Fast Adaptation with Behavioral Foundation Models

Harshit Sikchi, Andrea Tirinzoni, Ahmed Touati, Yingchen Xu, Anssi Kanervisto, Scott Niekum, Amy Zhang, Alessandro Lazaric, Matteo Pirotta

Keywords: Unsupervised Learning, Self Supervised learning, Zero shot RL, Adaptation, Finetuning

Summary

Unsupervised zero-shot reinforcement learning (RL) has emerged as a powerful paradigm for pretraining behavioral foundation models (BFMs), enabling agents to solve a wide range of downstream tasks specified via reward functions without additional test-time learning or planning. This is achieved by learning self-supervised task embeddings alongside corresponding near-optimal behaviors, and incorporating an inference procedure to directly retrieve the latent task embedding and associated policy for any given reward function. In this work, we demonstrate that existing unsupervised zero-shot RL pre-training methods discover a latent task embedding space containing more performant policies than those identified by their inference procedure, making them well-suited for fast adaptation. Motivated by this observation, we propose both actor-critic and actor-only fast adaptation strategies that search in the low-dimensional task-embedding space of the pre-trained BFM to rapidly improve the performance of its zero-shot policies on any downstream task. Notably, our approach mitigates the initial "unlearning" phase commonly observed when fine-tuning pre-trained RL models. We evaluate our fast adaptation strategies on top of four state-of-the-art zero-shot RL methods in multiple navigation and locomotion domains. Our results show that they achieve 10-40% improvement over their zero-shot performance in only a few episodes, outperforming existing baselines.

Contribution(s)

1. We empirically investigate the task-representation space learned by a family of unsupervised zero-shot RL methods and show that it contains policies achieving significantly higher returns than the one output by the zero-shot inference procedure.
   Context: Prior works in zero-shot RL (Touati et al., 2023; Park et al., 2024; Agarwal et al., 2024) implicitly assume that zero-shot inference is the optimal way to prompt a pre-trained model for behaviors optimizing tasks specified by reward functions. We challenge such an assumption and show that this is not the case.
2. We propose two fast-adaptation algorithms: a) Residual Latent Adaptation (ReLA), an approach that optimizes for a policy in the BFM's task-representation space by training an additional smaller critic to estimate the cumulative reward not captured by the pre-trained BFM. b) Lookahead Latent Adaptation (LoLA), a computationally efficient approach that leverages policy gradients with lookahead returns without updating the pre-trained critic.
   Context: Prior approaches to adaptation either fine-tune the entire pre-trained critic and perform policy optimization in the action space (Nair et al., 2020; Nakamoto et al., 2023), or learn policy residuals (Silver et al., 2018; Johannink et al., 2019; Rana et al., 2023).
3. We evaluate our approaches on top of four state-of-the-art zero-shot RL methods in multiple navigation and locomotion domains, and show that they achieve 10-40% improvement over their zero-shot performance. Furthermore, we observe that our approach LoLA avoids the initial "unlearning" phase commonly observed in the literature.
   Context: Prior approaches for fine-tuning RL models without retaining training data (Luo et al., 2023; Zhou et al., 2024) observe a sharp decrease in performance due to distribution shift.

Fast Adaptation with Behavioral Foundation Models

Harshit Sikchi[1,†], **Andrea Tirinzoni**[2], **Ahmed Touati**[2], **Yingchen Xu**[2], **Anssi Kanervisto**[2], **Scott Niekum**[3], **Amy Zhang**[1], **Alessandro Lazaric**[2], **Matteo Pirotta**[2]
hsikchi@utexas.edu, {lazaric,pirotta}@meta.com

[1]**The University of Texas at Austin**
[2]**FAIR at Meta**
[3]**UMass Amherst**

[†] Work done during an internship at FAIR, Meta

Abstract

Unsupervised zero-shot reinforcement learning (RL) has emerged as a powerful paradigm for pretraining behavioral foundation models (BFMs), enabling agents to solve a wide range of downstream tasks specified via reward functions in a zero-shot fashion, i.e., without additional test-time learning or planning. This is achieved by learning self-supervised task embeddings alongside corresponding near-optimal behaviors and incorporating an inference procedure to directly retrieve the latent task embedding and associated policy for any given reward function. Despite promising results, zero-shot policies are often suboptimal due to errors induced by the unsupervised training process, the embedding, and the inference procedure. In this paper, we focus on devising *fast adaptation* strategies to improve the zero-shot performance of BFMs in few steps of online interaction with the environment, while avoiding any performance drop during the adaptation process. Notably, we demonstrate that existing BFMs learn a set of skills containing more performant policies than those identified by their inference procedure, making them well-suited for fast adaptation. Motivated by this observation, we propose both actor-critic and actor-only fast adaptation strategies that search in the low-dimensional task-embedding space of the pre-trained BFM to rapidly improve the performance of its zero-shot policies on any downstream task. Notably, our approach mitigates the initial "unlearning" phase commonly observed when fine-tuning pre-trained RL models. We evaluate our fast adaptation strategies on top of four state-of-the-art zero-shot RL methods in multiple navigation and locomotion domains. Our results show that they achieve 10-40% improvement over their zero-shot performance in a few tens of episodes, outperforming existing baselines.

1 Introduction

Unsupervised (or self-supervised) pre-training has emerged as one of the key ingredients behind the recent breakthroughs in computer vision and language modeling (e.g., Radford et al., 2019; Devlin et al., 2019; Touvron et al., 2023; Caron et al., 2021). This technique allows utilizing large datasets of unlabeled data samples to learn generalizable representations that can be later fine-tuned for various downstream applications (Zhai et al., 2023; Brown et al., 2020; Driess et al., 2023). For instance, language models are pre-trained on internet-scale data with a next-token prediction objective and later fine-tuned for desired applications using high-quality examples. How to transpose this approach to reinforcement learning (RL) to train agents that can efficiently solve sequential decision-making problems is an open research question of paramount importance. Going beyond the

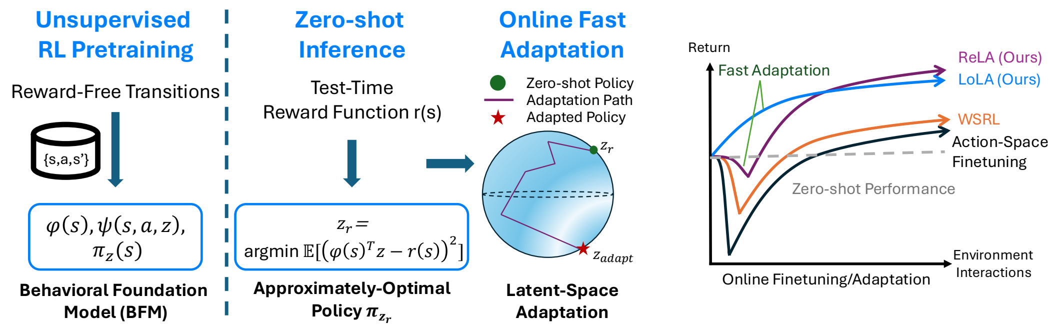

Figure 1: **Overview of our method**: Unsupervised zero-shot RL methods provide us with an initial policy π_{z_r}; we propose a way to leverage the latent space of learned policies as well as the pre-trained critic to rapidly adapt and improve π_{z_r} on few task-specific environment interactions. Right: Illustrative summary of our results.

tabula-rasa paradigm of classic RL requires an unsupervised pre-training objective and the ability to efficiently fine-tune or adapt pre-trained representations for downstream tasks. Recent developments in unsupervised RL propose various objectives to learn a repertoire of skills on top of reward-free data from the environment (Gregor et al., 2016; Wu et al., 2018; Hansen et al., 2019; Liu & Abbeel, 2021; Eysenbach et al., 2018; Zahavy et al., 2022; Park et al., 2023). Some of these methods are named "zero-shot", in the sense that they additionally provide a procedure to infer a performant policy for any given task specified by reward functions (Touati et al., 2023; Park et al., 2024; Agarwal et al., 2024; Cetin et al., 2024), demonstrations (Pirotta et al., 2024; Tirinzoni et al., 2025), or videos/language (Sikchi et al., 2024). The resulting pre-trained agents are commonly referred to as Behavioral Foundation Models (BFMs, Pirotta et al., 2024; Tirinzoni et al., 2025).

Zero-shot methods commonly pre-train two components: (1) a *state representation* $\varphi : \mathcal{S} \to \mathbb{R}^d$ that embeds state observations $s \in \mathcal{S}$ into a d-dimensional vector $\varphi(s)$, and (2) a space $\{\pi_z\}$ of *policies* parameterized by a latent vector $z \in \mathbb{R}^d$. The representation φ defines the set of all linear reward functions in φ, i.e., $\tilde{r}_z(s) = \varphi(s)^T z$ for all $z \in \mathbb{R}^d$, which in turn is used as a *self-supervised* objective function for the policy space: for each $z \in \mathbb{R}^d$, the policy π_z is trained to be approximately optimal for the reward $\tilde{r}_z$. Given a reward function $r(s)$ at test time, a zero-shot policy π_{z_r} can be obtained by projecting r onto the pre-trained state features φ through linear regression on top of the training data, hence approximating $r(s) \simeq \varphi(s)^T z_r$.

Although this inference method has proven effective in producing reasonable policies, it suffers from two main limitations yielding sub-optimal performance. First, the embedding φ is learned using unsupervised losses encoding inductive biases[1] that may not be suitable for the downstream tasks of interest. As a result, the projection of the reward function onto φ may remove crucial aspects of the task specification thus preventing from finding the optimal policy for the original reward. In an extreme scenario, if a reward function lies in the orthogonal subspace of the features' linear span, its projection onto these features becomes zero, making it uninformative. Second, BFMs are typically trained on task-agnostic datasets that may have poor coverage of the rewarding states relevant to the specific task. This limitation can result in zero-shot inference failing to accurately represent these states and ultimately hinder the learning of a good policy.

While the suboptimality of unsupervised pre-training of large models is somewhat unavoidable, it is natural to wonder whether these limitations can be overcome once a downstream reward function is given and the agent has online access to the environment. In this paper we focus on devising *fast adaptation* strategies that improve zero-shot performance of BFMs **1)** *rapidly*, i.e., in a handful of online episodes, and **2)** *monotonically*, i.e., avoiding any performance drop during the adaptation process. This motivates the main question of this work:

[1]For instance, some methods rely on low-rank assumptions in the policy dynamics (Touati & Ollivier, 2021; Agarwal et al., 2024), while others focus only on goal-reaching behaviors (Park et al., 2024)

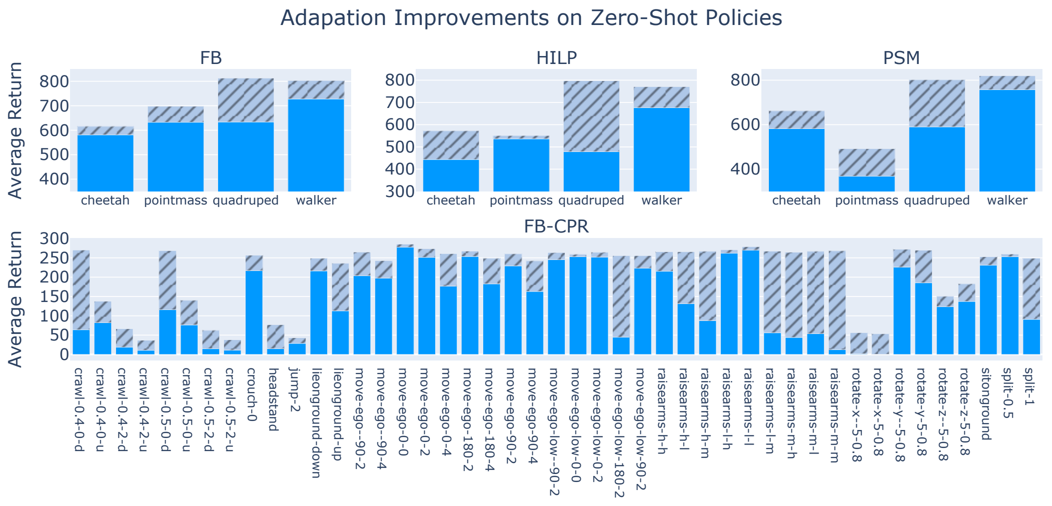

Figure 2: Performance comparison of zero-shot policy vs adapted policy in the BFM's latent space after 200 episodes. The shaded region shows the improvement of the adapted policies averaged across tasks.

