A TRUST REGION APPROACH FOR FEW-SHOT SIM-TO-REAL REINFORCEMENT LEARNING

Paul Daoudi[†] **Christophe Prieur**[‡] **Bogdan Robu**[‡] **Merwan Barlier**[†] **Ludovic Dos Santos**[§]
{paul.daoudi,merwan.barlier}@huawei.com
{christophe.prieur,bogdan.robu}@gipsa-lab.grenoble-inp.fr
l.dossantos@criteo.com

ABSTRACT

Simulation-to-Reality Reinforcement Learning (Sim-to-Real RL) seeks to use simulations to minimize the need for extensive real-world interactions. Specifically, in the few-shot off-dynamics setting, the goal is to acquire a simulator-based policy despite a dynamics mismatch that can be effectively transferred to the real-world using only a handful of real-world transitions. In this context, conventional RL agents tend to exploit simulation inaccuracies resulting in policies that excel in the simulator but underperform in the real environment. To address this challenge, we introduce a novel approach that incorporates a penalty to constrain the trajectories induced by the simulator-trained policy inspired by recent advances in Imitation Learning and Trust Region based RL algorithms. We evaluate our method across various environments representing diverse Sim-to-Real conditions, where access to the real environment is extremely limited. These experiments include high-dimensional systems relevant to real-world applications. Across most tested scenarios, our proposed method demonstrates performance improvements compared to existing baselines.

1 INTRODUCTION

Reinforcement Learning (RL) is often applied in simulation before deploying the learned policy on real systems (Ju et al., 2022; Muratore et al., 2019; Kaspar et al., 2020; Witman et al., 2019). This approach is considered to be one of the safest and most efficient ways of obtaining a near-optimal policy for complex systems (Jiang et al., 2021; Salvato et al., 2021; Hsu et al., 2023), as many of the challenges of applying RL to real-world systems (Dulac-Arnold et al., 2021) are mitigated. The agent can sample the simulator at will (Kamthe & Deisenroth, 2018; Schwarzer et al., 2021) without having to consider any safety constraints (Garcıa & Fernández, 2015; Achiam et al., 2017) during training.

However, simulators of complex systems are often inaccurate. Indeed, many physical laws such as contact forces, material elasticity, and fluid dynamics are difficult to model, leading simulators to rely on approximations (Koenig & Howard, 2004; Todorov et al., 2012). These small inaccuracies may accumulate over time, leading to increasing deviations between the simulated and real-world environments over time. Directly transferring a policy trained on simulators can therefore lead to unsatisfactory outcomes due to these compounding errors. Worse, modern optimization-based agents may exploit these discrepancies to find policies that perform exceptionally well in simulation but result in trajectories that are impossible to replicate in the real environment. This phenomenon - known as the *Simulation-to-Reality* (Sim-to-Real) gap (Höfer et al., 2021) - occurs in most simulators (Salvato et al., 2021).

In general, besides relying only on a simulator, it is still possible to deploy the agent in the environment to collect data. However, this deployment is limited due to safety and time considerations. As a result, the available data is often limited to a few narrow trajectories. Two orthogonal approaches are

[†]Huawei Noah's Ark Lab, Paris, France
[‡]GIPSA Lab, Grenoble, France
[§]Criteo AI Lab, Paris, France

possible to include this data in the derivation of the policy. The first one - well studied (Abbeel et al., 2006; Zhu et al., 2018; Desai et al., 2020a; Hanna et al., 2021) - leverages this data to improve the simulator, then learns a traditional RL agent on the upgraded simulator.

The second approach keeps the simulator fixed and biases the learning process to account for the dynamics discrepancies (Koos et al., 2012). This line of work is complementary to the one improving the simulator, as both could be combined to make the best use of the limited real-world samples. To the best of our knowledge, only a few works have taken this purely off-dynamics direction, and even fewer have focused on the low data regime scenario. Currently, the prominent approach is DARC (Eysenbach et al., 2020) which modifies the reward function to search for parts of the simulator that behave similarly to the real world. Although this method is effective for a few classes of problems, e.g. with the "broken" environments, we have found that it may fail drastically in others, limiting its application to a restrictive class of discrepancy between the simulator and the environment.

In this paper, we introduce the **F**ew-sh**O**t **O**ff **D**ynamics (FOOD) algorithm, a Trust Region method constraining the derived policy to be around the trajectories observed in the real environment. We theoretically justify this constraint, which directs the policy towards feasible trajectories in the real system, and thus mitigates the potential trajectory shifts towards untrustable regions of the simulator. Our constraint takes the form of a regularization between visitation distributions, we show that it can be practically implemented using state-of-the-art techniques from the Imitation Learning (IL) literature (Hussein et al., 2017). Our method is validated on a set of environments with multiple off-dynamics disparities. We show that, compared to other baselines, our approach is the most successful at taking advantage of the few available data. Our agent is also shown to be relevant for a wider range of dynamics discrepancies.

2 Related Work

Closing the Sim-to-Real gap for the dynamics shift is crucial for the successful deployment of RL-based policies in real-world systems. In this scenario, the algorithms trained on the simulator must have sufficient performance in the real world to have a tangible impact. This problem has been studied in two contexts, depending on the accessibility of the agent to transitions from the real environment. These settings are referred to as "Zero-Shot" and "Few-Shot" Sim-to-Real.

Zero-Shot Sim-to-Real RL Sampling data from real-world environments can be impossible due to strict safety constraints or time-consuming interactions. In such cases, simulators are used to ensure robustness (Morimoto & Doya, 2005; Moos et al., 2022) to guarantee a certain level of performance without sampling from the real system. It can take many forms. One possible choice is domain randomization (Mordatch et al., 2015; Tobin et al., 2017; Peng et al., 2018; Mehta et al., 2020) where relevant parts of the simulator are randomized to make it resilient to changes. Another line of work focuses on addressing the worst-case scenarios under stochastic simulator dynamics (Abdullah et al., 2019; Tanabe et al., 2022). Robustness can also be achieved w.r.t. actions (Jakobi et al., 1995; Pinto et al., 2017; Tessler et al., 2019), that arise when certain controllers become unavailable in the real environment. These techniques are outside the scope of this paper as they do not involve any external data in the learning process.

Few-Shot Sim-to-Real RL When data can be sampled from the real environment, two orthogonal approaches have been developed to bridge the Sim-to-Real gap. The first approach, well established, is to improve the accuracy of the simulator through various methods. If the simulator parameters are available, the simulator parameters can be optimized directly (Farchy et al., 2013; Zhu et al., 2018; Tan et al., 2018; Collins et al., 2021; Allevato et al., 2020; Du et al., 2021). Otherwise, expressive models can be used to learn the changes in dynamics (Abbeel et al., 2006; Saveriano et al., 2017; Lee et al., 2017; Golemo et al., 2018; Hwangbo et al., 2019). Within this category, a family of methods builds an action transformation mechanism that - when taken in simulation - produces the same transition that would have occurred in the real system (Hanna & Stone, 2017; Karnan et al., 2020; Desai et al., 2020b; Hanna et al., 2021). In particular, GARAT (Desai et al., 2020a) leverages recent advances in Imitation Learning from Observations (Torabi et al., 2018; 2019) to learn this action transformation and ground the simulator with only a few trajectories. All these algorithms are orthogonal to our work as once the simulator has been improved, a new RL agent has to be trained.

The second approach, more related to our work, is the line of inquiry that alters the learning process of the RL policy in the simulator to be efficient in the real environment. One group of approaches creates a policy - or policies - that can quickly adapt to a variety of dynamic conditions (Yu et al., 2018; Arndt et al., 2020; Yu et al., 2020; Kumar et al., 2021). It requires the ability to set the simulator parameters which may not always be feasible, e.g. if the simulator is a black box. A more general algorithm is DARC (Eysenbach et al., 2020). It learns two classifiers to distinguish transitions between the simulated and real environments and incorporates them into the reward function to account for the dynamics shift. Learning the classifiers is easier than correcting the dynamics of the simulator, but as we will see in the experiments, this technique seems to work mainly when some regions of the simulator accurately model the target environment and others don't. Another related work is H2O (Niu et al., 2022a) which extends the approach by considering access to a fixed dataset of transitions from a real environment. It combines the regularization of the offline algorithm CQL (Kumar et al., 2020; Yu et al., 2021; Daoudi et al., 2022) with the classifiers proposed by DARC. However, the performance of H2O is dependent on the amount of data available. In fact, it performed similarly, or worse, to the pure offline algorithm when only a small amount of data was available (Niu et al., 2022a, Appendix C.3).

3 Background

3.1 Preliminaries

Let $\Delta(\cdot)$ be the set of all probability measures on $(\cdot)$. The agent-environment interaction is modeled as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, r, P, \gamma, \rho_0)$, with a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition kernel $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, a reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [R_{\min}, R_{\max}]$, the initial state distribution ρ_0 and a discount factor $\gamma \in [0, 1)$. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ is a decision rule mapping a state over a distribution of actions. The value of a policy π is measured through the value function $V_P^\pi(s) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) | s_0 = s\right]$. The objective is to find the optimal policy maximizing the expected cumulative rewards $J_P^\pi = \mathbb{E}_{\rho_0}\left[V_P^\pi(s)\right]$. We also define the Q-value function $Q_P^\pi(s, a) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) | s_0 = s, a_0 = a\right]$ and the advantage value function $A_P^\pi(s, a) = Q_P^\pi(s, a) - V_P^\pi(s)$. Finally, let $d_P^\pi(s) = (1-\gamma)\mathbb{E}_{\rho_0,\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{P}\left(s_t = s\right)\right]$, $\mu_P^\pi(s, a) = (1-\gamma)\mathbb{E}_{\rho_0,\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{P}\left(s_t = s, a_t = a\right)\right]$ and $\nu_P^\pi(s, a, s') = (1-\gamma)\mathbb{E}_{\rho_0,\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{P}\left(s_t = s, a_t = a, s_{t+1} = s'\right)\right]$ the state, state-action and state-action-state visitation distributions. All these quantities are expectations w.r.t. both the policy and the transition probabilities.

The Sim-to-Real problem involves two MDPs: the source simulator $\mathcal{M}_{\text{s}}$ and the target environment $\mathcal{M}_{\text{t}}$. We hypothesize that the simulator and the real world are identical except for their transition probabilities $P_{\text{s}} \neq P_{\text{t}}$. Our hypothesis states that while most of the MDP parameters are known, the underlying physics of the environment is only estimated. This is a common setting known as off-dynamics Reinforcement Learning. It encapsulates many real-world applications: a model of the dynamics may have been previously learned, or practitioners may have created a simulator based on a simplification of the system's physics. We do not assume access to any parameter modifying the transition probabilities P_{s} to encompass black-box simulators. For readability purposes, we drop the P subscripts for the value functions as they are always associated with the source simulator $\mathcal{M}_{\text{s}}$.

