Relative Policy-Transition Optimization for Fast Policy Transfer

Jiawei Xu[1,2*], **Cheng Zhou**[1], **Yizheng Zhang**[1], **Baoxiang Wang**[2], **Lei Han**[1*]

[1]Tencent Robotics X
[2]The Chinese University of Hong Kong, Shenzhen
jiaweixu1@link.cuhk.edu.cn, {mikechzhou,yizhenzhang}@tencent.com
bxiangwang@cuhk.edu.cn, lxhan@tencent.com

Abstract

We consider the problem of policy transfer between two Markov Decision Processes (MDPs). We introduce a lemma based on existing theoretical results in reinforcement learning to measure the relativity gap between two arbitrary MDPs, that is the difference between any two cumulative expected returns defined on different policies and environment dynamics. Based on this lemma, we propose two new algorithms referred to as Relative Policy Optimization (RPO) and Relative Transition Optimization (RTO), which offer fast policy transfer and dynamics modelling, respectively. RPO transfers the policy evaluated in one environment to maximize the return in another, while RTO updates the parameterized dynamics model to reduce the gap between the dynamics of the two environments. Integrating the two algorithms results in the complete Relative Policy-Transition Optimization (RPTO) algorithm, in which the policy interacts with the two environments simultaneously, such that data collections from two environments, policy and transition updates are completed in one closed loop to form a principled learning framework for policy transfer. We demonstrate the effectiveness of RPTO on a set of MuJoCo continuous control tasks by creating policy transfer problems via variant dynamics.

Introduction

Deep reinforcement learning (RL) has demonstrated great successes in recent years, in solving a number of challenging problems like Atari (Mnih et al. 2015), GO (Silver et al. 2016, 2017), StarCraft II (Vinyals et al. 2019; Han et al. 2020), ViZDoom (Li et al. 2023) and legged robotics (Han et al. 2023). These successes demonstrate that current deep RL methods are capable to explore and exploit sufficiently in huge observation and action spaces, as long as sufficient and effective data samples can be generated for training, such as the cases in games. For example, AlphaGo Zero (Silver et al. 2017) costs 3 days of training over 4.9 million self-play games, and OpenAI Five (Berner et al. 2019) and AlphaStar (Vinyals et al. 2019) spend months of training using thousands of GPUs/TPUs over billions of generated matches. However, for environments that prohibit infinite interactions, e.g., robotics, real life traffic control and autopilot, etc., applying general RL is difficult because generating data is extremely expensive and slow. Even if parallel data collection is possible, for example, by deploying multiple robots or vehicles running simultaneously, the scale of collected data is still far below that in virtual games. Worse still, exploration in these environments is considerably limited for safety reasons, which further reduces the effectiveness of the generated data. Due to the above challenges, similar significant advances like in solving virtual games have not been witnessed in these applications yet.

Generally, there are three tracks of approaches targeting to alleviate the aforementioned situation to promote the widespread application of RL. They are improving the data efficiency, transfer learning and simulator engineering.

To improve data efficiency, many recent efforts have been paid on investigating offline RL algorithms (Rosete-Beas et al. 2023; Zhan et al. 2022; Lyu et al. 2022; Siegel et al. 2020; Fujimoto, Meger, and Precup 2019). Compared to standard on-policy or off-policy RL, offline RL aims to effectively use previously collected experiences stored in a given dataset, like supervised learning, without online interactions with the environment. The stored experiences may not be generated by a fixed or known policy, so offline RL algorithms can leverage any previously collected data and learn a provably better policy than those who generated the experiences in the dataset. Although offline RL can effectively take advantage of finite data samples, solving a complex real-world task still requires a huge amount of high-quality offline experiences. Another way to increase data efficiency is to adopt model-based RL. Compared to model-free methods, model-based RL (Bakker 2001; Schaefer, Udluft, and Zimmermann 2007; Kaiser et al. 2019; Janner et al. 2019; Delecki, Corso, and Kochenderfer 2023) learns a dynamics model that mimics the transitions in the true environment, and then the policy is free to interact with the learned dynamics. It has been proved that the true return can be improved by interacting with the learned dynamics model when the model error is bounded (Janner et al. 2019). However, learning an accurate dynamics model still requires sufficient transition by interacting with the true environment.