Does the policy space of a pre-trained BFM contain better behaviors than those returned by zero-shot inference? If so, can we retrieve them with few task-specific environment interactions?

To address this question, we propose searching over the latent space $\mathcal{Z}$ using a limited number of online task-specific interactions with the environment (cf. Figure 1). We introduce two algorithms that leverage the latent space and pre-trained components from BFMs to enable *fast adaptation* of their zero-shot policies: (1) Residual Latent Adaptation (ReLA), an off-policy actor-critic approach that trains a small *residual critic* to compensate for the reward projection errors, and Lookahead Latent Adaptation (LoLA), a hybrid actor-only approach that combines on-policy optimization while bootstrapping the frozen critic from the pre-trained BFMs.

We perform an extensive empirical evaluation on 5 domains with a total of 64 tasks spanning low-dimensional and high-dimensional problems with increasing complexity, including a whole-body humanoid control problem with a wide range of 45 diverse reward-based behaviors. We demonstrate the effectiveness of our proposed algorithms on four state-of-the-art BFMs: FB (Touati & Ollivier, 2021), HILP (Park et al., 2024), PSM (Agarwal et al., 2024) and FB-CPR (Tirinzoni et al., 2025). In particular, we answer the above question affirmatively: our fast adaptation algorithms achieve 10-40% improvement over the BFMs zero-shot performance in only a few episodes (Figure 2 and 3), while outperforming existing baselines. Moreover, we show that LoLA avoids any initial drop of performance, a phenomenon commonly observed by numerous prior works on fine-tuning RL policies (Nair et al., 2020; Nakamoto et al., 2023; Luo et al., 2023; Zhou et al., 2024).

2 Preliminaries

Markov decision process. We consider a reward-free Markov decision process (MDP) (Puterman, 2014; Sutton & Barto, 2018) which is defined as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, d_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ respectively denote the state and action spaces, P denotes the transition kernel with $P(s'|s,a)$ indicating the probability of transitioning from s to s' by taking action a, d_0 denotes the initial state distribution and $\gamma \in (0,1)$ specifies the discount factor. A policy π is a function $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ mapping a state s to probabilities of action in $\mathcal{A}$. We denote by $\Pr(\cdot \mid s, a, \pi)$ and $\mathbb{E}[\cdot \mid s, a, \pi]$ the probability and expectation operators under state-action sequences $(s_t, a_t)_{t \geq 0}$ starting at (s,a) and following policy π with $s_t \sim P(\cdot \mid s_{t-1}, a_{t-1})$ and $a_t \sim \pi(\cdot \mid s_t)$. Given any reward function $r : \mathcal{S} \to \mathbb{R}$, the Q-function of π for r is $Q_r^\pi(s,a) := \sum_{t \geq 0} \gamma^t \mathbb{E}[r(s_{t+1}) \mid s, a, \pi]$.

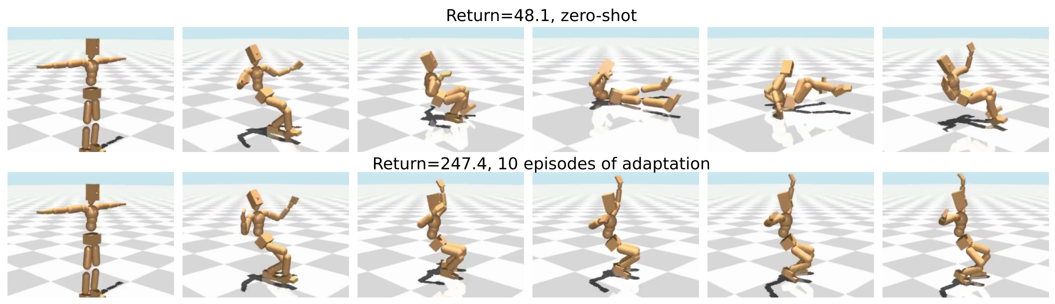

Figure 3: Qualitative difference in behaviors in 10 episodes of adaptation in HumEnv environment for the task move-ego-low-180-2 with our method LoLA.

Successor measures and features. The *successor measure* (Dayan, 1993; Blier et al., 2021) of state-action (s, a) under a policy π is the (discounted) distribution of future states obtained by taking action a in state s and following policy π thereafter:

$$M^\pi(X \mid s, a) := \sum_{t \geq 0} \gamma^t \Pr(s_{t+1} \in X \mid s, a, \pi) \quad \forall X \subset \mathcal{S}. \tag{1}$$

Importantly, successor measures disentangle the dynamics of the MDP and the reward function: for any reward r and policy π, the Q-function can be expressed linearly as $Q^\pi_r = M^\pi r$.

Given a feature map $\varphi : \mathcal{S} \to \mathbb{R}^d$ that embeds states into a d-dimensional space, the *successor features* (Barreto et al., 2017) is the expected discounted sum of features:

$$\psi^\pi(s, a) := \sum_{t \geq 0} \gamma^t \mathbb{E}[\varphi(s_{t+1}) \mid s, a, \pi]. \tag{2}$$

Successor features and measures are related: by definition, $\psi^\pi(s, a) = \int_s M^\pi(\mathrm{d}s' \mid s, a)\varphi(s')$. For any reward function in the linear span of φ, *i.e.*, $r(s) = \omega^\top \varphi(s)$ where ω is a weight vector in $\mathbb{R}^d$, the Q-function can be expressed compactly as $Q^\pi_r(s, a) = \omega^\top \psi^\pi(s, a)$.

Behavioral foundation models. A *behavioral foundation model*, for a given MDP, is an agent that can be trained in unsupervised fashion using reward-free transitions and yet can produce approximately optimal policies for a large class of reward functions r specified at test time, without performing additional learning or planning. In this work, we focus on zero-shot RL agents that are based on successor features and forward-backward representations.

Universal successor features (USFs) (Borsa et al., 2018) provide a generic framework for zero-shot RL. Given a feature map φ, USFs learn the successor features of a particular family of policies π_z parameterized by latent variables $z \in \mathcal{Z} \subset \mathbb{R}^d$:

$$\psi(s, a, z) = \mathbb{E}[\sum_{t \geq 0} \gamma^t \varphi(s_{t+1}) \mid s, a, \pi_z], \quad \pi_z(s) = \arg\max_a \psi(s, a, z)^\top z. \tag{3}$$

At test time, once a reward function r is specified, a reward-maximizing policy is inferred by performing a linear regression of r onto the features φ. In particular, we estimate $z_r = \arg\min_z \mathbb{E}_{s\sim\rho}[(r(s) - \varphi(s)^\top z)^2] = \mathbb{E}_{s\sim\rho}[\varphi(s)\varphi(s)^\top]^{-1}\mathbb{E}_{s\sim\rho}[\varphi(s)r(s)]$ where ρ is some dataset distribution over states. Then we return the pre-trained policy π_{z_r}. This policy is guaranteed to be optimal if the reward is in the linear span of the features φ (Borsa et al., 2018). Although USF is a generic framework, it requires specifying a training criterion to learn the basic features φ. Touati et al. (2023) compare several choices of unsupervised representation learning objectives across various empirical problems. In this work, we focus on two recent state-of-the-art feature learning methods for zero-shot RL: *Hilbert representations* (HILP) (Park et al., 2024) and *proto successor measures* (PSM) (Agarwal et al., 2024). HILP constructs features φ such that the distance $\|\varphi(s)-\varphi(s')\|$

between a state pair (s, s') encodes the optimal value function of reaching the state s' starting at s. PSM proposes to build the features φ by learning an *affine decomposition* of the successor measure for a discrete codebook of policies, *i.e.*, $M^{\pi_u}(\mathrm{d}s' \mid s, a)/\rho(\mathrm{d}s') \approx \phi(s, a)^\top \varphi(s')w(u) + b(s, a, s')$, where ϕ, w and b are vector-valued functions and where π_u is a deterministic policy that outputs an action in state s as a realization of the uniform distribution, determined by the random seed u.

Forward-backward representations (FB) (Touati & Ollivier, 2021) provide an alternative framework for zero-shot RL. Unlike USFs which use two separate criteria to learn features and their successor features, FB avoid the state featurization step and employ a single objective to learn a finite-rank decomposition of the successor measure for various policies. Namely, FB pre-train two representations $F : \mathcal{S} \times \mathcal{A} \times \mathcal{Z} \to \mathbb{R}^d$ and $B : \mathcal{S} \to \mathbb{R}^d$ such that:

$$F(s, a, z)^\top B(s')\rho(\mathrm{d}s') \approx M^{\pi_z}(\mathrm{d}s' \mid s, a), \quad \pi_z(s) = \arg\max_a F(s, a, z)^\top z. \tag{4}$$

FB representations are related to USFs, as $F(s, a, z)$ represents the successor features of $\mathbb{E}_{s\sim\rho}[B(s)B(s)^\top]^{-1}B(s)$ (Touati et al., 2023). In the sequel, to standardize the notations with the USFs, we will denote $\psi(s, a, z) = F(s, a, z)$ and $\varphi(s) = \mathbb{E}_{s\sim\rho}[B(s)B(s)^\top]^{-1}B(s)$.

Forward-Backward representations with Conditional Policy Regularization (FB-CPR) (Tirinzoni et al., 2025) is an online variant of FB that grounds the unsupervised policy learning toward imitating observation-only unlabeled behaviors.

3 Fast Adaptation for Behavioral Foundation Models

In this section, we introduce our two approaches for fast adaptation of pre-trained BFMs: an off-policy actor-critic algorithm (Section 3.1), and a hybrid on-policy actor-only algorithm (Section 3.2).

3.1 ReLA: Residual Latent Adaptation

Given a reward function r, ReLA begins with the latent variable $z = z_r$ inferred by the zero-shot procedure and uses an off-policy actor-critic approach to gradually update z towards better performance. The overall algorithm uses a standard online training procedure, interleaving between critic and actor updates (as described below), while gathering reward-labeled transitions in a replay buffer $\mathcal{D}_{\text{online}}$ through online interactions with the environment.

Residual critic learning. Instead of training a critic from scratch to model the Q-function of the policy π_z currently being learned for the reward r, ReLA uses a residual critic to correct for the reward projection error. This is made possible by the following decomposition:

$$\begin{aligned} Q_r^{\pi_z}(s, a) &= Q^{\pi_z}_{\varphi^\top z_r}(s, a) + Q^{\pi_z}_{r-\varphi^\top z_r}(s, a) \\ &= \psi(s, a, z)^\top z_r + Q^{\pi_z}_{r-\varphi^\top z_r}(s, a) \end{aligned} \tag{5}$$

where the last equality holds because ψ is pre-trained to estimate the successor features of φ and the projected reward $\varphi^\top z_r$ lies in the span of φ. Consequently, ReLA considers a network $Q^{\text{residual}}(s, a; \theta)$ parametrized by weights θ and trains it via off-policy TD learning so that $\psi(s, a, z_r)^\top z_r + Q^{\text{residual}}(s, a; \theta)$ approximates the Q-function $Q_r^{\pi_z}(s, a)$, while keeping the *base Q-function* $\psi(s, a, z_r)^\top z_r$ frozen. In practice, we shall use much smaller networks for the residual critic than for the pre-trained successor features, with the main intuition being that we only need to compensate for some projection error. For a more in-depth treatment of the Q-function decomposition we refer the readers to Appendix 7.1.