Many Few-Shot Sim-to-Real agents (Abbeel et al., 2006; Desai et al., 2020a; Hanna et al., 2021) typically employ the following procedure to handle complex environments. First, the policy and value functions are initialized in the simulator. The choice of objective at this stage can vary, although a classical approach is to solely maximize the rewards of the simulator. At each iteration, the policy is verified by experts. If it is deemed safe, N trajectories are gathered in the real environment and saved in a replay buffer $\mathcal{D}_{\text{t}}$. These trajectories are then used to potentially correct the simulator and/or the training objective and induce a new policy. This process is repeated until a satisfactory policy is found. This setup is time-consuming and may be risky even when the policy is verified by experts, hence the need to learn with as few data as possible from the real environment.

This work focuses on how to best modify the objective with few trajectories. If handled properly, this could reduce the number of interactions required by the whole process overcoming the need to build a perfect simulator. For the purpose of our study, we assume that $\mathcal{M}_{\text{t}}$ remains fixed throughout the process.

3.2 Trust Region algorithms and the visitation distribution constraint

Due to their efficiency and stability, Trust Region algorithms (Schulman et al., 2015; 2017; Kumar et al., 2020) have shown strong efficiency in various RL settings. Many of them are based on Kakade & Langford (2002), where an iteration scheme improves the policy by maximizing an estimate of the Advantage function within a Trust Region (Yuan, 2000) around the previous policy. This process has been extended by refining the Trust Region (Terpin et al., 2022; Moskovitz et al., 2020), for example by introducing a behavior b_P^π (Mouret, 2011) that encapsulates any additional property of the MDP (Pacchiano et al., 2020; Touati et al., 2020).

This family of algorithms is formalized as follows. A policy and a value function are parametrized with respective weights $\theta \in \Theta$ and $\omega \in \Omega$, that we denote from now on π_θ and $V_\omega^{\pi_\theta}$. At each iteration k, the policy is improved using the advantage function built from the approximated value function $V_{\omega_k}^{\pi_{\theta_k}}$ while ensuring that the policy remains in the Trust Region:

$$\begin{aligned} \underset{\theta \in \Theta}{\text{maximize}} \quad & \mathbb{E}_{s\sim d_P^{\pi_{\theta_k}}(\cdot), a\sim\pi_\theta(\cdot|s)} \left[A_{\omega_k}^{\pi_{\theta_k}}(s,a) \right] \\ \text{subject to} \quad & D\Big(b_P^{\pi_\theta} \,\Big\|\, b_P^{\pi_{\theta_k}} \Big) \leq \epsilon_k, \end{aligned} \tag{1}$$

where D is any kind of similarity metric and ϵ_k is a hyper-parameter. TRPO (Schulman et al., 2015; Shani et al., 2020) can be retrieved with $b_P^\pi = \pi$ and by setting D to be the Kullback-Leibler (KL) divergence. Alternative behavior options can be found in (Pacchiano et al., 2020; Touati et al., 2020; Moskovitz et al., 2020). In particular, Touati et al. (2020) proposed to encapsulate the whole trajectories induced by π and P by setting $b_P^\pi = d_P^\pi$. It resulted in better results both in terms of sample efficiency and final cumulative rewards than most of its counterparts. This is natural as the new constraint between the state visitation distributions takes the whole trajectories induced by the policy into account, providing more information than the policy alone.

4 Few-Shot Off Dynamics Reinforcement Learning

In this section, we propose a new objective to better transfer a policy learned in simulation to the real environment. We extend the Trust Region Policy Optimization objective to the off-dynamics setting. Then, we remind necessary results on Imitation Learning (IL) before deriving our practical algorithm **F**ew-sh**O**t **O**ff **D**ynamics (FOOD) Reinforcement Learning.

4.1 A new Sim-to-Real Trust Region objective

Given the discrepancies between the dynamics of the simulator and the real environment, applying the same policy to both environments may result in different trajectories. This poses a challenge as the agent may make the most of these differences to find policies that produce excellent trajectories in the simulator but are impossible to replicate in the real system.

We first analyze the difference between the objectives J_P^π associated with the target and source environments, depending on a metric between visitation distributions. For this, we apply the tools from traditional Trust Region methods (Pirotta et al., 2013; Schulman et al., 2015; Achiam et al., 2017) to the Sim-to-Real setting, and propose the following lower bound.

Proposition 4.1 *Let $J_P^\pi = \mathbb{E}_{\rho_0}[V_P^\pi(s)]$ the expected cumulative rewards associated with policy π, transitions P and initial state distribution ρ_0. For any policy π and any transition probabilities P_t and P_s, the following holds:*

$$J_{P_t}^\pi \geq J_{P_s}^\pi - \frac{2R_{max}}{1-\gamma} D_{TV}\big(\nu_{P_s}^\pi, \nu_{P_t}^\pi\big), \tag{2}$$

with D_{TV} the Total Variation distance $D_{TV}\big(\nu_{P_s}^\pi, \nu_{P_t}^\pi\big) = \sup_{s,a,s'} \left| \nu_{P_s}^\pi(s,a,s') - \nu_{P_t}^\pi(s,a,s') \right|$.

We defer the proof to Appendix A. It illustrates how the performance of the optimal policy in the real environment may differ from that of the simulator due to the metric $D_{\text{TV}}\big(\nu_{P_s}^\pi, \nu_{P_t}^\pi\big)$ quantifying their difference in trajectories. This is all the more important given that this term is exacerbated by the

factor $\frac{2R_{\max}}{1-\gamma}$. Finally, we note that the Total Variation distance could be replaced by the Kullback-Leibler divergence or by the Jensen-Shannon divergence using Pinsker's inequality (Csiszar & Körner, 1981) or the one in (Corander et al., 2021, Proposition 3.2), provided the minimal assumptions of having a finite state-action space and the absolute continuity of the considered measures.

Overall, this lower bound highlights a good transfer between the source and target environment is possible when $D_{\text{TV}}\left(\nu^{\pi}_{P_\text{s}}, \nu^{\pi}_{P_\text{t}}\right)$ is small, as it induces similar objectives J^{π}_{P}. Inspired by this insight, we adapt Trust Region methods to the Sim-to-Real setting (Pirotta et al., 2013; Schulman et al., 2015; Achiam et al., 2017) and propose a new constraint between trajectories by setting the behaviors b^{π}_{P} to be the state-action-state visitation distribution respectively associated with the transition probabilities of the source and target environment ν^{π}_{P}:

$$\begin{aligned} \underset{\theta \,\in\, \Theta}{\text{maximize}} \quad & \mathbb{E}_{s\sim d^{\pi_{\theta_k}}_{P_\text{s}}(\cdot), a\sim\pi_\theta(\cdot|s)}\left[A^{\pi_{\theta_k}}_{\omega_k}(s,a)\right] \\ \text{subject to} \quad & D\left(\nu^{\pi_\theta}_{P_\text{s}} \,\middle\|\, \nu^{\pi_{\theta_k}}_{P_\text{t}}\right) \leq \epsilon_k. \end{aligned} \tag{3}$$

The new constraint ensures that the policy is optimized for trajectories that are feasible in the real world, thus preventing the RL agent from exploiting any potential hacks that may exist in the simulator. In addition, remaining close to the data sampled from the real world can be beneficial when the simulator has been constructed using that data, as querying out-of-distribution data can yield poor results (Kang et al., 2022).

Unfortunately, the difference between the transition probabilities makes the constraint in Equation 3 difficult to compute. The previous work of Touati et al. (2020) addressed this by restricting D to f-divergences $D_f\left(\mu^{\pi_\theta}_{P} || \mu^{\pi_{\theta_k}}_{P}\right) = \mathbb{E}_{(s,a)\sim\pi_\theta}\left[f\left(\frac{\mu^{\pi_\theta}_{P}}{\mu^{\pi_{\theta_k}}_{P}}\right)\right]$ and by considering state-action visitation distributions. They Touati et al. (2020) used the DualDICE algorithm (Nachum et al., 2019) to directly estimate the relaxed ratio $\frac{\mu^{\pi_\theta}_{P}}{\mu^{\pi_{\theta_k}}_{P}}$ for any policy π_θ sufficiently close to π_{θ_k}, eliminating the need to sample data for each policy. However, this method is not applicable to our setting because DualDICE relies on a modified joined Bellman operator, which assumes that both distributions follow the same transition probabilities. Another solution would be to collect at least one trajectory per update. While this would not pose any safety concerns for the data would be sampled in the simulator, it can be time-consuming in practice.

4.2 Practical algorithm

In order to devise a practical algorithm for addressing Equation 3, we solve the penalized problem

$$\underset{\theta \,\in\, \Theta}{\text{maximize}} \quad \mathbb{E}_{s\sim d^{\pi_{\theta_k}}_{P_\text{s}}(\cdot), a\sim\pi_\theta(\cdot|s)}\left[A^{\pi_{\theta_k}}_{\omega_k}(s,a) + \alpha\, \mathcal{R}(s,a)\right], \tag{4}$$

where the regularization $\mathcal{R}(s,a)$ serves as a proxy for minimizing the divergence $D\left(\nu^{\pi_\theta}_{P_\text{s}} \,\middle\|\, \nu^{\pi_{\theta_k}}_{P_\text{t}}\right)$ and α is a hyper-parameter.

To construct a relevant proxy, we leverage the recent results from Imitation Learning (IL) (Hussein et al., 2017) that we briefly recall in this paragraph. In this field, the agent aims to reproduce an expert policy π_e using limited data sampled by that expert in the same MDP with generic transition probabilities P. Most current algorithms tackle this problem by minimizing a certain similarity metric D between the learning policy's state-action visitation distribution $\mu^{\pi_\theta}_{P}$ and the expert's $\mu^{\pi_e}_{P}$. The divergence minimization problem is transformed into a reward r_{imit} maximization one, resulting in an imitation value function $V^{\pi}_{\text{imit}} = \mathbb{E}_{\pi,P_\text{s}}\left[\sum_{t=0}^{\infty}\gamma^t r_{\text{imit}}(s_t,a_t,s_{t+1})|s_0=s\right]$. Since these algorithms are based on data, they can be used to minimize the chosen similarity metric D between two state-action-state visitation distributions with different transition probabilities. Applied to our setting, this is formalized as:

$$\arg\max_{\pi} V^{\pi}_{\text{imit}} = \arg\min_{\pi} D\left(\nu^{\pi_\theta}_{P_\text{s}} \,\middle\|\, \nu^{\pi_{\theta_k}}_{P_\text{t}}\right). \tag{5}$$

The choices for the divergence D are numerous, leading to different IL algorithms (Ho & Ermon, 2016; Fu et al., 2017; Xiao et al., 2019; Dadashi et al., 2020), some of which are summarized in Table 1.