Transfer learning in RL (Taylor and Stone 2009; Zhu et al. 2023) is practically useful to adapt a policy learned in a *source* environment to solve another task in the *target* environment. In the context of this paper, we consider the case *where the policy is free to explore in the source environment,*

*These authors contributed equally.

while the amount of collected data in the target environment should be as small as possible. When the source environment is a simulated one while the target environment takes place in reality, the transfer problem is also known as the simulation to reality (sim2real) problem. The simplest transfer method is to train the policy in the source environment and then use the converged parameters as the warm start for a (or part of a) new policy in the target environment, so that the amount of interactions with the target is expected to be largely reduced, as long as the tasks and dynamics in the two environments are closely related.

An important concept in transfer learning is that instead of directly solving the target problem, a source task is considered in advance. Sharing this spirit, the last track of approaches tries to build a proxy simulator that is as close as possible to the target environment, and hence we refer to such methods as simulator engineering. For example, in robotics control problems, there are many mature toolboxes that can offer simulation engineering, including MuJoCo, PyBullet, Gazebo, etc. Model-based RL can also be viewed as a specific form of simulator engineering that the simulator is composed of a pure neural network, which is trained to approach the target environment as with lower a model error as possible, while this might require a large amount of dynamics data in the target environment as mentioned above. Actually, to achieve more efficient and accurate simulator engineering, one recent rising direction is to integrate differentiable programming and physical systems to build a trainable simulator, which follows the physical laws as in reality and also whose key factors, such as the mass, length or friction of some objects, are trainable like the parameters in neural networks. Representative examples include the DiffTaichi (Hu et al. 2020), Brax (Freeman et al. 2021) and Nimble (Werling et al. 2021).

Overall, the existing methods focus on either directly improving the data efficiency in the target environment or bridging/reducing the gap between a proxy environment and the target environment, and there lacks a principled way that can incorporate the learning in the two environments through a unified framework. In this paper, we inherit the spirit of transfer learning and consider two environments, where one is cheap to interact and another is the goal to solve, and the number of interactions in the target environment should be as small as possible. We believe that there exist some explicit connections between the expected returns in the two environments, given two different policies. Actually, previous theoretical approaches have analyzed the difference between two expected returns under different policies (Kakade and Langford 2002; Schulman et al. 2015), or different dynamics (Luo et al. 2018), separately.

To provide an explicit value difference for the case where both the policy and dynamics vary in two Markov Decision Processes (MDPs), we introduce a lemma to combine the previous results in (Kakade and Langford 2002; Luo et al. 2018). Let $\mathcal{P}(s'|s,a)$ and $\mathcal{P}'(s'|s,a)$ denote two dynamics transition functions in any two arbitrary MDPs sharing the same state and action spaces, where (s,a,s') is the tuple of the state, action and next state. Let $\pi'(a|s)$ and $\pi(a|s)$ denote two arbitrary policies, and denote $J(\mathcal{P},\pi)$ as the expected return given any $\mathcal{P}$ and π. Then, the lemma gives an explicit form for the value difference $J(\mathcal{P}',\pi) - J(\mathcal{P},\pi')$, which is referred to as the *relativity gap* between the two MDPs.

Now, suppose $\mathcal{P}^{source}$ and $\mathcal{P}^{target}$ are the dynamics functions in the *source* and *target* MDPs respectively, and $J(\mathcal{P}^{source},\pi^{source})$ has been maximized by optimizing a parameterized π^{source}. Then, with fixed $\mathcal{P}^{source}$, $\mathcal{P}^{target}$ and π^{source}, maximizing the relativity gap over π^{target} by constraining π^{target} to be close to π^{source} will also improve the return $J(\mathcal{P}^{target},\pi^{target})$; on the other hand, for trainable $\mathcal{P}^{source}$, minimizing the relativity gap by optimizing $\mathcal{P}^{source}$ given fixed policies $\pi^{source} = \pi^{target}$ will reduce the dynamics gap, similar to what is pursued by conventional model-based RL methods. Based on the above two principles and the value relativity lemma, we then propose two new algorithms referred to as Relative Policy Optimization (RPO) and Relative Transition Optimization (RTO), respectively. RPO transfers the policy evaluated in the source environment to maximize the return in the target environment, while RTO updates a dynamics model to reduce the value gap in the two environments. Then, applying RPO and RTO simultaneously offers a complete algorithm named Relative Policy-Transition Optimization (RPTO), which can transfer the policy from the source to the target smoothly. RPO, RTO and RPTO interact with the two environments simultaneously, so that data collections from two environments, policy and transition updates are completed in a closed loop to form a principled learning framework. In the experimental section, we show how to practically apply RPO, RTO and RPTO algorithms. We demonstrate the effectiveness of these methods in the MuJoCo continuous control tasks, by varying the physical variables of the objects to create policy transfer problems. In the last section, we discuss a few interesting directions which are worthy of future investigations.