Latent actor update. ReLA updates the latent variable z using standard policy-gradient ascent, with the key difference being that the gradient is computed only with respect to z, while keeping the pre-trained actor parameters fixed,

$$\nabla_z \mathbb{E}_{s\sim\mathcal{D}_{\text{online}}}[\psi(s, \pi_z(s), z_r)^\top z_r + Q^{\text{residual}}(s, \pi_z(s); \theta)], \tag{6}$$

The main advantage over optimizing the whole actor network is that we only need to search in a low-dimensional space (in practice, z has in the order of hundreds of components, while the actor network of a BFM has in the order of millions of parameters).

3.2 LoLA: Lookahead Latent Adaptation

Although ReLA can take advantage of off-policy data collected in the replay buffer, it requires learning an additional residual network. Therefore, ReLA demands a certain budget of transitions and updates to mitigate the distribution shift issue (Luo et al., 2023) when learning the Q-function, which may impede improvements during the very early stages of adaptation. On the other hand, a purely on-policy approach will require rolling out entire trajectories under the current policy to estimate Monte Carlo returns $\sum_{t=0}^{T} \gamma^t r_t$ (where T is the episode length), and thus incur many environment interactions in the process. Alternatively, we propose **Lo**okahead **L**atent **A**daptation algorithm (LoLA) that uses fixed-horizon on-policy rollouts with a frozen terminal value function obtained from the BFM. LoLA parameterizes a policy over the latent space as a normal distribution $\pi_{\mu,\sigma} = \mathcal{N}(\mu, \sigma)$ with trainable mean μ (initialized with $\mu = z_r$), and fixed diagonal covariance σ. The pre-trained successor features from BFM are used to compute the estimate of a terminal value-function, thus estimating the n-step lookahead return of policy π_z starting from state s_0 as $R^n(s_0, z) = \sum_{t=0}^{n-1} \gamma^t r(s_{t+1}) + \gamma^n \psi(s_{n+1}, \pi_z(s_{n+1}), z)^\top z_r$.

Moreover, to further improve learning, LoLA incorporates the variance reduction strategy of leave-one-out baseline (Kool et al., 2019). This baseline has recently been shown to be empirically effective for fine-tuning large language models (Ahmadian et al., 2024). This leads to the following final gradient estimate[2]:

$$\mathbb{E}_{s_0 \sim \nu} \left[\frac{1}{k} \sum_{i=1}^{k} \left(R(s_0, z_i) \; - \; \frac{1}{k-1} \sum_{\substack{j=1 \\ j \neq i}}^{k} R(s_0, z_j) \right) \nabla_\mu \log \pi_{\mu,\sigma}(z_i) \right] \quad \text{for} \quad z_1, \ldots, z_k \sim \pi_{\mu,\sigma}(\cdot) \tag{7}$$

where s_0 is sampled from the distribution ν defined as mixture between the environment's initial distribution d_0 and the online replay buffer distribution $\mathcal{D}_{\text{online}}$. For each sampled starting state s_0, we sample k latent variables $\{z_i\}_{i \in [k]} \sim \pi_{\mu,\sigma}$ and generate k trajectories of length n $(s_0^{(i)}, a_0^{(i)}, s_1^{(i)}, a_1^{(i)}, \ldots, s_n^{(i)})$ by following the policy π_{z_i}. Computing the gradient requires the ability to reset of any state in support of distribution ν, which includes the states encountered during online adaptation.

4 Experimental Results

The goal of our experiments is to study how well latent policy adaptation works on top of existing BFMs. We perform several ablations to understand the efficacy of the proposed methods and evaluate our design choices. Precisely, *1)* Can we find better policies by online latent policy adaptation compared to the zero-shot policies? Or, equivalently, is the latent policy space easy to search over? *2)* How important is to leverage BFMs properties (e.g., Q-function estimate, zero-shot policy initialization)? *3)* What are the critical limitations of the zero-shot inference process?

Experimental setup. We investigate these questions by leveraging 4 different BFMs: FB, HILP, PSM and FB-CPR. While FB, HILP and PSM are trained offline, FB-CPR learns through online environmental interactions and it is regularized towards expert trajectories. We consider four environments from the DeepMind Control suite (Tassa et al., 2018) and train the BFMs on an exploratory dataset obtained from ExoRL (Yarats et al., 2022).[3] Further, we leverage the FB-CPR model released by Tirinzoni et al. (2025) for the `HumEnv` environment, a high-dimensional humanoid agent. Overall, we consider 7 tasks for `Pointmass`, 4 for `Cheetah`, 4 for `Quadruped`, 4 for `Walker`, and 45 tasks for `HumEnv`. Detailed information about the pre-training phase can be found in Appendix 9.

[2] In practice we work with z normalized on hypersphere using projected gradient descent

[3] We consider the dataset collected by running RND (Burda et al., 2019).

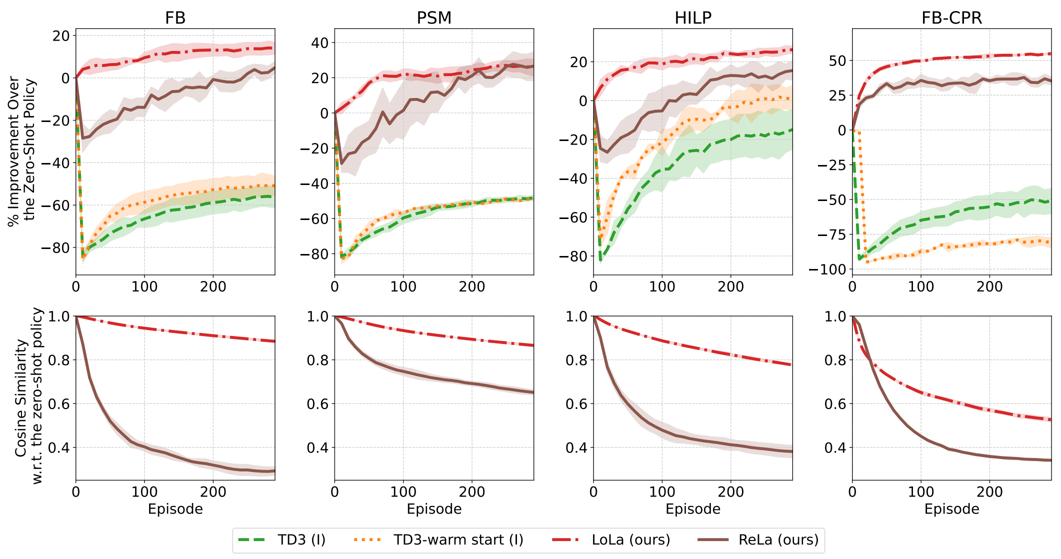

Figure 4: **Top**: Performance improvement w.r.t. the zero-shot policy for different online fast adaptation methods and BFMs. TD3(I) denotes standard action-based TD3 with zero-shot policy initialization, our methods are as described in Section 3. **Bottom**: Cosine similarity between the zero-shot policy z_r and the learned policy z for the methods working in the latent policy space. We report mean and standard deviation over 5 seeds. Results are averaged over 19 tasks for FB, PSM, HILP and 45 tasks for FB-CPR.

Protocol and baselines. While the paper focuses on adaptation in the latent policy space, we also investigate the common class of approaches for fine-tuning in action space (i.e., updating all policy network parameters) with zero-shot initialization (Nair et al., 2020; Nakamoto et al., 2023). In particular, we consider a TD3-based algorithm (Fujimoto et al., 2018) that train a critic from scratch and an actor initialized using the zero-shot policy (TD3 (I)).[4] Since collecting a few on-policy trajectories before starting updating the critic and the actor proved to be effective for offline to online adaptation, a strategy called Warm-Start RL (WSRL) (Zhou et al., 2024), we additionally consider this component for action-based algorithms. We further ablate several design choices (e.g., zero-shot initialization, bootstrapped critic) in Section 4.2 and, in Appendix 10 we report variations of our algorithms that operate by directly updating the parameters of the policy. See Table 1 for a complete list of algorithm variations.

We use a comparable architecture and hyperparameter search for all algorithms. For each BFM, we report the performance on the set of hyperparameters that performed best across all tasks and domains. We train all the online adaptation algorithms for 300 episodes and we use 5 seeds for each experiment. Evaluation is done by averaging results over 50 episodes. We also use TD3 as base off-policy algorithm for implementing `ReLA`. When using residual critic we use a small 2-layers MLP with hidden dimension 64, while when we learn the critic from scratch we use a 2-layers MLP with hidden dimension 1024. The policy has always the same size as the BFM policy. We provide further implementation details in Appendix 9.

4.1 Do ReLA and LoLA enable fast adaptation?

Figure 4 (*top*) shows our aggregated results across tasks for each domain: Latent policy adaptation leads to performance improvements w.r.t. the zero-shot policy in the range of 10-30% for DMC environment and 40-50% for HumEnv. Compared to Figure 2, these results show that significant improvements are already obtained in few online episodes. For example, `LoLA` leads to about 10% (resp. 40%) improvement for DMC (resp. for HumEnv) in only 20 episodes. These results show that

[4] We also tested vanilla RLOO (Kool et al., 2019) but did not get good results and decided not to report it.

i) the space of policies learned by the BFMs contains better policies than the one inferred by the zero-shot procedure and *ii)* such a space can be easily navigated using gradient-based approaches. While both ReLA and LoLA provide significant performance improvements, LoLA is the only method to achieve monotonic performance improvement across the board. As we can see from the per-task visualization in Appendix 10, the non-monotonic performance of ReLA is mostly due to the fact that the methods incurs a noticeable catastrophic forgetting in the pointmass environment where TD3-based methods seem to struggle in the online setting, probably due to exploration issues. As a result of training purely on online samples, critic learning in ReLA undergoes distribution shift which has been investigated to lead to initial unlearning (Zhou et al., 2024) whereas LoLA skips the critic learning step entirely. In addition, LoLA exploits a privileged information compared to ReLA, the ability to reset the environment to any arbitrary state, which further contributes in stabilizing and speeding up the learning process (see e.g., Mhammedi et al., 2024).

How does the adapted policy evolve in latent space? To try to better understand the learning dynamics of ReLA and LoLA we report the cosine similarity between the adapted z and the zero-shot policy z_r in Figure 4 (*bottom*). ReLA deviates much more in the latent space from the initial zero-shot policy than LoLA. This fast and significant change is associated with the drop in performance. On the other hand, despite the high learning rate (we found 0.1 or 0.05 to be the best based on the BFM), LoLA remains closer to the zero-shot policy. A potential cause for the significant change in ReLA may be difficulties in critic learning associated with distribution shift, which can impact policy directly. This visualization also shows that while converging to different policies, the performance of ReLA and LoLA is comparable after 300 episodes in the DMC environments. This reveals that policies with similar performance may be associated to with different latent vectors z.

Algorithm	Zero-Shot Policy Init.	Residual Critic(†)/ Bootstraped Return($^+$)	Critic Trained from scratch	Search space
LoLA	✓	✓($^+$)		z
LoLA (no-I)		✓($^+$)		z
LoLA (no-R)	✓		✓	z
LoLA (no-I, no-R)			✓	z
ReLA	✓	✓(†)		z
ReLA (no-I)		✓(†)		z
ReLA (no-R)	✓		✓	z
ReLA (no-I, no-R)			✓	z
TD3-z			✓	z
TD3 (I)	✓		✓	a
TD3-warm-start(I)	✓		✓	a
TD3-warm-start(I, R)	✓	✓(†)		a

Table 1: Summary of algorithm variations. Here, search space z indicates latent policy adaptation via the policy space $\{\pi_z\}$ constructed by the BFM, while a denotes finetuning in action space.