GAIL	AIRL	PWIL
$D_{\text{JS}}\left(X_{P_\text{s}}^{\pi_\theta} \middle\| X_{P_\text{t}}^{\pi_{\theta_k}}\right)$	$D_{\text{KL}}\left(X_{P_\text{s}}^{\pi_\theta} \middle\| X_{P_\text{t}}^{\pi_{\theta_k}}\right)$	$D_{\text{W}}\left(X_{P_\text{s}}^{\pi_\theta}, X_{P_\text{t}}^{\pi_{\theta_k}}\right)$

Table 1: Objective function for well-known Imitation Learning (IL) algorithms. The variable X can be chosen as either d, μ, or ν. Other IL agents can be found in (Ghasemipour et al., 2020).

These IL techniques enable efficient estimation of this value function using a small number of samples from $d_{P_\text{t}}^{\pi_{\theta_k}}$ and unlimited access to $\mathcal{M}_\text{s}$. Let $\xi \in \Xi$ be the weights of this parametrized value function. The new regularization is $\mathcal{R}(s,a) = A_{\text{imit}}^{\pi_{\theta_k},\xi_k}(s,a)$, which can be learned with any suitable IL algorithm.

This new agent is quite generic as it could be optimized with different divergences. It takes as input an online RL algorithm (Babaeizadeh et al., 2016; Schulman et al., 2017) denoted $\mathcal{O}$ and an Imitation Learning algorithm denoted $\mathcal{I}$. The whole Sim-to-Real algorithm process, which we denote Few-shOt Off Dynamics (FOOD) RL, is described as follows. First, the policy and the value weights are initialized in the simulator with $\mathcal{O}$. At each iteration k, the agent samples N new trajectories with π_{θ_k}[1]. Subsequently, the policy, traditional, and imitation value functions are retrained on the simulator with $\mathcal{O}$ and $\mathcal{I}$ according to Equation 4. The whole algorithm is summarized in Algorithm 1.

Algorithm 1 Few-shOt Off Dynamics (FOOD)

Input: Algorithms $\mathcal{O}$ and $\mathcal{I}$
Initialize policy and value weights θ_0 and ω_0 with $\mathcal{O}$
Randomly initialize the weights ξ_0
for $k \in (0, \dots, K-1)$ **do**
  Gather N trajectories $\{\tau_i, \dots, \tau_N\}$ with π_{θ_k} on the real environment $\mathcal{M}_t$ and add them in $\mathcal{D}_\text{t}$
  Remove trajectories that lead to drastic failures
  Learn the value function weights ω_{k+1} with $\mathcal{O}$ in the source environment $\mathcal{M}_\text{s}$
  Learn the imitation value function weights ξ_{k+1} with $\mathcal{I}$ in $\mathcal{M}_\text{s}$ using $\mathcal{D}_\text{t}$
  Learn the policy maximizing equation 4 using $\mathcal{D}_\text{t}$ and $\mathcal{M}_\text{s}$ with $\mathcal{O}$
end for

5 Experiments

In this section, we evaluate the performance of the FOOD algorithm in the off-dynamics setting in environments presenting different dynamics discrepancies, treated as black box simulators. These environments are based on Open AI Gym (Brockman et al., 2016) and the Minitaur environment (Coumans & Bai, 2016–2021) where the target environment has been modified by various mechanisms. These include gravity, friction, and mass modifications, as well as broken joint(s) systems for which DARC is known to perform well (Eysenbach et al., 2020, Section 6). We also add the Low Fidelity Minitaur environment, highlighted in previous works (Desai et al., 2020a; Yu et al., 2018) as a classical benchmark for evaluating agents in the Sim-to-Real setting. In this benchmark, the source environment has a linear torque-current relation for the actuator model, and the target environment - proposed by Tan et al. (2018) - uses accurate non-linearities to model this relation.

All of our FOOD experiments were carried out using both GAIL (Ho & Ermon, 2016), a state-of-the-art IL algorithm, as $\mathcal{I}$. We found that GAIL performed similarly, or better than other IL algorithms such as AIRL (Fu et al., 2017) or PWIL (Dadashi et al., 2020). FOOD is tested with its theoretically motivated metric between state-action-state visitation distributions ν_P^π, as well as with d_P^π and μ_P^π for empirically analyzing the performance associated with the different visitation distributions. We found that GAIL performed similarly, or better than other IL algorithms such as AIRL (Fu et al., 2017) or PWIL (Dadashi et al., 2020). The performance of our agent with the different IL algorithms can be found in Appendix C.5. We compare our approach against various baselines modifying the RL

[1]These trajectories could first be used to improve the simulator.

objective, detailed below. They cover current domain adaptation, robustness, or offline Reinforcement Learning techniques applicable to our setting. Further details of the experimental protocol can be found in Appendix C.

- **DARC** (Eysenbach et al., 2020) is our main baseline. It is a state-of-the-art off-dynamics algorithm that introduces an additional importance sampling term in the reward function to cope with the dynamics shift. In practice, this term is computed using two classifiers that distinguish transitions from the simulated and the real environment. In this agent, an important hyper-parameter is the standard deviation σ_{DARC} of the centered Gaussian noise injected into the training data to stabilize the classifiers (Eysenbach et al., 2020, Figure 7). DARC was originally optimized with SAC (Haarnoja et al., 2018) but to allow a fair comparison with FOOD, we re-implemented DARC in the same RL algorithm that is used in our method, drawing inspiration from the open-source code (Niu et al., 2022b).
- **Action Noise Envelope (ANE)** (Jakobi et al., 1995) is a robust algorithm that adds a centered Gaussian noise with standard deviation σ_{ANE} to the agent's actions during training. Although simple, this method outperformed other robustness approaches in recent benchmarks (Desai et al., 2020a) when the simulator is a black box.
- **CQL** (Kumar et al., 2020) is an offline RL algorithm that learns a policy using real-world data. It does not leverage the simulator in its learning process, which should make its resulting policy sub-optimal. This algorithm inserts a regularization into the Q-value functions, with a strength β. We use Geng (2021) to run the experiments.
- We also consider two RL agents, $\mathbf{RL_{Sim}}$ trained solely on the simulator (without access to any real-world data) and $\mathbf{RL_{Real}}$ trained solely on the real-world environment. Both algorithms were trained to convergence. Even though the latter baselines do not fit in the off-dynamics setting they give a rough idea of how online RL algorithms would perform in the real environment. The online RL algorithm $\mathcal{O}$ depends on the environment: we use A2C (Babaeizadeh et al., 2016) for Gravity Pendulum and PPO (Schulman et al., 2017) for the other environments.

Experimental protocol Our proposed model and corresponding off-dynamics/offline baselines require a batch of real-world data. To provide such a batch of data, we first train a policy and a value function of the considered RL agent until convergence by maximizing the reward on the simulated environment. After this initialization phase, 5 trajectories are sampled from the real environment to fit the restricted real data regime. They correspond to 500 data points for Pendulum and 1000 data points for the other environments. If some trajectories perform poorly in the real environment, we remove them for FOOD, DARC, and CQL to avoid having a misguided regularization. FOOD, DARC, and ANE are trained for 5000 epochs in the simulated environment. Both RL_{Sim} and RL_{Real} are trained until convergence. CQL is trained for 100000 gradient updates for Gravity Pendulum and 500000 gradient updates for all other environments. All algorithms are averaged over 4 different random seeds. Additional details can be found in Appendix C.

Hyperparameters Optimization We optimize the hyperparameters of the evaluated algorithms through a grid search for each different environment. Concerning DARC and ANE, we perform a grid search over their main hyper-parameter $\sigma_{\text{DARC}} \in \{0.0, 0.1, 0.5, 1\}$ and $\sigma_{\text{ANE}} \in \{0.1, 0.2, 0.3, 0.5\}$. The remaining hyperparameters were set to their default values according to the experiments reported in the open-source code (Niu et al., 2022b). For CQL, we perform a grid search over the regularization strength $\beta \in \{5, 10\}$, otherwise we keep the original hyper-parameters of Geng (2021). For RL_{Real} and RL_{Sim} we used the default parameters specific to each environment according to Kostrikov (2018) and trained them over 4 different seeds. We then selected the seed with the best performance. For our proposed algorithm FOOD, the regularization strength hyperparameter α is selected over a grid search depending on the underlying RL agent, $\alpha \in \{0, 1, 5, 10\}$ for A2C and $\alpha \in \{0.5, 1, 2, 5\}$ for PPO. This difference in choice is explained by the fact that the advantages are normalized in PPO, giving a more refined control over the regularization weight.

Results We monitor the evolution of the agents' performance by evaluating their average return $\mathcal{R}$ in the real environment during training. Note that we do not gather nor use the data from those evaluations in the learning process since we work under a few-shot framework. We compute the return of all methods averaged over 4 seeds, where we show the standard deviation being divided

by two for readable purposes. In all figures, the x-axis represents the number of epochs where each epoch updates the policy and value functions with 8 different trajectories from the simulator.

5.1 Comparison between the different agents

We evaluate the mentioned algorithms on the proposed environments. These experiments provide an overview of the efficiency of the different objectives in finetuning the policy, given reasonably good trajectories. Results are summarized in Table 2, where we also report the median of the normalized average return (NAR) $\frac{J_{P_t}^{\pi_{\text{agent}}} - J_{P_t}^{\pi_{\text{RL}_{\text{Sim}}}}}{J_{P_t}^{\pi_{\text{RL}_{\text{Real}}}} - J_{P_t}^{\pi_{\text{RL}_{\text{Sim}}}}}$ (Desai et al., 2020a) as well as the median of the NAR's standard deviations. The associated learning curves and the NAR per environment can be found in Appendix C.2.