Preliminaries

Reinforcement Learning. A standard RL problem can be described by a tuple $\langle \mathcal{E}, \mathcal{A}, \mathcal{S}, \mathcal{P}, r, \gamma, \pi \rangle$, where $\mathcal{E}$ indicates the environment that is an MDP with dynamics transition probability $\mathcal{P}$; at each time step t, $s_t \in \mathcal{S}$ is the global state in the state space $\mathcal{S}$, and $a_t \in \mathcal{A}$ is the action executed by the agent at time step t from the action space $\mathcal{A}$; the dynamics transition function $\mathcal{P}(s_{t+1}|s_t,a_t)$ is the probability of the state transition $(s_t,a_t) \to s_{t+1}$; for the most general case, the reward $r(s_t,a_t,s_{t+1})$ can be written as a function of s_t, a_t and s_{t+1}, while in many tasks it only relies on one or two of them, or it is even a constant in sparse rewards problem (Xu et al. 2023). For notation simplicity, we usually write $r(s_t,a_t,s_{t+1})$ as r_t; $\gamma \in [0,1]$ is a discount factor and $\pi(a_t|s_t)$ denotes a stochastic policy. The following equations define some important quantities in reinforcement learning. The objective of RL is to maximize the expected discounted return

$$J(\mathcal{P},\pi) = \mathbb{E}_{s_0,a_0,\cdots\sim\mathcal{P},\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right], \quad (1)$$

where $s_0 \sim \mathcal{P}(s_0),\ a_t \sim \pi(a_t|s_t),\ s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t,a_t)$.

At time step t, the state-action value $Q^{\mathcal{P},\pi}$, value function $V^{\mathcal{P},\pi}$, and advantage $A^{\mathcal{P},\pi}$ are defined as

$$Q^{\mathcal{P},\pi}(s_t,a_t)=\mathbb{E}_{s_{t+1},a_{t+1},\cdots\sim\mathcal{P},\pi}\left[\sum_{l=0}^{\infty}\gamma^l r_{t+l}\right], \quad (2)$$

$$V^{\mathcal{P},\pi}(s_t)=\mathbb{E}_{a_t,s_{t+1},\cdots\sim\mathcal{P},\pi}\left[\sum_{l=0}^{\infty}\gamma^l r_{t+l}\right], \quad (3)$$

and $A^{\mathcal{P},\pi}(s,a)=Q^{\mathcal{P},\pi}(s,a)-V^{\mathcal{P},\pi}(s)$. In the above standard definitions, we explicitly show their dependence on both the dynamics $\mathcal{P}$ and policy π, since we will analyze these functions defined on variant dynamics and policies. This convention will be kept throughout the paper.

The Policy Improvement Theorem. Given two arbitrary policies π and π', the policy improvement theorem (Kakade and Langford 2002; Schulman et al. 2015) is the fact revealed by the following equation

$$J(\mathcal{P},\pi)=J(\mathcal{P},\pi')+\mathbb{E}_{s_0,a_0,\cdots\sim\mathcal{P},\pi}\left[\textstyle\sum_{t=0}^{\infty}\gamma^t A^{\mathcal{P},\pi'}(s_t,a_t)\right]. \quad (4)$$

Based on this theorem, some widely adopted RL algorithms such as TRPO (Schulman et al. 2015) and PPO (Schulman et al. 2017) are developed.

The Telescoping Lemma. The telescoping lemma (Luo et al. 2018) reveals the value difference between any two dynamics functions $\mathcal{P}'$ and $\mathcal{P}$, given some fixed policy π. Using our notations, that is

$$V^{\mathcal{P}',\pi}-V^{\mathcal{P},\pi}=\tfrac{1}{1-\gamma}\big[\mathbb{E}_{s'\sim\mathcal{P}'}V^{\mathcal{P},\pi}(s')-\mathbb{E}_{s'\sim\mathcal{P}}V^{\mathcal{P},\pi}(s')\big]. \quad (5)$$

Eq. (5) was used to design the value difference bound and model-based RL algorithm in (Luo et al. 2018). In the following section, we will combine the policy improvement theorem in Eq. (4) and the telescoping lemma in Eq. (5) to obtain an explicit form of the value difference under the case for both different policies and dynamics functions.