When looking at the baselines, we can notice that all action-space adaptation algorithms suffer a much more significant drop compared to latent policy adaptation. The performance gap between action-based and latent policy adaptation becomes even larger when looking at FB-CPR. In this case, all action-based algorithms completely unlearn in a few steps and are not able to rapidly recover. We think this is due to the large dimensionality of observation space, action space and policy model that lead to a much more complicated optimization problem.[5] On the other hand, in contrast to other BFMs, FB-CPR does not suffer any initial performance drop when using ReLA. Indeed, all latent policy adaptation algorithms (see Appendix 10 for additional experiments) achieve monotonic performance improvement, stressing even more that structured search in the latent policy space may be simpler than finetuning the whole policy in high-dimensional problems. This may be due to the fact that FB-CPR is the only BFM that is pre-trained with online environmental interactions, a setting that may reduce the distribution shift between pre-training and adaptation. Finally, the performance improvement due to the latent policy adaption is much more significant in this domain. The reason may reside in the critic training objective of FB-CPR; indeed FB-CPR uses a discriminator-based loss to regularize the policy space towards expert demonstrations. This may prevent the zero-shot inference to correctly identify the best policy for the task, while online adaptation seems to better search the policy space.

[5] A way to address this problem may be through policy regularization but this is outside the scope of this paper.

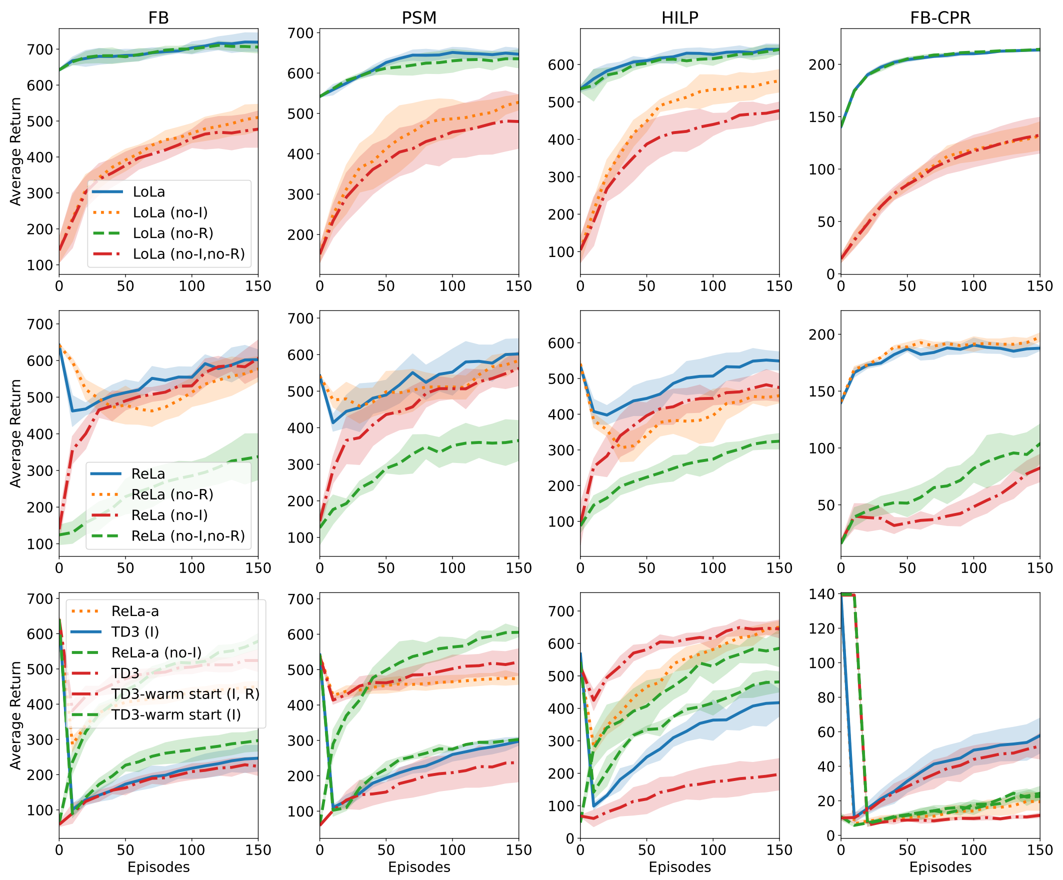

Figure 5: Average returns for several variations of `LoLA`, `ReLA`, and action-based TD3 with warm start. We use `no-R` to denote that we do not use the BFM's estimated value function (i.e., for `LoLA` we do not bootstrap the terminal state and for `ReLA` we learn a critic from scratch) and `no-I` to denote that we do not use zero-shot policy initialization. Finally, for TD3 we use `R` to denote that we use residual critic since the standard implementation learns a critic from scratch.

Finally, we would like to report an observation about the computational efficiency. On our hardware, LoLA runs at $\approx$ 157x the FPS of ReLA and other adaptation approaches. Specifically, ReLA runs at $\approx$ 14 FPS, and LoLA runs at $\approx 2,200$. This gaps presumably comes from the fact that `ReLA` needs to backpropagate gradient through the BFM estimated value function and policy both in the critic and actor updates, while `LoLA` has just a single actor update. The computational efficiency of LoLA along with its observed near-monotonic improvement for adaptation makes it appealing in practice.

4.2 What components are critical for fast adaptation?

In this section we assess the importance of leveraging BFM properties for fast online adaptation. We focus on ablating the need of *i)* zero-shot initialization and *ii)* BFM value function estimate, i.e., using a residual critic for `ReLA` and the bootstrapped Q-function for `LoLA`. We focus on the very early steps of the training to better inspect the results. Ablation variants are concisely shown in Table 1 for reference, and results are reported in Figure 5.

When *zero-shot initialization* is disabled not only the performance starts lower but also take significantly longer to match the baseline's returns (if they match at all). Unsurprisingly, zero-shot initialization helps in the search process. Leveraging the BFM's value function estimate does not hurt and often helps in reducing the initial performance drop. Looking at `LoLA`, BFM bootstrapping

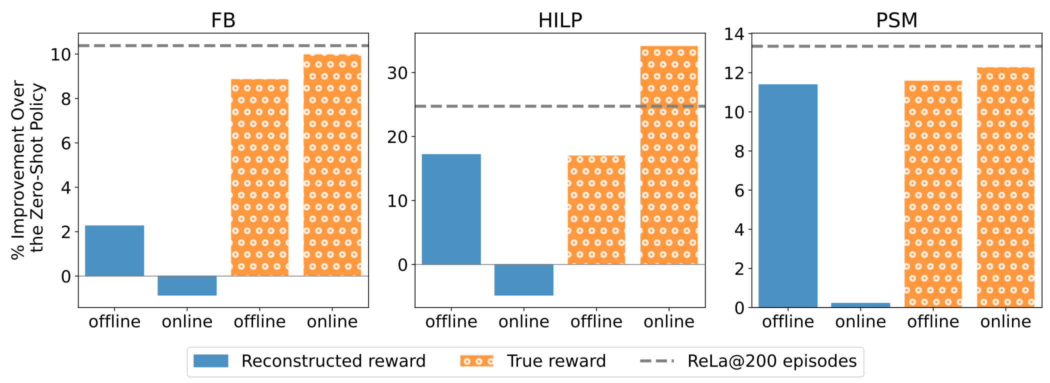

Figure 6: Performance improvement w.r.t. the zero-shot policy for a TD3-based method trained from scratch for 3M steps to perform search in the latent policy space (i.e., TD3-z). We report the results for both online and offline training using the ExoRL (Yarats et al., 2022) dataset. We also ablate learning with the true task reward and the reward reconstructed by the BFM methods. We average the results over all the task of the `Walker`, `Quadruped` and `Cheetah` domains. We report the average performance over 5 seeds. We additionally show the performance of `ReLA` after training for 200 episodes.

helps only marginally in all the domains. We believe that this is due to the small discount factor and large lookahead (we use 0.98 and a lookahead of 100 or 250); this combination significantly reduces the role of the bootstrapped Q-function (discounted by 0.13 or 0.006). When looking at `ReLA`, residual critic helps in DMC domains but not in the HumEnv, where zero-shot initialization is the most important dimension. On the other hand, when zero-shot initialization is disabled in the DMC domains, the importance of residual critic is particularly evident and leads to almost match the performance of the best algorithm. Finally, the residual critic is also very important when performing direct adaptation in the action-space and helps in a faster recovery from the initial drop.

4.3 Dissecting the Suboptimality of Zero-Shot RL Policies

The previous results show that BFMs are indeed learning skills that contain good policies for all the downstream tasks we study. This then raises the question on what causes the suboptimality of the zero-shot policy and the need of performing online adaptation to actually recover a better policy. We run a series of ablations with FB, HILP and PSM. We do not consider FB-CPR because our ablation involves offline training and we do not have access to an offline dataset for this model since it was trained online. We consider TD3 as learning algorithm since it is the building block of all the three BFMs and focus on latent policy adaptation (we call this approach TD3-z to avoid confusion with TD3 used in the previous sections to optimize the full policy network). For all experiments in this section, we consider the standard scenario of **training from scratch**, no zero-shot initialization and no-residual critic. Specifically when searching in z space, we use a pretrained BFM actor and initialize z along with the critic randomly and when learning in action space we initialize both the actor and critic randomly. We report the performance of TD3-z after 3M training steps when using the true reward function and the reward function reconstructed by the BFMs[6], both offline and online. We do not consider `pointmass` in this test since TD3 does not work well when trained online due to the challenging exploration in the long-horizon tasks considered in this domain.

Overall, these experiments confirm that BFMs can express much better policies than the zero-shot policy and that optimizing for true rewards is crucial to unlock their full performance. When optimizing for the latent reward, offline TD3-z can already improve the zero-shot performance revealing

[6]Latent or reconstructed reward is given by $\tilde{r}_z(s) = \varphi(s).z_r$

the difficulty of optimizing all polices $\{\pi_z\}$ simultaneously during the pre-training process.[7] Interestingly, when moving to online training on the latent reward performance can even drop. We conjecture the cause is the distribution shift between online and offline samples. Given that the models were trained offline, their reward prediction degrades on out-of-distribution samples encountered during online adaptation, further skewing towards learning policies that are even less correlated to the true reward. This is confirmed when looking at the performance when optimizing for the true reward, which consistently lead to better results across offline and online tests, with online methods being overall better. This ablation confirms that focusing on searching in the z-space, while correcting the embedding errors is the right strategy to achieve fast adaptation online. Indeed, we see that `ReLA` recovers better policies than the one obtained by training from scratch TD3-z online for 3M episodes in only 200 episodes. Even faster if we use `LoLA`. This shows that leveraging information from the BFM is useful in many cases.

5 Related Work

Unsupervised RL pre-training: For language and vision, unsupervised pretraining has paved the way to extracting meaningful structure from data, scaling up, and obtaining impressive results for transfer and zero-shot generalization to different downstream tasks. In recent years, approaches have been proposed for unsupervised reinforcement learning: a training paradigm where a learning agent attempts to extract world structure and representations that will later allow it to solve diverse multi-step decision-making problems in the environment. Various objectives have been proposed: world modeling (Bruce et al., 2024; Hansen et al., 2023), intrinsic rewards (Schmidhuber, 2019; Stadie et al., 2015; Sekar et al., 2020; Pathak et al., 2017), empowerment and mutual information skill learning (Klyubin et al., 2005; Eysenbach et al., 2018; Rajeswar et al., 2023; Gregor et al., 2016), goal-reaching (Ma et al., 2022; Park et al., 2023; 2024), successor measures (Touati & Ollivier, 2021; Agarwal et al., 2024; Sikchi et al., 2024) among others. In this work, we restrict our focus to the class of unsupervised RL objectives that learn a family of policies and allow us to query for a near-optimal policy given any test-time reward function without further learning or planning in the environment. Approaches belonging to this class often learn a state representation and use that to define the class of reward functions for which they learn the set of optimal policies. At inference time, they output the policy that is optimal for the projection of the reward function to this class of reward functions.