Environment	RL_{Sim}	RL_{Real}	CQL	ANE	DARC	FOOD (Ours)		
						μ_P^π	d_P^π	ν_P^π
Gravity Pendulum	-1964 ± 186	-406 ± 22	-1683 ± 142	-2312 ± 11	-3511 ± 865	-2224 ± 43	$\mathbf{-485 \pm 54^*}$	-2327 ± 14
Broken Joint Cheetah	1793 ± 1125	5844 ± 319	143 ± 104	3341 ± 132	2553 ± 405	$\mathbf{3801 \pm 155}$	$\mathbf{3888 \pm 201}$	$\mathbf{3950 \pm 97^*}$
Broken Joint Minitaur	7.4 ± 4.1	20.8 ± 4.8	0.25 ± 0.09	7.8 ± 6	$\mathbf{12.9 \pm 2}$	$\mathbf{14.9 \pm 3}$	$\mathbf{13.6 \pm 3.8}$	$\mathbf{16.6 \pm 4.7^*}$
Heavy Cheetah	3797 ± 703	11233 ± 1274	41 ± 34	$\mathbf{7443 \pm 330^*}$	3956 ± 1314	4876 ± 181	4828 ± 553	4743 ± 297
Broken Joint Ant	5519 ± 876	6535 ± 352	1042 ± 177	3231 ± 748	5041 ± 364	$\mathbf{6145 \pm 98}$	5547 ± 204	$\mathbf{6179 \pm 86^*}$
Friction Cheetah	1427 ± 674	9455 ± 3554	-466.4 ± 13	$\mathbf{6277 \pm 1405^*}$	3064 ± 774	3890 ± 1495	3212 ± 2279	3852 ± 733
Low Fidelity Minitaur	8.9 ± 5.8	27.1 ± 8	10.2 ± 1	6.4 ± 3	3.2 ± 1.8	$\mathbf{17 \pm 2}$	$\mathbf{15.7 \pm 2.8}$	$\mathbf{17.9 \pm 0.8^*}$
Broken Leg Ant	1901 ± 981	6430 ± 451	830 ± 8	$\mathbf{2611 \pm 220}$	$\mathbf{2336 \pm 565}$	$\mathbf{2652 \pm 356}$	$\mathbf{2345 \pm 806}$	$\mathbf{2733 \pm 719^*}$
Median NAR and std	0 ; 0.25	1 ; 0.26	-0.32 ; 0.02	0.1 ; 0.11	0.06 ; 0.13	**0.37 ; 0.09**	**0.29 ; 0.17**	**0.40 ; 0.06***

Table 2: Returns over 4 seeds of the compared methods on benchmark settings. The best agent w.r.t. the mean is highlighted with boldface and an asterisk. We perform an unpaired t-test with an asymptotic significance of 0.1 w.r.t. the best performer and highlight with boldface the ones for which the difference is not statistically significant.

All the experiments clearly demonstrate the insufficiency of training traditional RL agents solely on the simulator. The optimal policy for the simulator is far from optimal for the real world as we observe a large drop in performance from RL_{Real} to RL_{Sim} on all benchmarked environments. For example, the RL_{Sim} exploits the linear torque-current relation in Low Fidelity Minitaur and fails to learn a near-optimal policy for the real environment. Furthermore, RL_{Sim} often exhibits a large variance in real environments as it encounters previously unseen situations. This is welcome as relevant trajectories are gathered to guide the agent in the simulation.

Overall, we can see that our algorithm FOOD exhibits the best performances across all considered environments against all other baselines, whether it is constrained by state, state-action or state-action-state visitation distributions. Two exceptions are on Heavy and Friction Cheetah where ANE has very good results. We also note that FOOD with its theoretically motivated regularization between the state-action-state visitation distribution provides the best results with a lower variance.

In addition, we find that the prominent baseline DARC is not efficient in all the use cases. It seems to be particularly good at handling sharp dynamics discrepancies, e.g. when one or two joints are broken but struggles for more subtle differences. In fact, it deteriorates over the naive baseline RL_{Sim} by a large margin for the Gravity Pendulum and the Low Fidelity Minitaur environments. This may be explained by their reward modification Δ_r (see Appendix B.1) which prevents the agent from entering dissimilar parts of the simulator but seems unable to handle simulators with a slight global dynamics mismatch. Even when DARC improves over RL_{Sim}, our algorithm FOOD is able to match or exceed its performance. The robust agent ANE is a strong baseline in most environments but may degrade the performance of traditional RL agents, as seen in Low Fidelity Minitaur, Broken Joint Ant, and Gravity Pendulum. CQL did not provide any good results, except on Gravity Pendulum and Low Fidelity Minitaur, but this was to be expected given the few real trajectories the agent has access to. Finally, we note that the three agents FOOD, DARC, and ANE often reduce the variance originally presented in RL_{Sim}.

We attribute FOOD's success to its ability to force the simulator to improve the rewards of the simulator along real-world trajectories. Its regularization seems to be efficient even in the low data regime we are studying.

5.2 HYPERPARAMETER SENSITIVITY ANALYSIS

We previously reported the results of the best hyper-parameters of the different methods. In practice, it is important to have a robust range of hyper-parameters in which the considered method performs well. Indeed, to the best of our knowledge, there currently exists no accurate algorithm for selecting such hyper-parameters in a high dimensional environment when the agent has access to limited data gathered with a different policy (Fu et al., 2020). In this section, we detail the sensitivity of FOOD associated with PPO and GAIL to its hyper-parameter α in 3 environments. They were specifically chosen to illustrate the relevant range for α. FOOD's complete hyper-parameter sensitivity analysis, as well as the one of DARC and ANE, can respectively be found in Appendix C.4, Appendix C.6 and Appendix C.7.

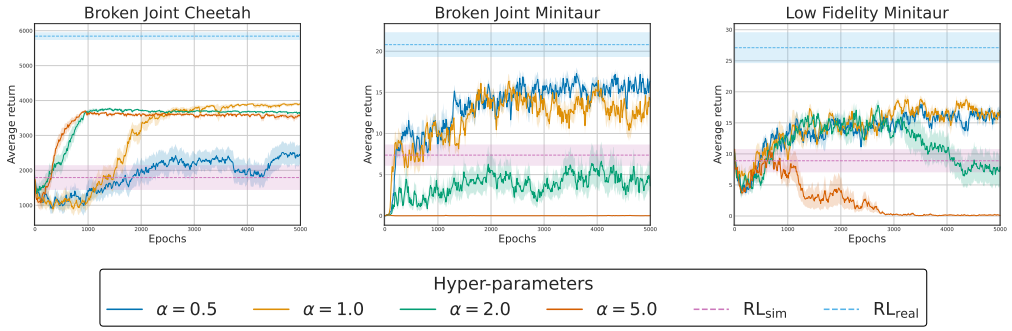

Figure 1: Hyperparameter sensibility analysis for FOOD on three environments.

The hyper-parameter α controls the strength of the regularization in FOOD. If it is too low, the agent will focus mainly on maximizing the rewards of the simulator and becomes very close to the naive baseline RL_{Sim}. This can be seen in Broken Joint Cheetah. On the other hand, setting α to a high value may induce the agent to solely replicate the trajectories from the real environment, in which case the agent may also be close to RL_{Sim}. Even worse, a hard regularization may degrade over RL_{Sim}, as shown in Low Fidelity Minitaur for $\alpha = 5$. However, 5 is an extremely high value as advantages are normalized in PPO, and this may increase the gradient too much and corrupt learning.

In any case, we have found that FOOD provides the best results when the regularization has the same scale as the traditional objective. This is also verified for the environments not displayed in this sub-section. We conclude that FOOD is relatively robust to this range of hyper-parameter, and recommend using PPO with α close to 1, a natural choice given that PPO normalizes its advantage functions.

6 CONCLUSION

In this work, we investigated different objectives to optimize a policy in different few-shot off-dynamics Sim-to-Real scenarios, including the state-of-the-art method DARC. We found that these objectives are either too simplistic or unable to cope with complex dynamics discrepancies, thereby limiting their application to real-world systems. To address this challenge, we introduced a novel trust region objective along with a practical algorithm leveraging imitation learning techniques. Through experimentations in different Sim-to-Real use cases, we have shown that our approach often outperforms the existing methods and seems to be more robust to dynamics changes. Our results emphasize the importance of leveraging a few real-world trajectories for a proper simulation-to-reality transfer with a well-defined objective.

Our agent could also benefit from new advances in the Imitation Learning literature to gain control in building its Trust Region. Finally, this Trust Region can be useful when the simulator has been improved using the available real-world trajectories as it avoids querying the simulator for Out-of-Distribution samples. This will be the primary focus of our future work.

REFERENCES

Pieter Abbeel, Morgan Quigley, and Andrew Y Ng. Using inaccurate models in reinforcement learning. In *Proceedings of ICML*, pp. 1–8, 2006.

Mohammed Amin Abdullah, Hang Ren, Haitham Bou Ammar, Vladimir Milenkovic, Rui Luo, Mingtian Zhang, and Jun Wang. Wasserstein robust reinforcement learning. *arXiv preprint arXiv:1907.13196*, 2019.

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of ICML*, pp. 22–31. PMLR, 2017.

Adam Allevato, Elaine Schaertl Short, Mitch Pryor, and Andrea Thomaz. Tunenet: One-shot residual tuning for system identification and sim-to-real robot task transfer. In *Proceedings of CoRL*, pp. 445–455. PMLR, 2020.

Karol Arndt, Murtaza Hazara, Ali Ghadirzadeh, and Ville Kyrki. Meta reinforcement learning for sim-to-real domain adaptation. In *Proceedings of ICRA*, pp. 2725–2731. IEEE, 2020.

Mohammad Babaeizadeh, Iuri Frosio, Stephen Tyree, Jason Clemons, and Jan Kautz. Reinforcement learning through asynchronous advantage actor-critic on a gpu. In *Proceedings of ICLR*, 2016.

Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, TB Dhruva, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. In *Proceedings of ICLR*, 2018.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Jack Collins, Ross Brown, Jürgen Leitner, and David Howard. Traversing the reality gap via simulator tuning. In *Proceedings of ACRA*. Australian Robotics and Automation Association (ARAA), 2021.

Jukka Corander, Ulpu Remes, and Timo Koski. On the jensen-shannon divergence and the variation distance for categorical probability distributions. *Kybernetika*, 2021.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

Imre Csiszar and J Körner. Coding theorems for discrete memoryless systems. In *Information Theory*. Akadémiai Kiadó (Publishing House of the Hungarian Academy of Sciences), 1981.

Robert Dadashi, Léonard Hussenot, Matthieu Geist, and Olivier Pietquin. Primal wasserstein imitation learning. *arXiv preprint arXiv:2006.04678*, 2020.

Paul Daoudi, Ludovic Dos Santos, Merwan Barlier, and Aladin Virmaux. Density estimation for conservative q-learning. In *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*, 2022.

Siddharth Desai, Ishan Durugkar, Haresh Karnan, Garrett Warnell, Josiah Hanna, and Peter Stone. An imitation from observation approach to transfer learning with dynamics mismatch. *Proceedings of NeurIPS*, 33:3917–3929, 2020a.

