the 4-layer CNN pretrained on the source `DeepMind Control` task (*i.e.*, either `Walker-Walk`, `Cartpole-Balance`, or `Hopper-Stand`) against the ConvNeXt pretrained on ImageNet. The CNN drastically outperforms the ConvNeXt despite having fewer parameters and trained on less data. We explain those results as follows. The smaller network is trained on data that is more relevant to the downstream task, thus has a smaller generalization gap to bridge. In other words, more data is useful insofar as it is relevant to the downstream task. More parameters might help (*i.e.*, pretraining the ConvNeXt on `DeepMind Control` tasks) but it is notoriously difficult to train very deep architectures from scratch with reinforcement learning, as shown with `De Novo` on `Habitat` tasks.

C.5 `De Novo` FINETUNING WITH `PiSCO`

As an additional ablation, we investigate the ability of `PiSCO` to improve reinforcement learning from scratch. In Figure 13, we compare the benefits of using `PiSCO` with when representations are (partly) frozen *v.s.* fully finetunable (`De Novo`). We see that `PiSCO` always improves upon learning from scratch (`De Novo+PiSCO` outperforms `De Novo`) but that it is most beneficial when the task-agnostic features are frozen (`Frozen+PiSCO` outperforms `De Novo+PiSCO`).

C.6 `PiSCO` WITHOUT PROJECTION LAYERS

The original SimSiam formulation includes a projector layer $h(\cdot)$, and Chen & He (2020) show that this projector is crucial to prevent representation collapse. *Does this also holds true for RL?*

To answer this question, we compare the performance of a (partially) frozen feature extractor finetuned with `PiSCO`, with and without a projection layer, in Figure 14. The results clearly answer our question

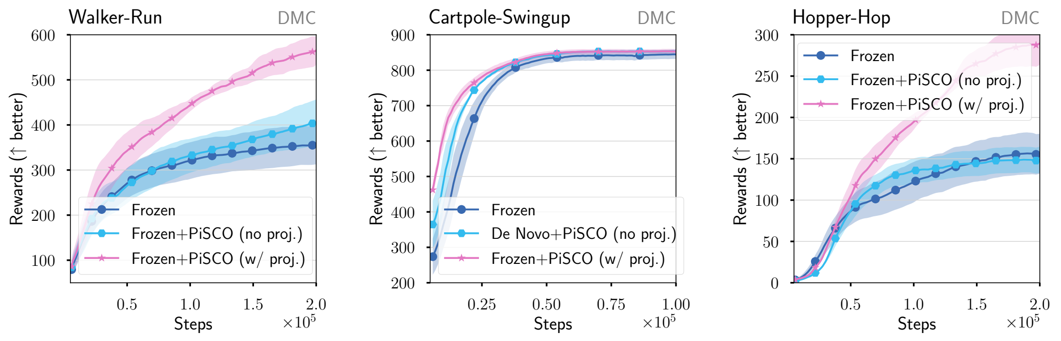


Figure 14: **Is the projector $h(\cdot)$ necessary in `PiSCO`'s formulation?** Yes. Removing projection layer when finetuning task-specific layers (and freezing task-agnostic layers) drastically degrades performance on all `DeepMind Control` tasks.

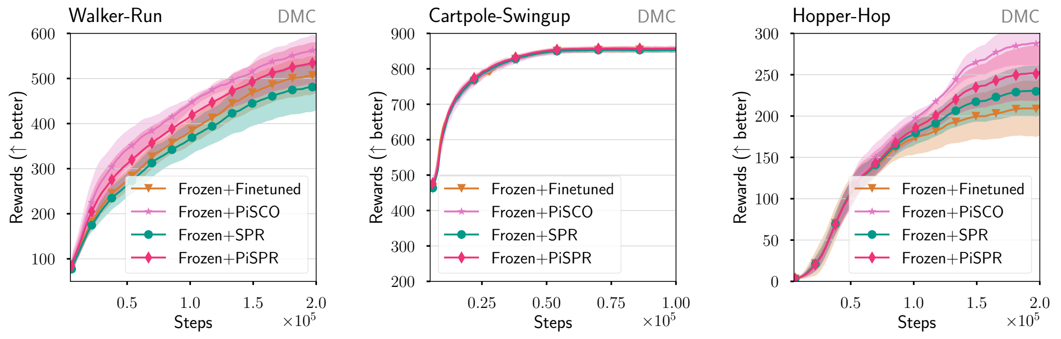


Figure 15: **Combining `SPR` with `PiSCO` significantly improves the performance of `SPR`.** Swapping the cosine similarity objective in `SPR` (Schwarzer et al., 2021) for the policy-induced objective suggested by our analysis significantly improves finetuning. Still, finetuning with `PiSCO` (based on SimSiam (Chen & He, 2020)) yields the best performance, while remaining easier to implement and faster in terms of wall-clock time.

in the affirmative: the projector is also required and removing it drastically degrades performance on the downstream task.

C.7 Comparing and combining with SPR

To illustrate future avenues for extending the ideas motivating `PiSCO`, we combine them with the self-predictive representation (`SPR`) objective of Schwarzer et al. (2021). We replace SimSiam for `SPR` as the auxiliary self-supervised objective used during finetuning. Concretely, reimplemented `SPR` for `DeepMind Control` tasks and compare the original implementation (which uses cosine similarity $\cos(p_1, p_2)$ to compare representations) against a variant (`PiSPR`) which uses the policy-induced objective $\frac{1}{2}\left(\mathcal{D}(p_1, p_2)) + \mathcal{D}(p_2, p_1)\right)$[5].