Fine-tuning and adaptation with unsupervised RL models: Similar to supervised pre-training approaches, unsupervised zero-shot RL models are not expected to output optimal policies for the given task. Rather they are expected to output a reasonable policy initialization that can be later fine-tuned or adapted. Prior approaches for policy adaptation with pre-trained RL models have mostly studied the offline-to-online setting. In this setting, a policy is trained with reward-labeled transitions first using offline data with specialized offline RL algorithms and then allowed to fine-tune by interacting with the environment and *retaining access to the offline data*. Offline RL algorithms (Levine et al., 2020; Sikchi et al., 2023) incorporate pessimism to avoid overestimation by restricting the policy to visit states closer to the dataset. Naively using the same algorithm to finetune online has been observed to lead to slow performance improvements and using an online RL algorithm leads to performance collapse at the beginning of fine-tuning (Luo et al., 2023). This behavior has been attributed to a distribution shift for critic-learning (Yu & Zhang, 2023), and prior works have investigated various techniques, such as calibration of Q-functions to mitigate this problem (Nakamoto et al., 2023). Our work, considers a different but practical paradigm for adaptation where a) no offline data is retained during finetuning and b) we learn from reward-free transitions. First, by only retaining pre-trained models we reduce compute requirements of learning from large pre-training offline datasets (Zhou et al., 2024), and by considering reward-free transitions we have a single model that can adapt to any downstream task (Kim et al., 2024). Learning online without retaining

[7]The fact that performance of HILP and PSM improve by about than 10-15% by offline training on the reconstructed reward might may be due to a non-perfect pre-train. Indeed, the pre-training condition should ensure that the actor is already optimal on any reconstructed reward on the training data distribution.

offline data suffers a significant initial drop of performance with respect to the pre-trained policy as recently investigated by Zhou et al. (2024).

6 Conclusion

Unsupervised zero-shot RL pre-training can result in an agent (a type of Behavioral Foundation Model, BFM) capable of accomplishing a wide variety of tasks having a noticeable but expected degree of suboptimality. This paper investigates and addresses the question of how to adapt these agents to be better at a task specified during test-time with limited environment interactions. We propose two fast adaptation strategies (`LoLA` and `ReLA`); The key insight behind our methods is to reuse pre-trained knowledge from BFM strategically and search over the learned latent policy space that provides a low-dimensional landscape favorable for gradient-based optimization. We have demonstrated the effectiveness of these strategies across various zero-shot BFMs. Notably, `LoLA`, an actor-only adaptation algorithm, demonstrates monotonic performance improvement on all domains and BFMs, making it a reliable choice when privileged resets are permitted. However, our findings also reveal an initial performance drop when employing any actor-critic method, including our proposed `ReLA` algorithm. This highlights the need for further investigation into mitigating forgetting in the actor-critic class of approaches. Future research directions include exploring meta-learning adaptation techniques, including in-context adaptation by learning to adapt in multi-task settings to optimize learning costs and improve overall performance.

7 Acknowledgments

This work was primarily done at FAIR at Meta. This work has also partly taken place in part in the Safe, Correct, and Aligned Learning and Robotics Lab (SCALAR) at The University of Massachusetts Amherst. HS and SN as part of the SCALAR research are supported in part by the NSF (IIS-2323384), the Center for AI Safety (CAIS), and the Long-Term Future Fund. HS and AZ are funded by NSF 2340651, NSF 2402650, DARPA HR00112490431, and ARO W911NF-24-1-0193.

Broader Impact Statement

Our work seeks to advance the adaptability of learning agents that interact with the environment. Our work pushes the frontier on scalable and adaptable agents by building upon unsupervised learning objectives that allow us to reuse a single trained model capable of a variety of tasks and proposing approaches that make these models more proficient at a task specified during test time rapidly without retaining any data used for pretraining. Prior works have studied a variety of objectives for unsupervised RL but fall short in their investigation of adaptation or propose a strategy that unlearns the policy before starting to improve performance. Our proposed approach is the first to our knowledge, to demonstrate monotonic performance improvement for pretrained RL agents without access to training data. There are potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Siddhant Agarwal, Harshit Sikchi, Peter Stone, and Amy Zhang. Proto successor measure: Representing the space of all possible solutions of reinforcement learning. *arXiv preprint arXiv:2411.19418*, 2024.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *arXiv preprint arXiv:2101.07123*, 2021.

Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*, 2018.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=H1lJJnR5Ym.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.

Edoardo Cetin, Ahmed Touati, and Yann Ollivier. Finer behavioral foundation models via autoregressive features and advantage weighting. *arXiv preprint arXiv:2412.04368*, 2024.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. 2023.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

Steven Hansen, Will Dabney, Andre Barreto, Tom Van de Wiele, David Warde-Farley, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. *arXiv preprint arXiv:1906.05030*, 2019.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pp. 6023–6029. IEEE, 2019.

Junsu Kim, Seohong Park, and Sergey Levine. Unsupervised-to-online reinforcement learning. *arXiv preprint arXiv:2408.14785*, 2024.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. Empowerment: A universal agent-centric measure of control. In *2005 ieee congress on evolutionary computation*, volume 1, pp. 128–135. IEEE, 2005.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! 2019.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pp. 6736–6747. PMLR, 2021.

Yicheng Luo, Jackie Kay, Edward Grefenstette, and Marc Peter Deisenroth. Finetuning from offline reinforcement learning: Challenges, trade-offs and practical solutions. *arXiv preprint arXiv:2303.17396*, 2023.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

Zakaria Mhammedi, Dylan J. Foster, and Alexander Rakhlin. The power of resets in online reinforcement learning. In *NeurIPS*, 2024.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *arXiv preprint arXiv:2303.05479*, 2023.

Seohong Park, Oleh Rybkin, and Sergey Levine. Metra: Scalable unsupervised rl with metric-aware abstraction. *arXiv preprint arXiv:2310.08887*, 2023.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. *arXiv preprint arXiv:2402.15567*, 2024.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Matteo Pirotta, Andrea Tirinzoni, Ahmed Touati, Alessandro Lazaric, and Yann Ollivier. Fast imitation via behavior foundation models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=qnWtw3l0jb.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Sai Rajeswar, Pietro Mazzaglia, Tim Verbelen, Alexandre Piché, Bart Dhoedt, Aaron Courville, and Alexandre Lacoste. Mastering the unsupervised reinforcement learning benchmark from pixels. In *International Conference on Machine Learning*, pp. 28598–28617. PMLR, 2023.

Krishan Rana, Ming Xu, Brendan Tidd, Michael Milford, and Niko Sünderhauf. Residual skill policies: Learning an adaptable skill-based action space for reinforcement learning for robotics. In *Conference on Robot Learning*, pp. 2095–2104. PMLR, 2023.

Juergen Schmidhuber. Reinforcement learning upside down: Don't predict rewards–just map them to actions. *arXiv preprint arXiv:1912.02875*, 2019.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International conference on machine learning*, pp. 8583–8592. PMLR, 2020.

Harshit Sikchi, Wenxuan Zhou, and David Held. Learning off-policy with online planning. In *Conference on Robot Learning*, pp. 1622–1633. PMLR, 2022.

Harshit Sikchi, Qinqing Zheng, Amy Zhang, and Scott Niekum. Dual rl: Unification and new methods for reinforcement and imitation learning, 2023.

Harshit Sikchi, Siddhant Agarwal, Pranaya Jajoo, Samyak Parajuli, Caleb Chuck, Max Rudolph, Peter Stone, Amy Zhang, and Scott Niekum. Rl zero: Zero-shot language to behaviors without any supervision. *arXiv preprint arXiv:2412.05718*, 2024.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. Deepmind control suite. *CoRR*, abs/1801.00690, 2018.

Andrea Tirinzoni, Ahmed Touati, Jesse Farebrother, Mateusz Guzek, Anssi Kanervisto, Yingchen Xu, Alessandro Lazaric, and Matteo Pirotta. Zero-shot whole-body humanoid control via behavioral foundation models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9sOR0nYLtz.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. In *NeurIPS*, pp. 13–23, 2021.

Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In *ICLR*. OpenReview.net, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in rl: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*, 2018.

Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.

Zishun Yu and Xinhua Zhang. Actor-critic alignment for offline-to-online reinforcement learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 40452–40474. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/yu23k.html.

Tom Zahavy, Yannick Schroecker, Feryal Behbahani, Kate Baumli, Sebastian Flennerhag, Shaobo Hou, and Satinder Singh. Discovering policies with domino: Diversity optimization maintaining near optimality. *arXiv preprint arXiv:2205.13521*, 2022.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 11975–11986, 2023.

Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data. *arXiv preprint arXiv:2412.07762*, 2024.

Supplementary Materials

The following content was not necessarily subject to peer review.

7.1 Motivation for residual learning

Notation: We use matrix form and we identify for any $z \in \mathcal{Z}$, $\psi_z : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ and $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ by $\psi_z \in \mathbb{R}^{d \times |\mathcal{S} \times \mathcal{A}|}$ and $\phi \in \mathbb{R}^{d \times |\mathcal{S}|}$ respectively and $D_\rho = \text{diag}((\rho(s))_{s \in \mathcal{S}})$. Similarly, we identify for any $z \in \mathcal{Z}$, $F_z : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ and $B : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ by $F_z \in \mathbb{R}^{d \times |\mathcal{S} \times \mathcal{A}|}$ and $B \in \mathbb{R}^{d \times |\mathcal{S}|}$ respectively. We denote by $\Pi_\phi = \phi^\top \left(\phi D_\rho \phi^\top\right)^{-1} \phi D_\rho$ the $L^2(\rho)$ orthogonal projection onto the linear span of ϕ.

Proposition 1. *Let $\phi : S \to \mathbb{R}^d$ a state feature map and $\{\psi_z\}_{z \in Z}$ the corresponding universal successor features for the policy family $\{\pi_z\}_{z \in Z}$, i.e $\psi_z(s,a) = \mathbb{E}[\sum_{t \geq 0} \gamma^t \phi(s_{t+1}) \mid (s,a), \pi_z]$ Then, for any reward function $r : S \to R$, we have: $Q_r^{\pi_z} = \psi_z^\top z_r + Q_{r-\phi^\top z_r}^{\pi_z}$ where $z_r = \mathbb{E}_{s \sim \rho}[\phi(s)\phi(s)^\top]^{-1} \mathbb{E}_{s \sim \rho}[\phi(s) r(s)]$, and $Q_{r-\phi^\top z_r}^{\pi_z}$ is the Q-function of the residual reward $r - \phi^\top z_r = (I - \Pi_\phi) r$.*

Proof. for any reward function $r \in \mathbb{R}^S$, we have $r = (\Pi_B + I - \Pi_B) r$, then

$$\begin{aligned}
Q_r^{\pi_z} &= M^{\pi_z} r \\
&= M^{\pi_z} (\Pi_B + I - \Pi_B) r \\
&= M^{\pi_z} \Pi_B r + M^{\pi_z} (I - \Pi_B) r \\
&= M^{\pi_z} \phi^\top \left(\phi D_\rho \phi^\top\right)^{-1} \phi D_\rho r + M^{\pi_z} (r - \phi^\top \left(\phi D_\rho \phi^\top\right)^{-1} \phi D_\rho r) \\
&= \psi_z^\top z_r + Q_{r-\phi^\top z_r}^{\pi_z}
\end{aligned}$$

where the last equation follows from the fact that $\psi_z^\top = M^{\pi_z} \phi^\top$ and $z_r = \mathbb{E}_{s \sim \rho}[\phi(s)\phi(s)^\top]^{-1} \mathbb{E}_{s \sim \rho}[\phi(s) r(s)] = \left(\phi D_\rho \phi^\top\right)^{-1} \phi D_\rho r$ ◻

Proposition 2. *Let assume that for any $z \in Z$, F_z is a stationary point of the FB training loss $\ell(F, B)$, namely, the functional derivative $\frac{\partial l}{\partial F_z}$ of the loss with respect of F_z is 0. Then,*

$$\begin{aligned}
Q_r^{\pi_z} &= F_z^\top z_r + Q_{(I - \Pi_B) r}^{\pi_z} \\
&= F_z^\top z_r + \left(M^{\pi_z} - F_z^\top B D_\rho\right) (r - \Pi_B r)
\end{aligned}$$

where $z_r = \mathbf{E}_{s \sim \rho}[B(s) r(s)]$.