Siddharth Desai, Haresh Karnan, Josiah P Hanna, Garrett Warnell, and Peter Stone. Stochastic grounded action transformation for robot learning in simulation. In *Proceedings of IROS*, pp. 6106–6111. IEEE, 2020b.

Yuqing Du, Olivia Watkins, Trevor Darrell, Pieter Abbeel, and Deepak Pathak. Auto-tuned sim-to-real transfer. In *Proceedings of ICRA*, pp. 1290–1296. IEEE, 2021.

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

Benjamin Eysenbach, Swapnil Asawa, Shreyas Chaudhari, Sergey Levine, and Ruslan Salakhutdinov. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. *arXiv preprint arXiv:2006.13916*, 2020.

Alon Farchy, Samuel Barrett, Patrick MacAlpine, and Peter Stone. Humanoid robots learning to walk faster: From the real world to simulation and back. In *Proceedings of AAMAS*, pp. 39–46, 2013.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *Proceedings of ICLR*, 2017.

Justin Fu, Mohammad Norouzi, Ofir Nachum, George Tucker, Alexander Novikov, Mengjiao Yang, Michael R Zhang, Yutian Chen, Aviral Kumar, Cosmin Paduraru, et al. Benchmarks for deep off-policy evaluation. In *Proceedings of ICLR*, 2020.

Tanmay Gangwani. Airl code. https://github.com/tgangwani/RL-Indirect-imitation/tree/master, 2021.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *JMLR*, 16(1):1437–1480, 2015.

Xinyang (Young) Geng. Code for conservative q learning for offline reinforcement learning. https://github.com/young-geng/CQL, 2021.

Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In *Proceedings of CoRL*, pp. 1259–1277. PMLR, 2020.

Florian Golemo, Adrien Ali Taiga, Aaron Courville, and Pierre-Yves Oudeyer. Sim-to-real transfer with neural-augmented robot simulation. In *Proceedings of CoRL*, pp. 817–828. PMLR, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of ICML*, pp. 1861–1870. PMLR, 2018.

Josiah Hanna and Peter Stone. Grounded action transformation for robot learning in simulation. In *Proceedings of AAAI*, volume 31, 2017.

Josiah P Hanna, Siddharth Desai, Haresh Karnan, Garrett Warnell, and Peter Stone. Grounded action transformation for sim-to-real reinforcement learning. *Machine Learning*, 110(9):2469–2499, 2021.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Proceedings of NeurIPS*, 29, 2016.

Sebastian Höfer, Kostas Bekris, Ankur Handa, Juan Camilo Gamboa, Melissa Mozifian, Florian Golemo, Chris Atkeson, Dieter Fox, Ken Goldberg, John Leonard, et al. Sim2real in robotics and automation: Applications and challenges. *IEEE transactions on automation science and engineering*, 18(2):398–400, 2021.

Kai-Chieh Hsu, Allen Z Ren, Duy P Nguyen, Anirudha Majumdar, and Jaime F Fisac. Sim-to-lab-to-real: Safe reinforcement learning with shielding and generalization guarantees. *Artificial Intelligence*, 314:103811, 2023.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Advances in Artificial Life: Third European Conference on Artificial Life Granada, Spain, June 4–6, 1995 Proceedings 3*, pp. 704–720. Springer, 1995.

Yifeng Jiang, Tingnan Zhang, Daniel Ho, Yunfei Bai, C Karen Liu, Sergey Levine, and Jie Tan. Simgan: Hybrid simulator identification for domain adaptation via adversarial reinforcement learning. In *Proceedings of ICRA*, pp. 2884–2890. IEEE, 2021.

Hao Ju, Rongshun Juan, Randy Gomez, Keisuke Nakamura, and Guangliang Li. Transferring policy of deep reinforcement learning from simulation to reality for robotics. *Nature Machine Intelligence*, pp. 1–11, 2022.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of ICML*, pp. 267–274, 2002.

Sanket Kamthe and Marc Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In *Proceedings of AISTATS*, pp. 1701–1710. PMLR, 2018.

Katie Kang, Paula Gradu, Jason J Choi, Michael Janner, Claire Tomlin, and Sergey Levine. Lyapunov density models: Constraining distribution shift in learning-based control. In *Proceedings of ICML*, pp. 10708–10733. PMLR, 2022.

Haresh Karnan, Siddharth Desai, Josiah P Hanna, Garrett Warnell, and Peter Stone. Reinforced grounded action transformation for sim-to-real transfer. In *Proceedings of IROS*, pp. 4397–4402. IEEE, 2020.

Manuel Kaspar, Juan D Muñoz Osorio, and Jürgen Bock. Sim2real transfer for reinforcement learning without dynamics randomization. In *Proceedings of IROS*, pp. 4383–4388. IEEE, 2020.

Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Proceedings of IROS*, volume 3, pp. 2149–2154. IEEE, 2004.

Sylvain Koos, Jean-Baptiste Mouret, and Stéphane Doncieux. The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Transactions on Evolutionary Computation*, 17(1): 122–145, 2012.

Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: rapid motor adaptation for legged robots. In *Robotics: Science and Systems*, 2021.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Proceedings of NeurIPS*, 33:1179–1191, 2020.

Gilwoo Lee, Siddhartha S Srinivasa, and Matthew T Mason. Gp-ilqg: Data-driven robust optimal control for uncertain nonlinear dynamical systems. *arXiv preprint arXiv:1705.05344*, 2017.

Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Proceedings of CoRL*, pp. 1162–1176. PMLR, 2020.

Janosch Moos, Kay Hansel, Hany Abdulsamad, Svenja Stark, Debora Clever, and Jan Peters. Robust reinforcement learning: A review of foundations and recent advances. *Machine Learning and Knowledge Extraction*, 4(1):276–315, 2022.

Igor Mordatch, Kendall Lowrey, and Emanuel Todorov. Ensemble-cio: Full-body dynamic motion planning that transfers to physical humanoids. In *Proceedings of IROS*, pp. 5307–5314. IEEE, 2015.

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

Ted Moskovitz, Michael Arbel, Ferenc Huszar, and Arthur Gretton. Efficient wasserstein natural gradients for reinforcement learning. *arXiv preprint arXiv:2010.05380*, 2020.

Jean-Baptiste Mouret. Novelty-based multiobjectivization. In *New Horizons in Evolutionary Robotics: Extended Contributions from the 2009 EvoDeRob Workshop*, pp. 139–154. Springer, 2011.

Fabio Muratore, Michael Gienger, and Jan Peters. Assessing transferability from simulation to reality for reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, 43(4): 1172–1183, 2019.

Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. *Proceedings of NeurIPS*, 32, 2019.

Haoyi Niu, Shubham Sharma, Yiwen Qiu, Ming Li, Guyue Zhou, Jianming Hu, and Xianyuan Zhan. When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning. *arXiv preprint arXiv:2206.13464*, 2022a.

Haoyi Niu, Shubham Sharma, Yiwen Qiu, Ming Li, Guyue Zhou, Jianming HU, and Xianyuan Zhan. When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning. https://github.com/t6-thu/H2O, 2022b.

Aldo Pacchiano, Jack Parker-Holder, Yunhao Tang, Krzysztof Choromanski, Anna Choromanska, and Michael Jordan. Learning to score behaviors for guided policy optimization. In *Proceedings of ICML*, pp. 7445–7454. PMLR, 2020.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *Proceedings of ICRA*, pp. 3803–3810. IEEE, 2018.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of ICML*, pp. 2817–2826. PMLR, 2017.

Matteo Pirotta, Marcello Restelli, Alessio Pecorino, and Daniele Calandriello. Safe policy iteration. In *Proceedings of ICML*, pp. 307–315. PMLR, 2013.

Antonin Raffin. Rl baselines3 zoo. https://github.com/DLR-RM/rl-baselines3-zoo, 2020.

Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning. *IEEE Access*, 9:153171–153187, 2021.

Matteo Saveriano, Yuchao Yin, Pietro Falco, and Dongheui Lee. Data-efficient control policy search using residual dynamics learning. In *Proceedings of IROS*, pp. 4709–4715. IEEE, 2017.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of ICML*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. *Proceedings of NeurIPS*, 34:12686–12699, 2021.

Lior Shani, Yonathan Efroni, and Shie Mannor. Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. In *Proceedings of AAAI*, volume 34, pp. 5668–5675, 2020.

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *Robotics: Science and Systems XIV*, 2018.

Takumi Tanabe, Rei Sato, Kazuto Fukuchi, Jun Sakuma, and Youhei Akimoto. Max-min off-policy actor-critic method focusing on worst-case robustness to model misspecification. In *Proceedings of NeurIPS*, 2022.

Antonio Terpin, Nicolas Lanzetti, Batuhan Yardim, Florian Dorfler, and Giorgia Ramponi. Trust region policy optimization with optimal transport discrepancies: Duality and algorithm for continuous actions. *Proceedings of NeurIPS*, 35:19786–19797, 2022.

Chen Tessler, Yonathan Efroni, and Shie Mannor. Action robust reinforcement learning and applications in continuous control. In *Proceedings of ICML*, pp. 6215–6224. PMLR, 2019.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proceedings of IROS*, pp. 23–30. IEEE, 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Proceedings of IEEE-RSJ*, pp. 5026–5033. IEEE, 2012.

Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *ICML Workshop on Imitation, Intent, and Interaction*, 2018.

Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. In *Proceedings of IJCAI*, pp. 6325–6331. ijcai.org, 2019.

Ahmed Touati, Amy Zhang, Joelle Pineau, and Pascal Vincent. Stable policy optimization via off-policy divergence regularization. In *Proceedings of UAI*, pp. 1328–1337. PMLR, 2020.

Matthew Witman, Dogan Gidon, David B Graves, Berend Smit, and Ali Mesbah. Sim-to-real transfer reinforcement learning for control of thermal effects of an atmospheric pressure plasma jet. *Plasma Sources Science and Technology*, 28(9):095019, 2019.

Huang Xiao, Michael Herman, Joerg Wagner, Sebastian Ziesche, Jalal Etesami, and Thai Hong Linh. Wasserstein adversarial imitation learning. *arXiv preprint arXiv:1906.08113*, 2019.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Proceedings of NeurIPS*, 34: 28954–28967, 2021.

Wenhao Yu, C Karen Liu, and Greg Turk. Policy transfer with strategy optimization. In *Proceedings of ICLR*, 2018.

Wenhao Yu, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. Learning fast adaptation with meta strategy optimization. *IEEE Robotics and Automation Letters*, 5(2):2950–2957, 2020.