In terms of implementation details, we found that setting the number of contiguous steps sampled from the replay buffer to $K = 3$ yielded the best results on `DeepMind Control` tasks. The transition model is a 2-layer MLP (1024 hidden units) with layer normalization and ReLU activations. We used the same projector as as for SimSiam (which does not reduce dimensionality), and a linear predictor with layer-normalized inputs.

Figure 15 reports the results for partially frozen feature extractors with different finetuning objectives. We observe that using `SPR` typically performs as well as finetuning the task-specific layers with the RL objective. In that sense, finetuning with `PiSCO` offers an compelling alternative: it is substantially cheaper ($\tilde{3}$x faster in wall-clock time), doesn't require learning a transition model nor keeping track of momentum models, and is simpler to tune (fewer hyper-parameters). More interestingly, we find that replacing the cosine similarity in `SPR` for the policy-induced objective (*i.e.*, `PiSPR`) significantly improves performance over `SPR`. Those results further validate the generality of our analysis insights.

[5]Note: p_1 and p_2 correspond to $\hat{y}_{t+k}$ and $\tilde{y}_{t+k}$ in the original `SPR` notation.

D Pseudocode

Here we provide pseudocode for implementing PiSCO on top of two popular reinforcement learning algorithms. Algorithm 1 adds PiSCO to PPO (Schulman et al., 2017), while Algorithm 2 adds it to DrQ-v2 (Yarats et al., 2022).

Algorithm 1 PPO with PiSCO

```
# sample transitions from replay
$s$, $a$, $r$, $s'$ = replay.sample(batch_size)
$z$ = $\phi(s)$

# compute value loss
$\mathcal{L}^V$ = 0.5 · ($V(z)$ - discount($r$))$^2$

# compute policy loss
$\Delta_\pi$ = $\log \pi(a \mid z) - \log \pi_{\texttt{old}}(a \mid z)$
$A$ = GAE($V(z)$, $r$, $\gamma$, $\tau$)
$\mathcal{L}^\pi$ = $\min(\exp(\Delta_\pi) \cdot A,$ clip$(\exp(\Delta_\pi) \cdot A, 1-\epsilon, 1+\epsilon)$

# compute PiSCO loss
$z_1$, $z_2$ = $\phi$(data_augment($s$)), $\phi$(data_augment($s$))
$p_1$, $p_2$ = $h(z_1)$, $h(z_2)$
$\mathcal{L}^{\texttt{PiSCO}}$ = 0.5 · KL($\perp(\pi(\cdot \mid z_1)) \| \pi(\cdot \mid p_2)$) + 0.5 · KL($\perp(\pi(\cdot \mid z_2)) \| \pi(\cdot \mid p_1)$)

adam.optimize($\mathcal{L}^\pi + \nu \cdot \mathcal{L}^V + \lambda \cdot \mathcal{L}^{\texttt{PiSCO}} - \beta \cdot H(\pi(\cdot \mid z))$)  # optimizes $V$, $\pi$, $h$, and $\phi$
```

Algorithm 2 DrQ-v2 with PiSCO

```
# sample transitions from replay
$s$, $a$, $r$, $s'$ = replay.sample(batch_size, n_steps, $\gamma$)

# compute policy loss
$z$ = $\perp$($\phi$(data_augment($s$)))
$\hat{a}$ = $\pi(\cdot \mid z)$.rsample()
$\mathcal{L}^\pi$ = - $\min$($Q_1(z, \hat{a})$, $Q_2(z, \hat{a})$)
adam.optimize($\mathcal{L}^\pi$)  # only optimizes $\pi$

# compute action-value loss
$z$ = $\phi$(data_augment($s$))
$z'$ = $\phi$(data_augment($s'$))
$a'$ = $\pi(\cdot \mid z')$.sample()
$q_1$, $q_2$ = $Q_1(z, a)$, $Q_2(z, a)$
$q'$ = $\perp$($r$ + $\gamma$ · $\min$($Q_1(z', a')$, $Q_2(z', a')$))
$\mathcal{L}^Q$ = 0.5 · ($q_1$ - $q'$)$^2$ + 0.5 · ($q_2$ - $q'$)$^2$

# compute PiSCO loss
$z_1$, $z_2$ = $\phi$(data_augment($s$)), $\phi$(data_augment($s$))
$p_1$, $p_2$ = $h(z_1)$, $h(z_2)$
$\mathcal{L}^{\texttt{PiSCO}}$ = 0.5 · KL($\perp(\pi(\cdot \mid z_1)) \| \pi(\cdot \mid p_2)$) + 0.5 · KL($\perp(\pi(\cdot \mid z_2)) \| \pi(\cdot \mid p_1)$)

adam.optimize($\mathcal{L}^Q + \lambda \cdot \mathcal{L}^{\texttt{PiSCO}}$)  # only optimizes $Q_1$, $Q_2$, $h$, and $\phi$
```

E Code Release

Together with our submission, we release example code to implement PiSCO in the `code/` directory of the Supp. Material. We plan to publicly release complete implementations, together with examples for the pretrain-and-transfer pipeline. We describe the main files below.

- `msr_jump.py`: our GPU implementation of `MSR Jump` tasks.
- `ppo_pisco.py`: our PyTorch (Paszke et al., 2019) implementation of `PiSCO` on top of PPO.
- `drqv2_pisco.py`: our PyTorch (Paszke et al., 2019) implementation of `PiSCO` on top of DrQ-v2.