Soft Actor-Critic. SAC (Haarnoja et al. 2018) is an off-policy RL algorithm. It aims to maximize the cumulative return as well as the policy entropy. SAC uses the soft policy iteration to update the model parameters. In the soft policy evaluation stage, it updates the soft Q-value with parameters μ by minimizing the soft Bellman residual

$$\begin{aligned}\mathcal{L}_Q(\mu)&=\mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[\left(Q_\mu(s_t,a_t)-r_t-\gamma\bar{V}(s_{t+1})\right)^2\right],\\ &\text{with } \bar{V}(s_t)=\mathbb{E}_{a_t\sim\pi}\left[Q_{\bar{\mu}}(s_t,a_t)-\alpha\log\pi(a_t|s_t)\right],\end{aligned} \quad (6)$$

where $\mathcal{D}$ is the data replay buffer, $\bar{\mu}$ is the target Q parameters, and α is the temperature coefficient. In the soft policy improvement stage, the policy π with parameters θ is updated by minimizing the objective

$$\mathcal{L}_\pi(\theta)=\mathbb{E}_{s_t\sim\mathcal{D}}\Big[\mathbb{E}_{a_t\sim\pi_\theta}\left[\alpha\log(\pi_\theta(a_t|s_t))-Q_\mu(s_t,a_t)\right]\Big]. \quad (7)$$

The policy π in SAC is a stochastic policy which is modelled as a Gaussian with mean and covariance given by the neural network for continuous action space. In this work, we develop our algorithm on the basis of SAC to handle continuous control tasks.

The Value Relativity Lemma

The following lemma integrates the policy improvement theorem with the telescoping lemma and measures the relative difference between any two expected returns under different policies and dynamics functions.

Lemma 1. *Given two Markov Decision Processes (MDPs) denoted by $\mathcal{E}'$ and $\mathcal{E}$, who share the same state and action spaces $\mathcal{S}$, $\mathcal{A}$ and reward function r, their dynamics transition probabilities are defined as $\mathcal{P}'(s_{t+1}|s_t,a_t)$ and $\mathcal{P}(s_{t+1}|s_t,a_t)$ for any transition $(s_t,a_t)\to s_{t+1}$ in $\mathcal{E}'$ and $\mathcal{E}$ respectively. Assume the initial state distributes identically in the two MDPs that $\mathcal{P}'(s_0)=\mathcal{P}(s_0)$. Let $J(\mathcal{P},\pi)$ denote the expected return defined on dynamics $\mathcal{P}$ and policy π. Then, the relativity gap between any two expected returns under different dynamics and policies is defined as*

$$\underbrace{J(\mathcal{P}',\pi)-J(\mathcal{P},\pi')}_{\textit{relativity gap}}=\underbrace{J(\mathcal{P}',\pi)-J(\mathcal{P},\pi)}_{\textit{dynamics-induced gap}}+\underbrace{J(\mathcal{P},\pi)-J(\mathcal{P},\pi')}_{\textit{policy-induced gap}}, \quad (8)$$

such that the dynamics-induced gap can be derived from the telescoping lemma (Luo et al. 2018) in Eq. (5) *as*

$$\begin{aligned}&J(\mathcal{P}',\pi)-J(\mathcal{P},\pi)=\\ &\mathbb{E}_{s_0,a_0,\cdots\sim\mathcal{P}',\pi}\sum_{t=0}^{\infty}\gamma^t\left[r_t+\gamma V^{\mathcal{P},\pi}(s_{t+1})-Q^{\mathcal{P},\pi}(s_t,a_t)\right],\end{aligned} \quad (9)$$

and the policy-induced gap is revealed by the policy improvement theorem (Kakade and Langford 2002) in Eq. (4), rewritten by switching π and π' as

$$J(\mathcal{P},\pi)-J(\mathcal{P},\pi')=\mathbb{E}_{s_0,a_0,\cdots\sim\mathcal{P},\pi}\sum_{t=0}^{\infty}\gamma^t A^{\mathcal{P},\pi'}(s_t,a_t). \quad (10)$$

The proofs of Eqs. (9) and (10) can follow from (Luo et al. 2018) and (Kakade and Langford 2002), respectively, both using the telescoping expansion as the main technique. In Appendix A, we also provide another complete proof by directly expanding the expected returns in Eq. (8). Although these gaps in Lemma 1 have been referred to as the policy improvement theorem and telescoping lemma in (Luo et al. 2018) and (Kakade and Langford 2002), we use the terms value relativity gap, dynamics-induced gap and policy-induced gap, and refer to the above lemma as the *Value Relativity Lemma* in this paper for uniformity.