Ya-xiang Yuan. A review of trust region algorithms for optimization. In *Proceedings of ICIAM*, volume 99, pp. 271–282, 2000.

Shaojun Zhu, Andrew Kimmel, Kostas E. Bekris, and Abdeslam Boularias. Fast model identification via physics engines for data-efficient policy search. In *Proceedings of IJCAI*, pp. 3249–3256. ijcai.org, 2018.

A PROOFS

In this section, we present the proof of our Proposition 4.1 that we restate below, as well as its extensions using different discrepancy measures.

Proposition A.1 *Let $\nu_P^\pi(s,a,s')$ the state-action-state visitation distribution, where $\nu_P^\pi(s,a,s') = (1-\gamma)\mathbb{E}_{\rho_0,\pi,P}\left[\sum_{t=0}^\infty \gamma^t \mathbb{P}\left(s_t=s, a_t=a, s_{t+1}=s'\right)\right]$. For any policy π and any transition probabilities P_t and P_s, the following holds:*

$$J_{P_t}^\pi \geq J_{P_s}^\pi - \frac{2R_{max}}{1-\gamma} D_{TV}\big(\nu_{P_s}^\pi, \nu_{P_t}^\pi\big), \tag{6}$$

with D_{TV} the Total Variation distance.

Proof A.1 *It is known that $J_P^\pi = \frac{1}{1-\gamma}\mathbb{E}_{\nu_P^\pi}\left[r(s,a,s')\right]$. Now:*

$$\begin{aligned}
\left|J_{P_t}^\pi - J_{P_s}^\pi\right| &= \frac{1}{1-\gamma}\left|\Big(\mathbb{E}_{\nu_{P_t}^\pi}\left[r(s,a,s')\right] - \mathbb{E}_{\nu_{P_s}^\pi}\left[r(s,a,s')\right]\Big)\right| && (7)\\
&= \frac{1}{1-\gamma}\left|\int_{s,a,s'}\Big(r(s,a,s')\nu_{P_t}^\pi(s,a,s') - r(s,a,s')\nu_{P_s}^\pi(s,a,s')\Big)\,\mathrm{d}\{sas'\}\right| && (8)\\
&= \frac{1}{1-\gamma}\left|\int_{s,a,s'} r(s,a,s')\left(\nu_{P_t}^\pi(s,a,s') - \nu_{P_s}^\pi(s,a,s')\right)\mathrm{d}\{sas'\}\right| && (9)\\
&\leq \frac{2R_{max}}{1-\gamma} D_{TV}\big(\nu_{P_s}^\pi, \nu_{P_t}^\pi\big). && (10)
\end{aligned}$$

The last inequality is an application of Holder's inequality, by setting p to ∞ and q to 1.

An application of Pinsker inequality (Csiszar & Körner, 1981) provides a similar upper bound with the ullback Leibleir divergence.

Proposition A.2 *Let $\nu_P^\pi(s,a,s')$ the state-action-state visitation distribution, where $\nu_P^\pi(s,a,s') = (1-\gamma)\mathbb{E}_{\rho_0,\pi,P}\left[\sum_{t=0}^\infty \gamma^t \mathbb{P}\left(s_t=s, a_t=a, s_{t+1}=s'\right)\right]$. For any policy π and any transition probabilities P_t and P_s such that $\nu_{P_s}^\pi$ is absolutely continuous with respect to $\nu_{P_t}^\pi$, the following holds:*

$$J_{P_t}^\pi \geq J_{P_s}^\pi - \frac{\sqrt{2}R_{max}}{1-\gamma}\sqrt{D_{KL}\big(\nu_{P_s}^\pi \,\|\, \nu_{P_t}^\pi\big)}, \tag{11}$$

with D_{KL} the Kullback Leibleir divergence.

A lower bound with the Jensen Shannon divergence can also be found thanks to Corander et al. (2021, Proposition 3.2).

Proposition A.3 *We assume the state-action space. Let $\nu_P^\pi(s,a,s')$ the state-action-state visitation distribution, where $\nu_P^\pi(s,a,s') = (1-\gamma)\mathbb{E}_{\rho_0,\pi,P}\left[\sum_{t=0}^\infty \gamma^t \mathbb{P}\left(s_t=s, a_t=a, s_{t+1}=s'\right)\right]$. We assume the support of $\nu_{P_s}^\pi$ and $\nu_{P_s}^\pi$ is $\mathcal{S}\times\mathcal{A}\times\mathcal{S}$. Then, for any policy π and any transition probabilities P_t and P_s, the following holds:*

$$J_{P_t}^\pi \geq J_{P_s}^\pi - \frac{4R_{max}}{(1-\gamma)}\sqrt{D_{JS}\big(\nu_{P_s}^\pi \,\|\, \nu_{P_t}^\pi\big)}, \tag{12}$$

with D_{JS} the Jensen Shannon divergence.

B ALGORITHMS DETAILS

In this section, we further present the different algorithms used in this paper.

B.1 Domain Adaptation with Rewards from Classifiers (DARC)

We introduce our main baseline Domain Adaptation with Rewards from Classifiers (DARC), which is the prominent state-of-the-art algorithm that tackles the Sim-to-Real task by modifying the RL objective.

DARC takes a variational perspective to the off-dynamics RL problem. Given a trajectory $\tau = (s_0, a_0, s_1, a_1, \dots)$, the target distribution $p(\tau)$ over trajectories is defined as the trajectories maximizing the exponentiated rewards in the real environment:

$$p(\tau) = \rho(s_0) \left(\prod_t P_{\mathrm{t}}(s_{t+1}|s_t, a_t) \right) \exp \left(\sum_t r(s_t, a_t) \right). \tag{13}$$

Let the agent's distributions over trajectories in the simulator $q^{\pi_\theta}(\tau)$ be:

$$q^{\pi_\theta}(\tau) = \rho(s_0) \left(\prod_t P_{\mathrm{s}}(s_{t+1}|s_t, a_t) \right) \pi_\theta(a_t|s_t). \tag{14}$$

DARC minimizes the reversed KL-divergence between $q^{\pi_\theta}(\tau)$ and $p(\tau)$, which results in the following objective expression:

$$-D_{\mathrm{KL}}(q^{\pi_\theta}(\tau) \parallel p(\tau)) = \mathbb{E}_{\tau \sim q^{\pi_\theta}(\cdot)} \left[\sum_{t=1}^{T} r(s_t, a_t) + \mathcal{H}\left(\pi_\theta(\cdot|s_t)\right) + \Delta r(s_t, a_t, s_{t+1}) \right], \tag{15}$$

with $\Delta r(s_t, a_t, s_{t+1}) = \log P_{\mathrm{t}}(s_{t+1}|s_t, a_t) - \log P_{\mathrm{s}}(s_{t+1}|s_t, a_t)$ and $\mathcal{H}(\cdot)$ the entropy.

The additional reward term incentivizes the agent to select transitions from the simulator that are similar to the real environment. We hypothesize that this is why DARC may struggle when the discrepancies between the simulated and real environment are global. Since the transition probabilities are unknown, DARC uses a pair of binary classifiers to infer whether transitions come from the simulated or real environment. These classifiers are then used to create a proxy equivalent to Δr.

B.2 Generative Adversarial Imitation Learning (GAIL)

Generative Adversarial Imitation Learning (GAIL) (Ho & Ermon, 2016) is a state-of-the-art Imitation Learning algorithm. Its goal is to recover an expert policy π_e by minimizing the Jensen-Shanon divergence between the state-action visitation distributions of the expert and the learning policy. Both the expert and the learning policy interact with the same MDP with transition probabilities P. For ease of notation, let $\mathbb{E}_\pi\left[c(s,a)\right] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t c(s,a) | s_0 \sim \rho, a_t \sim (\cdot|s_t), s_{t+1} \sim P(\cdot|s_t, a_t)\right]$ for any cost function $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

The authors define the general objective to solve by introducing a convex cost function regularizer $\psi : \mathbb{R}^{S \times A} \to \mathbb{R}$ and its convex conjugate ψ^*:

$$\underset{\theta \,\in\, \Theta}{\text{minimize}} \quad \psi^*(\mu_P^{\pi_\theta} - \mu_P^{\pi_e}) - \mathcal{H}\left(\pi_\theta\right). \tag{16}$$

Following Equation 13 of Ho & Ermon (2016) which defines ψ_{GAIL}, the authors establish the following equivalence:

$$\psi^*_{\text{GAIL}}(\mu_P^{\pi_\theta} - \mu_P^{\pi_e}) = \sup_{D \in (0,1)^{S \times A}} \mathbb{E}_{\pi_\theta}\left[\log\left(D(s,a)\right)\right] + \mathbb{E}_{\pi_e}\left[\log\left(1 - D(s,a)\right)\right] \tag{17}$$

where $D : \mathcal{S} \times \mathcal{A} \to (0,1)$ is a classifier. Finally, it is demonstrated this specific convex cost function induces the following objective:

$$\underset{\theta \in \Theta}{\text{minimize}} \quad \psi^*_{\text{GAIL}}(\mu_P^{\pi_\theta} - \mu_P^{\pi_e}) - \mathcal{H}(\pi_\theta) = D_{\text{JS}}(\mu_P^{\pi_\theta} \parallel \mu_P^{\pi_e}) - \mathcal{H}(\pi_\theta). \tag{18}$$

In practice, the classifier D is trained to distinguish between samples $(s, a) \in (\mathcal{S} \times \mathcal{A})$ from $\mu_P^{\pi_\theta}$ and $\mu_P^{\pi_e}$. The reward used for optimizing the RL agent is given by $r_{\text{imit}} = -\log(D(s, a))$. If the classifiers learn to distinguish between state-action-state samples, then the algorithm minimizes $D_{\text{JS}}(\nu_P^{\pi_\theta} \parallel \nu_P^{\pi_e}) - \mathcal{H}(\pi_\theta)$.

B.3 Conservative Q-Learning (CQL)

In the offline setting, agents aim to learn a good policy from a fixed data set of M transitions $\mathcal{D} = \{(s_i, a_i, s_{i+1}\}_{i=0}^{M}$ that was collected with an unknown behavioral policy π_β. Offline RL algorithms have demonstrated impressive results when the data set is gathered with a sufficiently good policy and possesses enough transitions, often outperforming the behavioral policy.