Proof. Let's remind the FB training loss:

$$\ell(F, B) = \mathbb{E}_{\substack{z, (s,a) \sim \rho \\ s^+ \sim \rho}} \left[\left(F(s,a,z)^\top B(s^+) - P(\mathrm{d}s^+ \mid s,a)/\rho(\mathrm{d}s^+) - (P^{\pi_z} \bar{F})(s,a,z)^\top \bar{B}(s^+) \right)^2 \right]$$

In matrix form, we obtain:

$$\ell(F, B) = \mathbb{E}_z \left[\text{Trace} \left(\left(F_z^\top B - P D_\rho^{-1} - \gamma P^{\pi_z} \bar{F}_z^\top \bar{B} \right)^\top D_\rho \left(F_z^\top B - P D_\rho^{-1} - \gamma P^{\pi_z} \bar{F}_z^\top \bar{B} \right) D_\rho \right) \right]$$

if F_z satisfies the stationarity conditions, *i.e*, $\frac{\partial \ell}{\partial F_z} = 0$, then, we have

$$\begin{aligned}\frac{\partial \ell}{\partial F_z} = 0 &\Rightarrow 2BD_\rho \left(F_z^\top B - PD_\rho^{-1} - \gamma P^{\pi_z} F_z^\top B\right)^\top D_\rho = 0\\ &\Rightarrow 2D_\rho \left(F_z^\top B - PD_\rho^{-1} - \gamma P^{\pi_z} F_z^\top B\right) D_\rho B^\top = 0\\ &\Rightarrow F_z^\top BD_\rho B^\top = PB^\top + \gamma P^{\pi_z} F_z^\top BD_\rho B^\top\\ &\Rightarrow F_z^\top = M^{\pi_z} B^\top \left(BD_\rho B^\top\right)^{-1}\\ &\Rightarrow F_z^\top BD_\rho = M^{\pi_z} B^\top \left(BD_\rho B^\top\right)^{-1}\\ &\Rightarrow F_z^\top BD_\rho = M^{\pi_z} B^\top \left(BD_\rho B^\top\right)^{-1} BD_\rho\end{aligned}$$

Therefore $F_z^\top BD_\rho = M^{\pi_z}\Pi_B$ where $\Pi_B = B^\top \left(BD_\rho B^\top\right)^{-1} BD_\rho$ is the $L^2(\rho)$ orthogonal projection onto the linear span of B.

Let $\Pi_{B^\perp}$ the orthogonal projection onto the orthogonal of B. By definition, we have $\Pi_{B^\perp} = I - \Pi_B$

We have:

$$\begin{aligned}Q_r^{\pi_z} &= M^{\pi_z} r\\ &= M^{\pi_z}(\Pi_B + \Pi_{B^\perp})r\\ &= M^{\pi_z}\Pi_B r + M^{\pi_z}\Pi_{B^\perp} r\\ &= F_z^\top BD_\rho r + M^{\pi_z}\Pi_{B^\perp} r\\ &= F_z^\top z_r + M^{\pi_z}\Pi_{B^\perp} r\end{aligned}$$

where $z_r = BD_\rho r = \mathbb{E}_{s\sim\rho}[B(s)r(s)]$

Therefore, we have:

$$Q_r^{\pi_z} = F_z^\top z_r + Q^{\pi_z}_{\Pi_{B^\perp} r}$$

where the second term is the the Q-function of the residual reward $\Pi_{B^\perp} r$

Moreover, since $\Pi_{B^\perp}^2 = \Pi_{B^\perp}$, we can write:

$$\begin{aligned}Q_r^{\pi_z} &= F_z^\top z_r + (M^{\pi_z}\Pi_{B^\perp})(\Pi_{B^\perp} r)\\ &= F_z^\top z_r + \left(M^{\pi_z} - F_z^\top BD_\rho\right)(r - \Pi_B r)\end{aligned}$$

Which means that the residual term can be expressed as the successor measure approximation error (due to the low-rank decomposition of FB model) multiplied by the reward error (due to the reward embedding in the span of B). □

8 Psuedocode

Algorithm 1 and 2 outline pseudocode for ReLA and LoLA respectively. The use of an terminal off-policy critic LoLA has also been previously motivated in prior works to contribute to error reduction in value function (Sikchi et al., 2022; Hansen et al., 2022).

9 Experimental Setup

9.1 Environments

We list the continuous control environments from the DeepMind Control Suite (Tassa et al., 2018) and Humenv (Tirinzoni et al., 2025) used in this work in Table 2.

```
Algorithm 1: ReLA
Load Frozen BFM's successor features $\psi(s, a, z)$ and
  policy $\pi_z(s)$ networks.
Initialize residual critic networks $Q^{\text{residual}}_{\theta_1}$, $Q^{\text{residual}}_{\theta_2}$, replay
  buffer $\mathcal{D}_{\text{online}}$, exploration std $\sigma$, Update to Data ratio
  (UTD) $M$, Initialize target networks: $Q^{\text{residual}}_{\theta'_1} \leftarrow Q^{\text{residual}}_{\theta_1}$,
  $Q^{\text{residual}}_{\theta'_2} \leftarrow Q^{\text{residual}}_{\theta_2}$.
Compute zero-shot latent $z_r$ using inference samples for
  the BFM agent with test-time reward function.
for each environment step $t$ do
    Select $a_t = \pi_z(s_t) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$
    Execute $a_t$; observe $r_t, s_{t+1}$
    Store $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}_{\text{online}}$
    Sample M mini-batches
      $\text{Batch}_i = \{(s_i, a_i, r_i, s'_i)\} \sim \mathcal{D}_{online}$
    Compute target Q-value:
      $y_i = r_i + \gamma(\psi(s'_i, \pi_z(s'_i), z_r) \cdot z_r +$
      $\min\{Q_{\theta'_1}(s'_i, \pi_z(s'_i)),\ Q_{\theta'_2}(s'_i, \pi_z(s'_i))\})$
    TemporalDifferenceUpdate$(\psi(s_i, a_i, z_r) \cdot z_r +$
      $Q^{\text{residual}}_{\theta_{k \in [1,2]}}, y_i)$ for $i \in [M]$ using critic
      parameterization from Eq 5
    Latent Policy Update Update $z$ taking gradient step as
    in Eq 6 on $\cup_{i \in [m]} \text{Batch}_i$ .
    Update target networks by polyak averaging;
end
```

```
Algorithm 2: LoLA
Load Frozen BFM's successor features $\psi(s, a, z)$ and
  policy $\pi_z(s)$ networks
Initialize latent policy $\pi_{\mu,\sigma} = \mathcal{N}(\mu = z_r, \sigma)$, replay
  buffer $\mathcal{D}_{\text{online}}$, sampling state distribution $\nu(\mathcal{D}_{\text{online}}, d_0)$, z
  budget $k$, intial state budget $m$, horizon $n$
Compute zero-shot latent $z_r$ using inference samples for
  the BFM agent with test-time reward function.
for each gradient step do
    for b=1..m do
        $s_0 \sim \mu(\mathcal{D}_{\text{online}}, d_0)$
        for i=1..k do
            $z^i_b \sim \pi_{\mu,\sigma}$, Reset to $s_0$
            Rollout trajectory $\tau^i_b$ by taking actions given
              by $a_t = \pi_{z^i_b}(s_t)$
            Compute $R(s_0, z^i_b)$
            Collect states from $\tau^i_b$ in $\mathcal{D}_{\text{online}}$
        end
    end
    Update $\pi_{\mu,\sigma}$ by taking gradient step in Eq 7.
end
```

Figure 7: Pseudocode of our proposed adaptation methods: Residual Latent Adaptation (ReLA) and Lookahead Latent Adaptation (LoLA).

Domain	Observation dimension	Action dimension	Episode length
Pointmass	4	2	1000
Walker	24	6	1000
Cheetah	17	6	1000
Quadruped	78	12	1000
HumEnv	358	69	300

Table 2: Overview of observation spaces, action spaces and episode length of environments used in this work.

9.2 Behavioral Foundation Models

We trained all the BFMs except for the FB-CPR model that is publicly available (code link).

Offline BFMs. We train the BFMs using the publicly available dataset from ExoRL (Yarats et al., 2022) collected using the RND algorithm (Burda et al., 2019). We used the authors implementation for FB and FB-CPR (code link) and reimplemented PSM and HILP. We report in Table 3 the set of hyperparameter used for the algorithms.

FB architecture. The backward representation network $B(s)$ is represented by a feedforward neural network with two hidden layers, each with 256 units, that takes as input a state and outputs a d-dimensional embedding. For the forward network $F(s, a, z)$, we first preprocess separately (s, a) and (s, z) by two feedforward networks with one single hidden layer (with 1024 units) to 512-dimentional space. Then we concatenate their two outputs and pass it into two heads of feedforward networks (each with one hidden layer of 1024 units) to output a d-dimensional vector. For the policy network $\pi(s, z)$, we first preprocess separately s and (s, z) by two feedforward networks with one single hidden layer (with 1024 units) to 512-dimentional space. Then we concatenate their two outputs and pass it into another one single hidden layer feedforward network (with 1024 units) to output to output a d_A-dimensional vector, then we apply a `Tanh` activation as the action space is $[-1, 1]^{d_A}$.

Table 3: BFM hyperparameters. We largely reuse the hyperparameters from Pirotta et al. (2024) for FB, from (Park et al., 2024) for HILP.