Before proposing new algorithms based on the above lemma, we need to emphasize a few points implied in Eqs. (9) and (10):

- In Eq. (9), the expectation is taken over the trajectory $s_0,a_0,\cdots$ sampled from the pair $(\mathcal{P}',\pi)$, while the value and state-action value functions in the expectation, i.e., $V^{\mathcal{P},\pi}(s_{t+1})$ and $Q^{\mathcal{P},\pi}(s_t,a_t)$, are defined on $(\mathcal{P},\pi)$. This gives a practically useful hint that given a fixed policy π, the dynamics-induced gap can be calculated by measuring the value functions $V^{\mathcal{P},\pi}(s_{t+1})$ and $Q^{\mathcal{P},\pi}(s_t,a_t)$ in the dynamics $\mathcal{P}$ (imagining this is the source environment, where infinite data can be generated cheaply to accurately evaluate the value functions), while collecting (a few) data samples in $\mathcal{P}'$ (imagining this is the target environment) to estimate the expectation.

- In Eq. (9), $\mathbb{E}_{s_{t+1}\sim\mathcal{P}'(\cdot|s_t,a_t)}[r_t + \gamma V^{\mathcal{P},\pi}(s_{t+1})] \neq Q^{\mathcal{P},\pi}(s_t,a_t)$, because the transition $(s_t,a_t) \rightarrow s_{t+1}$ takes place in $\mathcal{P}'$ instead of $\mathcal{P}$, and hence Eq. (9) is not zero whereas $\mathcal{P}'(s_{t+1}|s_t,a_t) \neq \mathcal{P}(s_{t+1}|s_t,a_t)$ happens for non-zero r_t and $V^{\mathcal{P},\pi}(s_{t+1})$ with high probability, especially for high-dimensional deep neural networks.

In the following sections, we will introduce two new algorithms inspired by the Value Relativity Lemma, where one algorithm is for fast policy transfer from the source environment to the target environment, and another algorithm updates the parameterized dynamics in the source environment to be close to the dynamics in the target environment. Then, by combining the two algorithms, we obtain the complete algorithm to fast transfer a policy from the source environment to the target environment.

Relative Policy Optimization (RPO)

As discussed previously, Eq. (9) in the Value Relativity Lemma suggests a way of estimating the dynamics-induced value gap by evaluating $Q^{\mathcal{P},\pi}(s_t,a_t)$ and $V^{\mathcal{P},\pi}(s_{t+1})$ in $\mathcal{P}$ while sampling data in $\mathcal{P}'$, given π. Practically, it is of less interest to estimate the exact dynamics-induced gap. Instead, if we have trained a policy π^* in $\mathcal{P}^{source}$ that maximizes $J(\mathcal{P}^{source},\pi)$, then we are interested in finding another $\hat{\pi}$ such that $\hat{\pi} = \arg\max_\pi[J(\mathcal{P}^{target},\pi) - J(\mathcal{P}^{source},\pi^*)]$, which improves $J(\mathcal{P}^{target},\pi)$. Normally, as long as $\mathcal{P}^{target}$ is not far from $\mathcal{P}^{source}$, finding $\hat{\pi}$ can use π^* as a warm start. Motivated by this, we propose the following theorem to get a lower bound of the dynamics-induced value gap.