Conservative Q-Learning (CQL) (Kumar et al., 2020) is a state-of-the-art offline RL algorithm. It modifies the learning procedure of the Q-functions to favor transitions appearing in the data set. At iteration k, the Q-values are updated as follows:

$$\underset{\omega \in \Omega}{\text{minimize}} \quad \beta\, \mathbb{E}_{s \sim \mathcal{D}} \left[\left(\log \sum_{a \in \mathcal{A}} \exp\left(Q_\omega^{\pi_{\theta_k}}(s, a) \right) - \mathbb{E}_{a \sim \pi_\beta(\cdot|s)} \left[Q_\omega^{\pi_{\theta_k}}(s, a) \right] \right) \right] + \mathcal{E}\left(Q_\omega^{\pi_{\theta_k}} \right), \tag{19}$$

where $\mathcal{E}(Q)$ represents the traditional loss associated with the Q-functions. The regularization, controlled by the hyper-parameter β, penalizes the Q-values associated with state-action pairs not appearing in the data set.

C Experimental Details

In this section, in addition to the values of the hyperparameters necessary to replicate our experiments, we provide further details of the experimental protocol and training. In this section, considering the possible high variance of RL_{Sim}, the standard deviation is multiplied by a factor of 0.3. The original variance can be found in Table 2.

C.1 Environment Details

In all the considered environments, one property is modified in the real environment.

Gravity Pendulum Gravity is increased to 14 instead of 10. Since the pendulum requires more time to reach the objective, we also increase the length of each episode to 500 time-steps in the real environment, while keeping the original length of 200 time-steps in the simulator.

Broken Joint or Leg environments In these environments, the considered robot - either HalfCheetah or Ant - is crippled in the target domain, where the effect of one or two joints is removed. In practice, this means that it sets one or two dimensions of the action to 0. These environments were extracted from the open source code of (Eysenbach et al., 2020).

Heavy Cheetah The total mass of the HalfCheetah MuJoCo robot is increased from 14 to 20.

Friction Cheetah The friction coefficient of the HalfCheetah MuJoCo robot's feet is increased from 0.4 to 1.

Low Fidelity Minitaur The original Minitaur environment uses a linear torque-current linear relation for the actuator model. It has been improved in (Tan et al., 2018) by introducing non-linearities into this relation where they managed to close the Sim-to-Real gap for a real Minitaur environment.

In practice, the Minitaur environment can be found in the PyBullet library (Coumans & Bai, 2016–2021). The high fidelity is registered as MinitaurBulletEnv-v0. The low fidelity environment can be recovered by calling MinitaurBulletEnv-v0 and by setting the argument `accurate motor model enabled` to False and `pd control enabled` to True.

C.2 Learning curves and normalized return

We report here the learning curves of the different agents mentioned in this paper. For clarity purposes, we keep all baselines fixed except for our agent and DARC, our main competitor. Here, FOOD uses the regularization with d_P^π for Gravity Pendulum and ν_P^π for the other environments as GAIL proved to be more stable when FOOD used PPO.

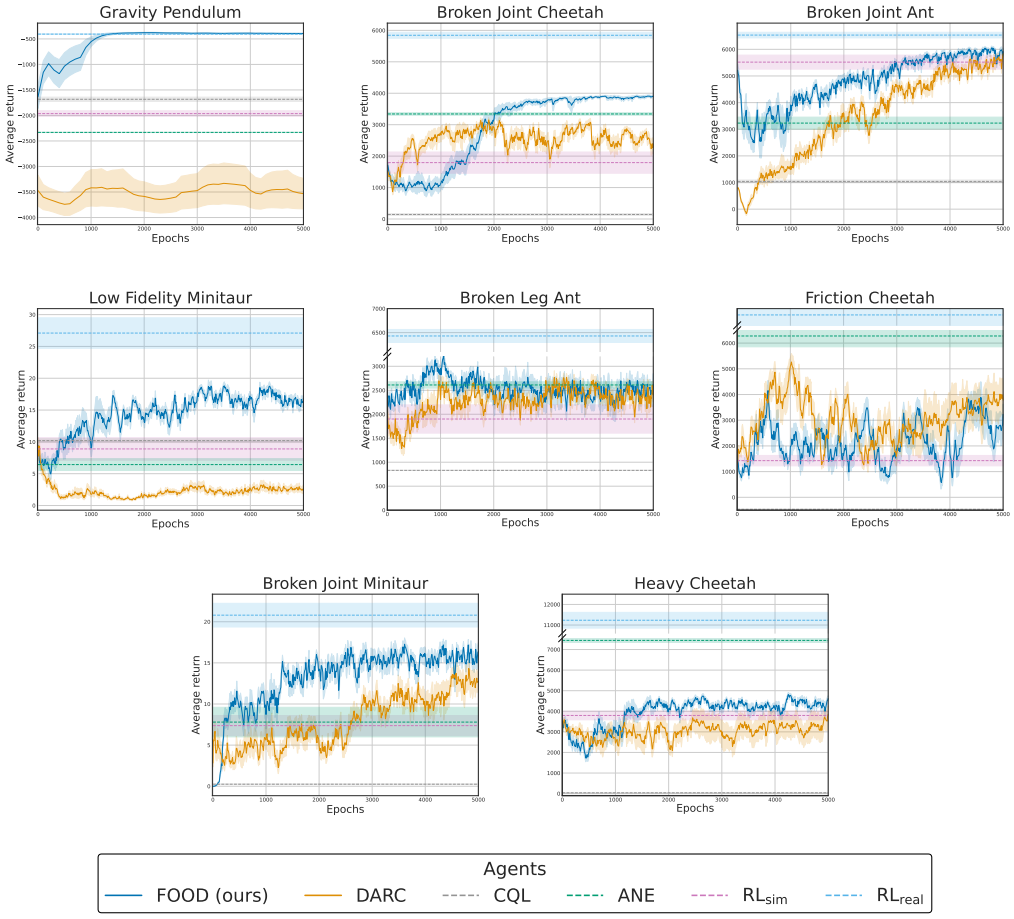

Figure 2: Learning curves of FOOD and DARC for all the proposed environments.

Finally, we report the normalized return of all environments in Table 3.

C.3 Global hyper-parameters

Our experiments are based on the A2C and PPO implementations proposed by the open-source code (Kostrikov, 2018). We also found that it may be profitable to add a TanH function at the end of the network's policy for the PPO agent. We have selected their hyper-parameters according to the source (Raffin, 2020) and included them in the table below.

The Minitaur environments As proposed by the PyBullet library (Coumans & Bai, 2016–2021), γ is set to 0.995 for the Minitaur environments. Besides, unlike the Gym and Mujoco environments,

Environment	CQL	ANE	DARC	FOOD (Ours)		
				d_P^π	μ_P^π	ν_P^π
Gravity Pendulum	0.18 ± 0.09	-0.22 ± 0.01	-0.99 ± 0.55	$\mathbf{0.95 \pm 0.03}^*$	-0.17 ± 0.03	-0.23 ± 0.01
Broken Joint Cheetah	-0.41 ± 0.03	0.38 ± 0.03	0.19 ± 0.1	$\mathbf{0.52 \pm 0.05}$	$\mathbf{0.5 \pm 0.04}$	$\mathbf{0.53 \pm 0.02}^*$
Broken Joint Minitaur	-0.53 ± 0.01	0.03 ± 0.45	$\mathbf{0.41 \pm 0.15}$	$\mathbf{0.46 \pm 0.29}$	$\mathbf{0.56 \pm 0.22}$	$\mathbf{0.68 \pm 0.34}^*$
Heavy Cheetah	-0.51 ± 0.01	$\mathbf{0.49 \pm 0.04}^*$	0.02 ± 0.18	0.14 ± 0.07	0.15 ± 0.02	0.13 ± 0.04
Broken Joint Ant	-4.41 ± 0.17	-2.25 ± 0.74	-0.47 ± 0.36	0.03 ± 0.2	$\mathbf{0.6 \pm 0.1}$	$\mathbf{0.65 \pm 0.1}^*$
Friction Cheetah	-0.24 ± 0.01	$\mathbf{0.6 \pm 0.18}^*$	0.2 ± 0.1	0.22 ± 0.08	0.3 ± 0.19	0.3 ± 0.09
Low Fidelity Minitaur	0.07 ± 0.05	-0.13 ± 0.17	-0.3 ± 0.1	$\mathbf{0.38 \pm 0.15}$	$\mathbf{0.45 \pm 0.1}$	$\mathbf{0.5 \pm 0.05}^*$
Broken Leg Ant	-0.24 ± 0.01	$\mathbf{0.16 \pm 0.05}$	0.01 ± 0.12	$\mathbf{0.1 \pm 0.18}$	$\mathbf{0.17 \pm 0.08}$	$\mathbf{0.18 \pm 0.15}$

Table 3: Normalized average return of the RL agents on different Sim-to-Real settings using 5 trajectories from the real environment.

Table 4: Chosen hyper-parameters for both A2C and PPO. The PPO hyper-parameters were fixed for the other environments.

Hyperparameters	**A2C**	**PPO**
num-processes	8	8
num-steps	200	1000
lr	$2.5 * 10^{-4}$	$3.0 * 10^{-4}$
γ	0.99	0.99
use-gae	True	True
gae-lambda	0.9	0.95
entropy-coef	0.01	0.001
value-loss-coef	0.4	0.5
use-linear-lr-decay	True	True
ppo-epoch	N/A	5
num-mini-batch	N/A	32
clip-param	N/A	0.1
TanH Squash	False	True

they do not use a Tanh squashing function in their policy and the `num-processes` hyper-parameter is set to 1.

Discriminators training Both FOOD and DARC incorporate classifiers in their objective. At each epoch, 1000 data points are sampled from both simulated and real transition data sets. The classifiers are then trained with batch sizes of 128 for Pendulum and 256 for the MuJoCo environments. They share the same network structure: a 2 hidden layer MLP with 64 (for Pendulum) or 256 (for MuJoCo) units and ReLU activations. We did not find that the size of the networks played an important role in the results.

C.4 FOOD Hyper-parameters Sensitivity Analysis

This subsection investigates the impact of our main hyper-parameter α, which regulates the strength of regularization that defines a threshold between maximizing the rewards of the simulator and staying close to the real trajectory. All FOOD results are summarized in Figure 3, where, similar to the previous section, FOOD uses the regularization with d_P^π in Gravity Pendulum and ν_P^π for the other environments. Note that for the Gravity Pendulum environment, $\alpha \in \{0, 1, 5, 10\}$.