		Hyperparameter	**Walker**	**Cheetah**	**Quadruped**	**Pointmass**
FB	Forward Backward (Touati & Ollivier, 2021)	Embedding Dimension d	100	50	50	100
		Embedding Prior	S^d	S^d	S^d	S^d
		Embedding Prior Goal Prob.	0.5	0.5	0.5	0.5
		B Normalization	ℓ_2	ℓ_2	ℓ_2	ℓ_2
		Orthonormal Loss Coeff.	1	1	1	1
	Optimizer (Adam) (Kingma & Ba, 2015)	Learning Rate (F, B)	$(10^{-4}, 10^{-4})$	$(10^{-4}, 10^{-4})$	$(10^{-4}, 10^{-4})$	$(10^{-4}, 10^{-6})$
		Learning Rate (π)	10^{-4}	10^{-4}	10^{-4}	10^{-6}
		Target Network EMA	0.99	0.99	0.99	0.99
HILP	Hilbert Representations (Park et al., 2024)	Embedding Dimension d	50	50	50	100
		Feature Learning Expectile	0.5	0.5	0.5	0.5
		Feature Learning Discount Factor	0.98	0.98	0.98	0.98
		Successor feature loss	Q-loss	Q-loss	Q-loss	Q-loss
	Optimizer (Adam)	Learning Rate (SF, F)	$(10^{-4}, 10^{-5})$	$(10^{-4}, 10^{-4})$	$(10^{-4}, 10^{-4})$	$(10^{-4}, 10^{-4})$
		Learning Rate (π)	10^{-4}	10^{-4}	10^{-4}	10^{-4}
		Target Network EMA features	0.995	0.995	0.995	0.995
		Target Network EMA SF	0.99	0.99	0.99	0.99
PSM	Proto Successor Measures (Agarwal et al., 2024)	Embedding Dimension d	100	50	50	100
		Policy Codebook Size	2^{16}	2^{16}	2^{16}	2^{16}
		Feature Learning Timesteps	$400k$	$400k$	$400k$	$400k$
		Embedding Prior Goal Prob.	0.5	0.5	0.5	0.5
		B Normalization	ℓ_2	ℓ_2	ℓ_2	ℓ_2
		Orthonormal Loss Coeff.	1	1	1	1
	Optimizer (Adam) (Kingma & Ba, 2015)	Learning Rate (F, B)	$(10^{-4}, 10^{-4})$	$(10^{-4}, 10^{-4})$	$(10^{-4}, 10^{-4})$	$(10^{-4}, 10^{-6})$
		Learning Rate (π)	10^{-4}	10^{-4}	10^{-4}	10^{-6}
		Target Network EMA	0.99	0.99	0.99	0.99
	Policy (TD3) (Fujimoto et al., 2018)	Target Policy Noise	$\mathcal{N}(0, 0.2)$	$\mathcal{N}(0, 0.2)$	$\mathcal{N}(0, 0.2)$	$\mathcal{N}(0, 0.2)$
		Target Policy Clipping	0.3	0.3	0.3	0.3
		Policy Update Frequency	1	1	1	1
	Common	Batch Size	1024	1024	1024	1024
		Gradient Steps	3M	3M	3M	3M
		Discount Factor γ	0.98	0.98	0.98	0.99
		Reward Inference Samples	$250,000$	$250,000$	$250,000$	$250,000$
		ExoRL number of trajectories	$5,000$	$5,000$	$5,000$	$5,000$

For all the architectures, we apply a layer normalization and `Tanh` activation in the first layer in order to standardize the states and actions. We use `Relu` for the rest of layers. We also pre-normalize $z : z \leftarrow \sqrt{d}\frac{z}{\|z\|_2}$ in the input of F, and π.

HILP architecture. We use the same policy architecture as FB as well F-architecture for the successor features. We learn the HILP features using a 2 layers MLP with hidden dimension 1024. Even in this case, z is normalized.

PSM architecture. We use the same policy architecture as FB as well F-architecture for the successor features. We learn the PSM features using a 2 layers MLP with hidden dimension 256. Even in this case, z is normalized.

9.3 Algorithm Implementation

All the actor-critic algorithms are implemented using TD3 (Fujimoto et al., 2018) as the base off-policy algorithm. When learning from scratch we use a 2 two layers MLP with hidden dimension 1024 and `Relu` activation. We use the same configuration also for the critic.

For `ReLA`, we use a small 2 layer MLPs with 64 hidden dimensions and ReLU activation as residual network. In the ablation, when residual critic is deactivated, we use the same critic network as for standard TD3.

For `LoLA`, we use a Gaussian policy centered around the learned z and learn simultaneously mean and standard deviation.

9.4 Hyperparameters

For all baselines and our method, we run a hyperparameter sweep across domains and tasks and choose the configuration that performs the best across tasks for each BFM.

TD3-based algorithms. We run a hyperparameter sweep on Update to Data ratio (UTD) in $[1, 4, 8]$, actor update in frequency in $[1, 4]$. We use a small 2 layer MLP with 64 hidden nodes for the residual network which we found to work best for fast adaptation. When not using residual critic, we learn a critic from scratch using a 2 layer MLP with 1024 hidden nodes. We use 10^{-4} as learning rate for both critic and actor. We use either warm start of 0 steps or 5000 steps.

`LoLA`. We consider hyperparameter sweep between a lookahead horizons of $[50, 100, 250]$, the number of total trajectories per update to be 10, and number of trajectories for a sampled state to be 5 (for calculating baseline). We sweep between $[0, 0.2, 0.5]$ for the probability of resetting to initial state distribution and otherwise sampling from states encountered in replay buffer. We sweep also the learning rate in $[0.1, 0.05]$.

10 Additional Experiments

As mentioned in the main paper, `pointmass` is the domain where actor-critic algorithms incurs a significant initial drop. Figure 8 shows the average performance improvement without the `pointmass` domain. As we can see, `ReLA` has still a initial drop but it is much more reduced compared to what reported in the main. Previous papers (e.g. Pirotta et al., 2024), noticed that a smaller learning rate helped in `pointmass`. In our experiments we kept the learning rate fixed at 10^{-4} for all the domains, it would be interesting to test different values.

10.1 Per Algorithm Per Domain Ablation Studies

We conduct extensive ablation studies to understand the impact of key design choices in our methods, specifically: (1) zero-shot initialization in LoLA and ReLA variants, (2) value function bootstrapping in LoLA, (3) residual critics in ReLA variants and action-based TD3 with warm start. Table 4 provides a comprehensive list of the algorithm variants considered.

We evaluated these variants across four DMC domains (Quadruped, Pointmass, Cheetah, Walker) using FB, HILP and PSM, and on HumEnv using FB-CPR, each experiment conducted over five random seeds. The results are shown in Figure 9, 10, 11 and 12.

Zero-Shot Initialization (no-zs-init): Removing zero-shot initialization consistently degraded early-stage performance across all methods and domains, with the only exception being ReLA-a with FB and PSM on pointmass. The benefit of zero-shot initialization is especially significant on LoLA.

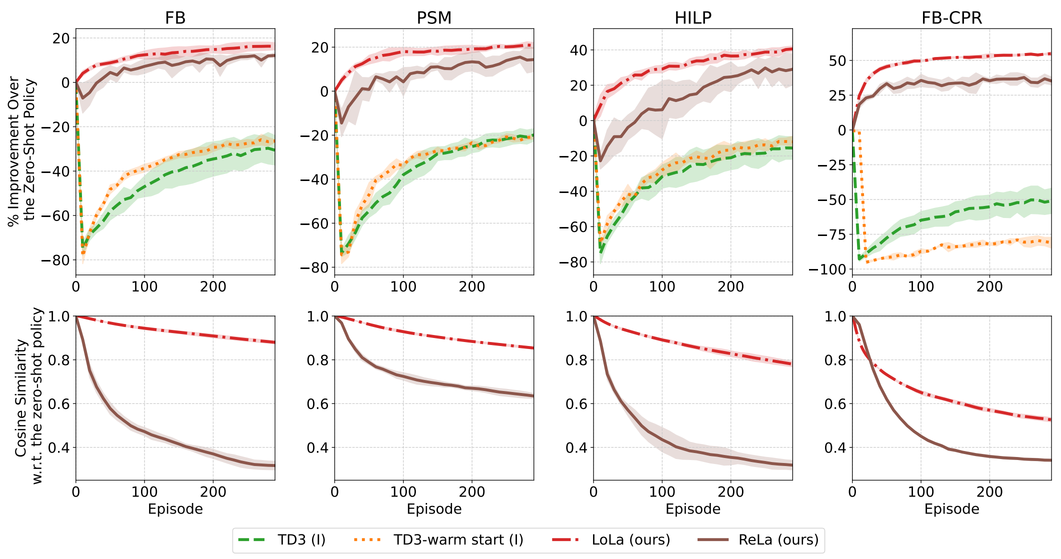

Figure 8: **Top**: Performance improvement w.r.t. the zero-shot policy for different online fast adaptation methods and BFMs without the `Pointmass` domain.

Algorithm	Zero-Shot Policy Initialization	Residual Critic (†) or Bootstrapped Return ($^{+}$)	Critic Trained from scratch	Search space	WSRL	
`LoLA`	✓	✓($^{+}$)		z		
`LoLA` (no-I)		✓($^{+}$)		z		actor-only
`LoLA` (no-R)	✓		✓	z		
`LoLA` (no-I, no-R)			✓	z		
`ReLA`	✓	✓(†)		z		
`ReLA`-warm-start	✓	✓(†)		z	✓	actor-critic
`ReLA` (no-I)		✓(†)		z		
`ReLA` (no-R)	✓		✓	z		
`ReLA`-warm-start (no-R)	✓		✓	z	✓	
`ReLA` (no-I, no-R)			✓	z		
`ReLA`-a	✓	✓(†)		a		
`ReLA`-a (no-I)		✓(†)		a		actor-critic
`ReLA`-a (no-R)	✓		✓	a		
`ReLA`-a (no-I, no-R)			✓	a		
TD3-z			✓	z		
TD3 (I)	✓		✓	a		actor-critic
TD3-warm-start (I) (i.e., using WSRL)	✓		✓	a	✓	
TD3-warm-start (I, R)	✓	✓(†)		a	✓	

Table 4: Summary of the algorithm variations considered in the main paper. Search space z means latent policy adaptation leveraging the policy space $\{\pi_z\}$ constructed by the BFM. Search space a denotes fine-tuning in action space (i.e., updating all policy network parameters).

Bootstrapping (no-bootstrap): We hypothesized that value functiom bootstrapping could help stabilizing LoLA. However, we did not notice such benefit from our ablation experiments.

Residual critics (no-residual): Removing residual critics in ReLA variants and TD3-based algorithm strongly impaired the effectiveness of the algorithm. This effect was especially pronounced for ReLA-a and TD3 on DMC domains.

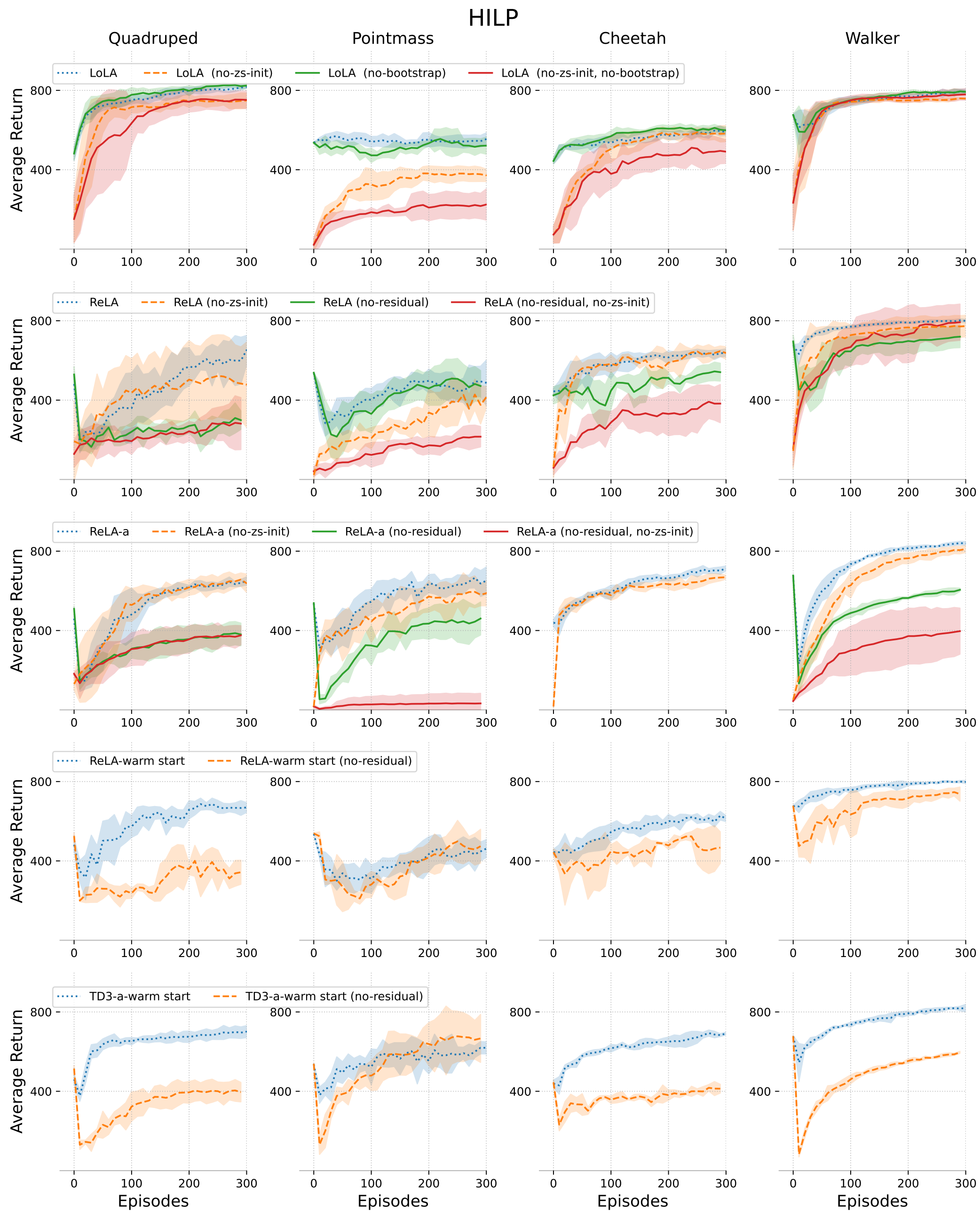

Figure 9: Ablation studies evaluating adaptations of HILP on four DMC tasks (Quadruped, Pointmass, Cheetah, Walker). Experiments include disabling zero-shot initialization ("no-I") and/or removing residual critics ("no-R") from LoLA, ReLA, and three additional variants: (1) ReLA-a: update a instead of z in ReLA, (2) ReLA with warm start (ReLA-warm start), and (3) action-based TD3 with warm start (TD3-warm start). Shaded areas represent standard errors across 5 seeds.

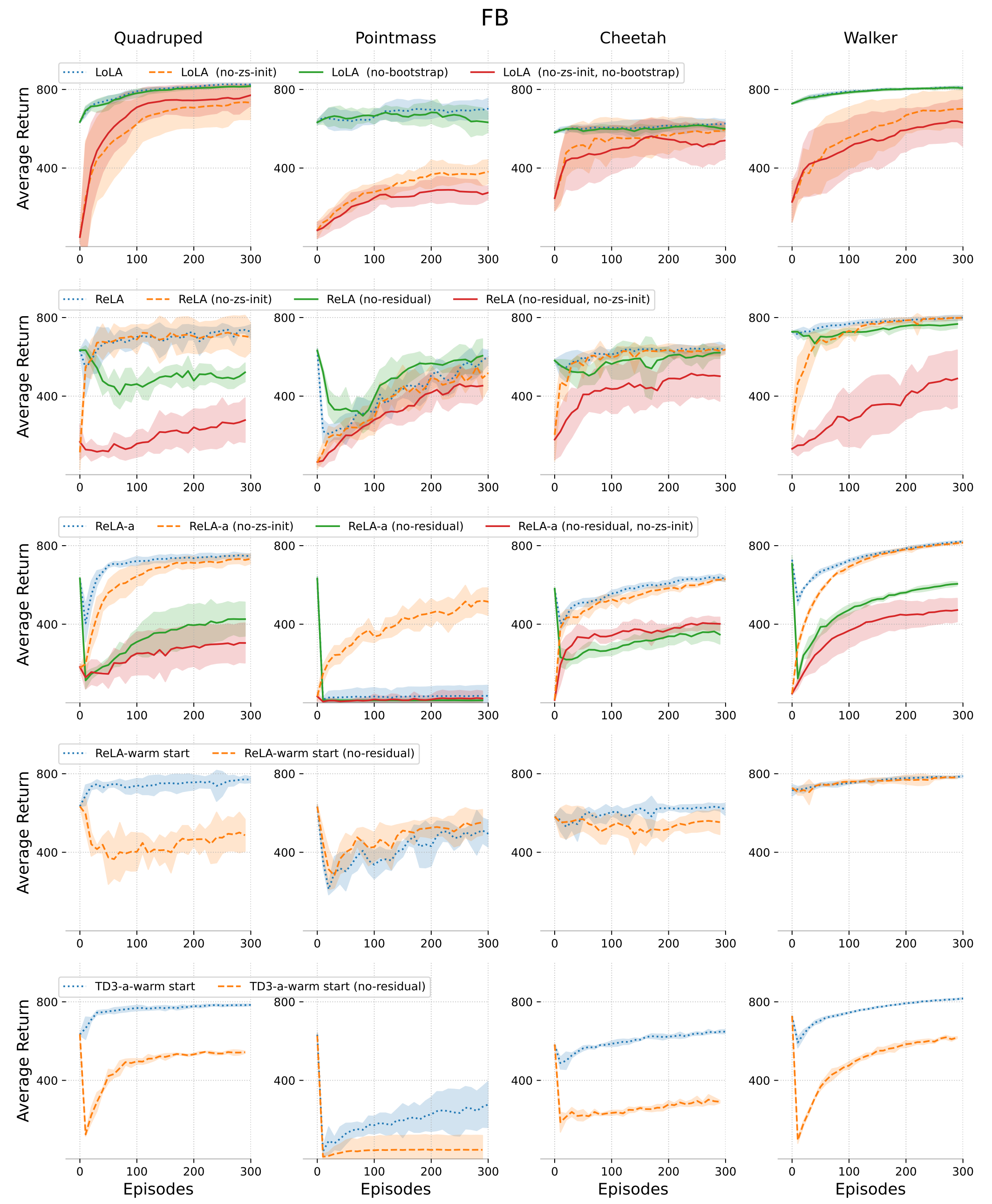

Figure 10: Ablation studies evaluating adaptations of FB on four DMC tasks (Quadruped, Pointmass, Cheetah, Walker). Experiments include disabling zero-shot initialization ("no-I") and/or removing residual critics ("no-R") from LoLA, ReLA, and three additional variants: (1) ReLA-a: update a instead of z in ReLA, (2) ReLA with warm start (ReLA-warm start), and (3) action-based TD3 with warm start (TD3-warm start). Shaded areas represent standard errors across 5 seeds.

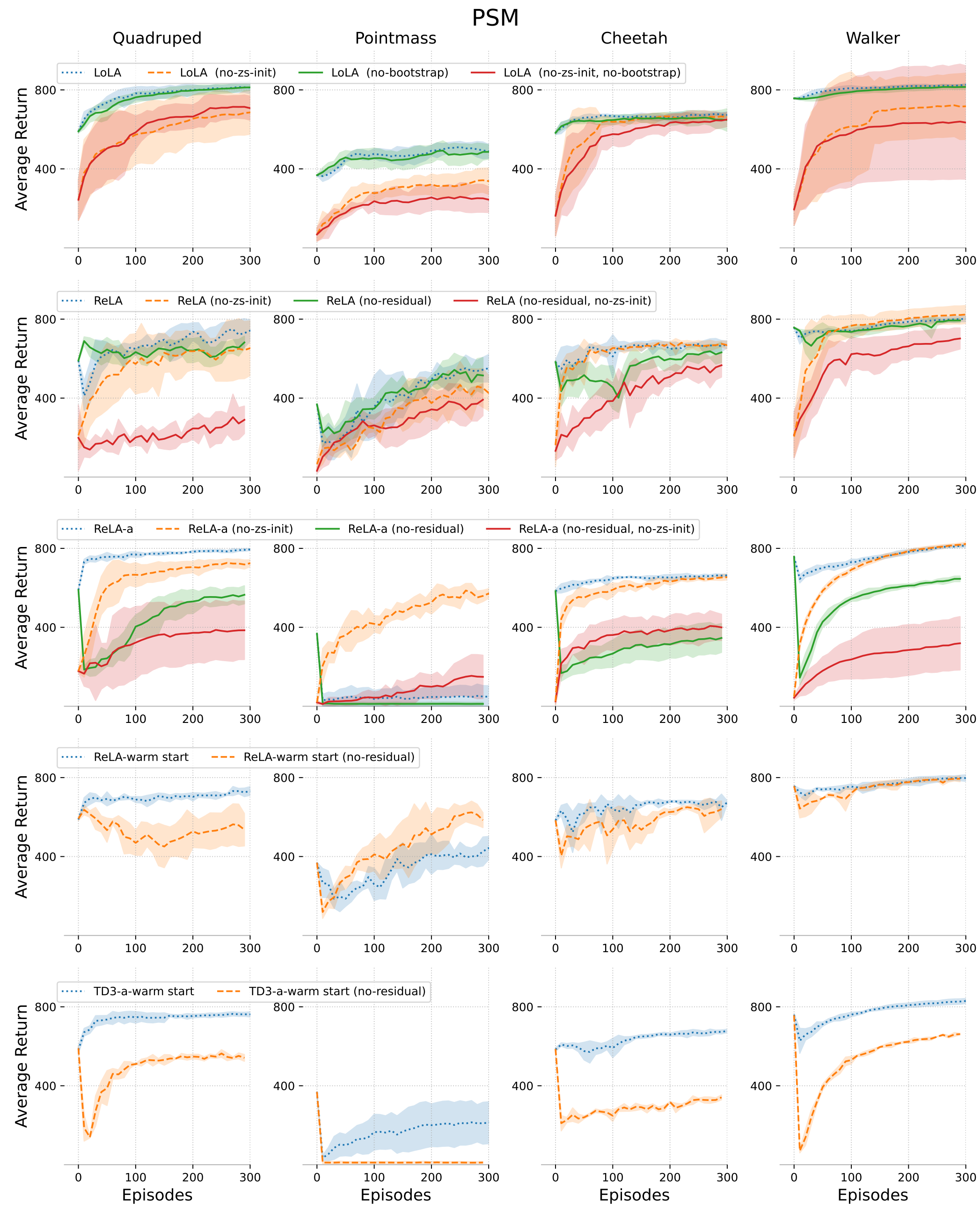

Figure 11: Ablation studies evaluating adaptations of PSM on four DMC tasks (Quadruped, Pointmass, Cheetah, Walker). Experiments include disabling zero-shot initialization ("no-I") and/or removing residual critics ("no-R") from LoLA, ReLA, and three additional variants: (1) ReLA-a: update action instead of z in ReLA, (2) ReLA with warm start (ReLA-warm start), and (3) action-based TD3 with warm start (TD3-warm start). Shaded areas represent standard errors across 5 seeds.

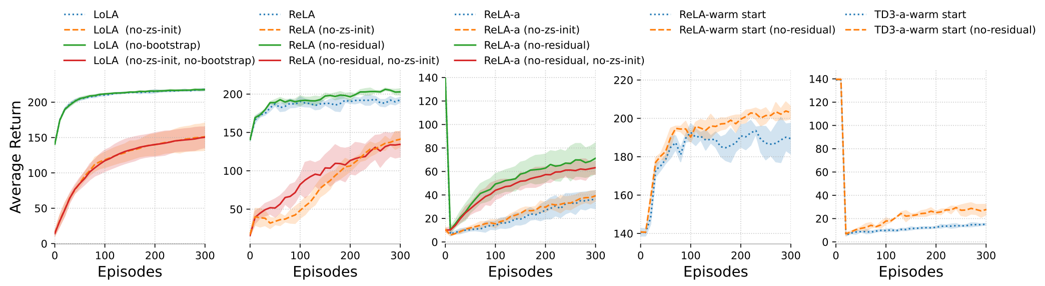

Figure 12: Ablation studies for adaptation with FB-CPR on 45 HumEnv tasks. Experiments include disabling zero-shot initialization ("no-I") and/or re-moving residual critics ("no-R") from LoLA, ReLA, and three additional variants: (1) ReLA-a: update action instead of z in ReLA, (2) ReLA with warm start (ReLA-warm start), and (3) action-based TD3 with warm start (TD3-warm start). Shaded areas represent standard errors across 5 seeds.