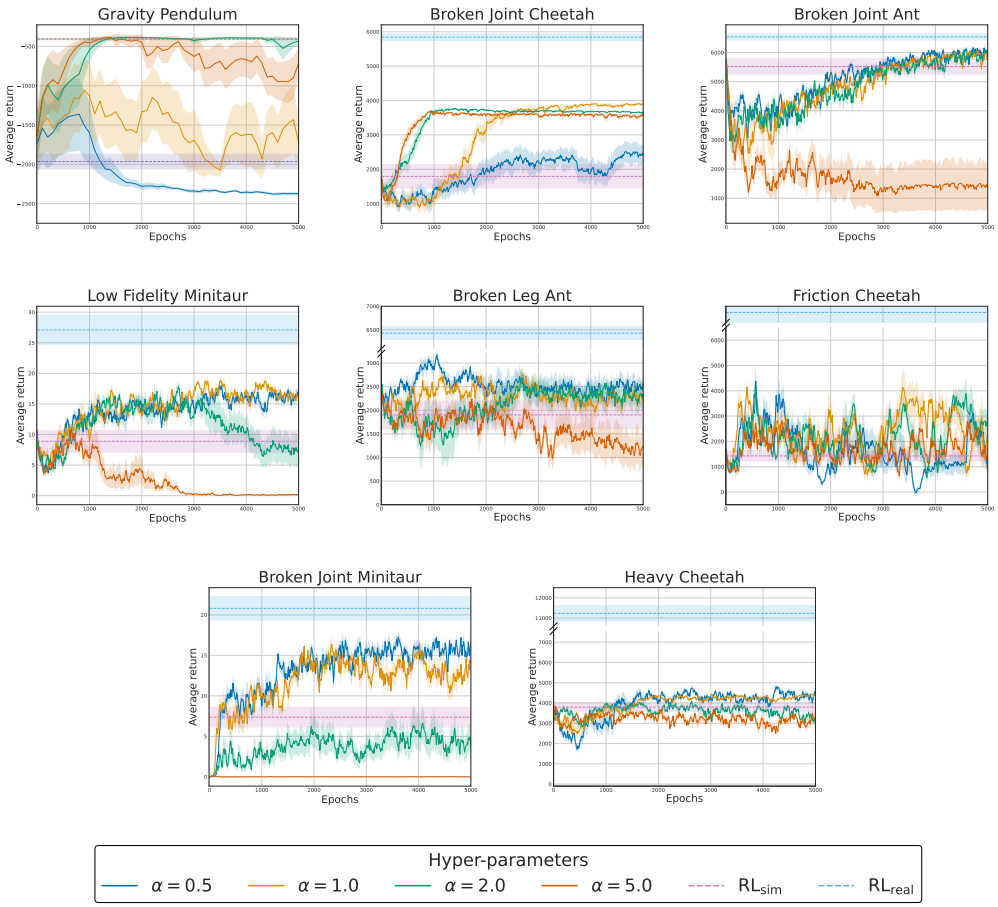

Figure 3: Complete hyperparameter sensitivity analysis for the best FOOD agent on the different Sim-to-Real environments.

In all the studied environments where PPO was used, we observe that unless for the low or high values of α ($\alpha \in \{0.5, 5\}$), the FOOD agent improves performance compared to RL_{Sim}. Both cases can be explained. If the value is too high, it may disrupt the gradients and prevent convergence to a good solution. As mentioned in the main paper, this phenomenon also affects the performance in the simulator, so it would be easy for practitioners to remove such bad hyper-parameters. It may also happen that the strength of the regularization is too low. In that case, FOOD has approximately the same performance as RL_{Sim}, as illustrated in Broken Joint HalfCheetah.

Hence, we recommend setting the regularization to have approximately the same weight as the average return. For this, since its advantages are normalized, we recommend using PPO and setting the α parameter to 1.

C.5 Comparison between the different IL algorithms for the FOOD agent

FOOD is a general algorithm that may use any chosen Imitation Learning algorithm. Each algorithm minimizes a certain type of divergence between state or state-action visitation distributions, as summarized in Table 1. Here, we investigate which IL is better suited for the considered environments.

We compare GAIL (Ho & Ermon, 2016), GAIL-S, GAIL-SAS, AIRL (Fu et al., 2017), PWIL (Dadashi et al., 2020) and PWIL-S in Table 5. GAIL and its extensions were extracted directly from (Kostrikov, 2018), AIRL from (Gangwani, 2021), and PWIL and its extensions were recoded from scratch.

Environment	GAIL-d	GAIL-μ	GAIL-ν	AIRL-μ	PWIL-d	PWIL-μ	PWIL-ν
Gravity Pendulum	$\mathbf{-485 \pm 54^*}$	-2224 ± 43	-2327 ± 14	-1926 ± 572	$\mathbf{-980 \pm 838}$	$\mathbf{-948 \pm 789}$	$\mathbf{-978 \pm 816}$
Broken Joint Cheetah	$\mathbf{3888 \pm 201}$	$\mathbf{3801 \pm 155}$	$\mathbf{3950 \pm 97^*}$	3617 ± 225	3537 ± 248	2999 ± 752	$\mathbf{3797 \pm 389}$
Heavy Cheetah	$\mathbf{4828 \pm 553}$	$\mathbf{4876 \pm 181^*}$	$\mathbf{4743 \pm 297}$	$\mathbf{4604 \pm 184}$	2945 ± 856	2771 ± 1235	3494 ± 318
Broken Joint Ant	5547 ± 204	$\mathbf{6145 \pm 98}$	$\mathbf{6179 \pm 86^*}$	5014 ± 401	3725 ± 988	3483 ± 747	3182 ± 1337
Friction Cheetah	$\mathbf{3212 \pm 2279}$	$\mathbf{3890 \pm 1495}$	$\mathbf{3852 \pm 733}$	$\mathbf{2957 \pm 1526}$	$\mathbf{3451 \pm 361}$	$\mathbf{3926 \pm 735}$	$\mathbf{4227 \pm 740^*}$
Broken Joint Minitaur	$\mathbf{13.6 \pm 3.8}$	$\mathbf{14.9 \pm 3}$	$\mathbf{16.6 \pm 4.7^*}$	$\mathbf{15.8 \pm 2.3}$	$\mathbf{14.6 \pm 1.9}$	$\mathbf{12.1 \pm 5.2}$	$\mathbf{10.5 \pm 6.1}$
Low Fidelity Minitaur	$\mathbf{15.7 \pm 2.8}$	$\mathbf{17 \pm 2}$	$\mathbf{17.9 \pm 0.8^*}$	7.5 ± 5.7	$\mathbf{13.6 \pm 5.1}$	11.4 ± 3.5	12.1 ± 5.5
Broken Leg Ant	$\mathbf{2345 \pm 806}$	$\mathbf{2652 \pm 356}$	$\mathbf{2733 \pm 719^*}$	1634 ± 857	1490 ± 714	1554 ± 886	1697 ± 393

Table 5: FOOD sensitivity analysis with respect to the Imitation Learning agent used. We report the average return over 4 seeds associated with their best hyper-parameter α.

Overall, we observe that all GAIL-associated algorithms have the best results. We attribute this success to the implementation we used, which was optimized for the PPO agent. In addition, FOOD with PWIL has poor results in some environments. This can be attributed to two factors. First, we cannot rule out an error in our code, as we coded it from scratch. Second, this algorithm was introduced in the D4PG agent (Barth-Maron et al., 2018): it is possible that PPO does not leverage well the PWIL's rewards.

An interesting discussion is about GAIL-S, GAIL and GAIL-SAS. The latter focuses on state visitation distributions which should give the FOOD agent more freedom to find a better action. This is for example what is observed in the Gravity Pendulum environment. However, in most cases, we encourage practitioners to use GAIL-SAS, as it gives stable results. This is consistent with both Propositions 4.1 and 4.1: an additional constant appears when only the state visitation distributions are considered.

C.6 DARC Hyperparameters Sensitivity Analysis

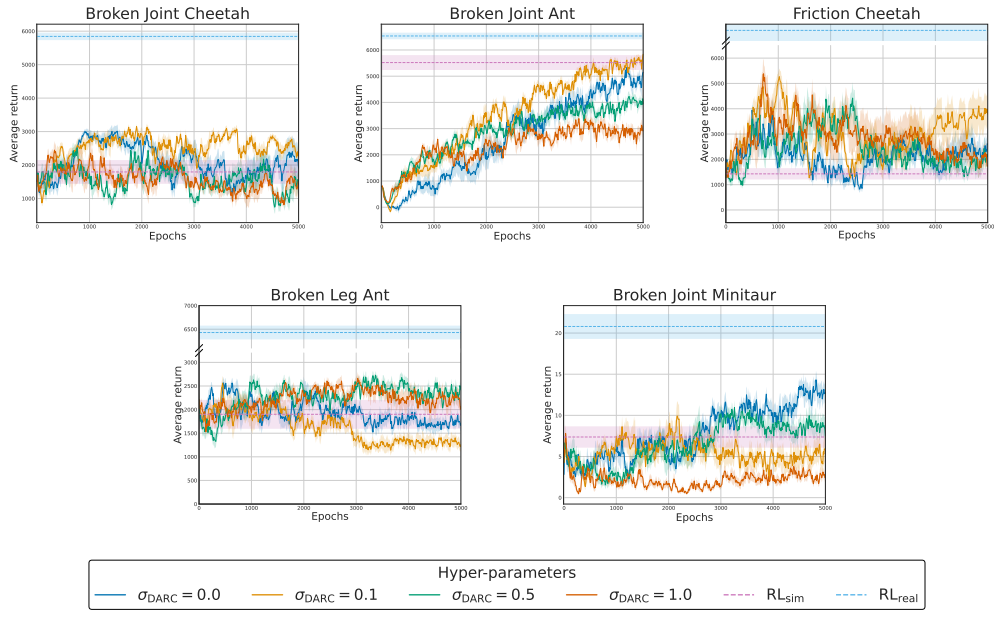

Figure 4: Hyper-parameter sensitivity analysis for the DARC agent on the different Sim-to-Real environments where DARC worked well.

We observe a clear dependence on the noise added to the discriminator, although there seems to be no pattern for choosing the right hyper-parameter. For instance, the best hyper-parameter for Broken

Joint Cheetah and Broken Joint Ant is $\sigma_{\text{DARC}} = 0.1$, but this value leads to worse performance than RL_{Sim} on the two other presented environments.

C.7 ANE Hyperparameters Sensitivity Analysis

We also detail the ANE's results for all environments. As a reminder, ANE adds a centered Gaussian noise with std $\sigma_{\text{ANE}} \in \{0.1, 0.2, 0.3, 0.5\}$ to the action during training.

Figure 5: Hyper-parameter sensitivity analysis for the ANE agent on the different Sim-to-Real environments.

These figures are not easily interpretable. This technique may work very well as observed for Heavy Cheetah, but may fail for other environments such as Broken Joint Ant or Low Fidelity Minitaur.