Theorem 2. *Define $D_{TV}^{max}(p,q) = \max_x D_{TV}(p(\cdot|x)||q(\cdot|x))$ as the total variation divergence between two distributions $p(\cdot|x)$ and $q(\cdot|x)$, where $D_{TV}(p(\cdot|x)||q(\cdot|x)) = \frac{1}{2}\sum_y |p(y|x) - q(y|x)|$. Define $\epsilon = \max_{s,a}|A^{\mathcal{P},\pi}(s,a)|$, where $A^{\mathcal{P},\pi}(s,a) = Q^{\mathcal{P},\pi}(s,a) - V^{\mathcal{P},\pi}(s)$ is the advantage. Let $\delta_1 = D_{TV}^{max}(\mathcal{P}'(\cdot|s,a),\mathcal{P}(\cdot|s,a))$ for any $\mathcal{P}'$, $\mathcal{P}$, and $(s,a) \in \mathcal{S}\times\mathcal{A}$, and let $\delta_2 = D_{TV}^{max}(\pi'(\cdot|s),\pi(\cdot|s))$ for any $s \in \mathcal{S}$, and policies π' and π. Let $r_{max} = \max_{s,a,s'} r(s,a,s')$ be the max reward for all (s,a,s').*[1] *Let $\Delta^{\mathcal{P}',\mathcal{P}}(\pi) = J(\mathcal{P}',\pi) - J(\mathcal{P},\pi)$ be a function of $\mathcal{P}'$, $\mathcal{P}$ and π. Now, we import a new policy π' and define the following function*

$$\sum_{t=0}^{\infty}\gamma^t\,\mathbb{E}_{s_t\sim\mathcal{P}',\pi'}\sum_{a_t}\pi(a_t|s_t)\sum_{s_{t+1}}\mathcal{P}'(s_{t+1}|s_t,a_t)\cdot \left[r(s_t,a_t,s_{t+1}) + \gamma V^{\mathcal{P},\pi'}(s_{t+1}) - Q^{\mathcal{P},\pi'}(s_t,a_t)\right], \quad (11)$$

as an approximation of $\Delta^{\mathcal{P}',\mathcal{P}}(\pi)$ by both sampling $s_0,a_0,\cdots,s_t$ and evaluating $V^{\mathcal{P},\pi'}$ and $Q^{\mathcal{P},\pi'}$ using π'. Then, we have the following lower bound

$$\Delta^{\mathcal{P}',\mathcal{P}}(\pi) \geq L_{\pi'}(\pi) - C_1,$$
$$\text{where } C_1 = \frac{4\gamma r_{max}\delta_1}{(1-\gamma)^2}\min\left(\frac{\delta_2(\gamma^2+2)}{1-\gamma}, 1+\frac{\delta_2}{1-\gamma}\right). \quad (12)$$

[1]The above terms and assumptions have been widely used in RL in the literature such as (Kakade and Langford 2002; Schulman et al. 2015; Luo et al. 2018; Janner et al. 2019).

The proof is provided in Appendix C. In Theorem 2, we import $L_{\pi'}(\pi)$ defined on two policies π and π', because in practice we will need an algorithm iterating over the current policy parameter and its old parameter since last update. This technique will be adopted multiple times in the following context. Based on Theorem 2, we can further obtain the following lower bound of the entire relativity gap.

Proposition 3. *The entire relativity gap in Eq. (8) has the following lower bound*

$$J(\mathcal{P}',\pi) - J(\mathcal{P},\pi') \geq \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\mathcal{P}',\pi'},a,s'\sim\mathcal{P}',\pi'}\frac{\pi(a|s)}{\pi'(a|s)}\cdot [r(s,a,s') + \gamma V^{\mathcal{P},\pi'}(s') - V^{\mathcal{P},\pi'}(s)] - C_2, \quad (13)$$

where s' is the next state that $(s,a)\rightarrow s'$, and

$$C_2 = \frac{2\gamma\epsilon(\delta_1+2\delta_2^2)}{(1-\gamma)^2} + \frac{4\gamma r_{max}}{(1-\gamma)^2}\cdot\min\left(\frac{\delta_2(\gamma^2+2)}{1-\gamma}, 1+\frac{\delta_2}{1-\gamma}\right), \quad (14)$$

is a constant relying on the dynamics discrepancy δ_1 and policy discrepancy δ_2.

The proof is provided in Appendix D. By taking $\pi = \pi_\theta$ and $\pi' = \pi_{\theta_{old}}$ for some policy parameters θ and its old version θ_{old} since last update, Proposition 3 suggests maximizing the following objective

$$\mathbb{E}_{\substack{s\sim d^{\mathcal{P}',\pi_{\theta_{old}}},\\ a,s'\sim\mathcal{P}',\pi_{\theta_{old}}}}\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}\cdot[r(s,a,s') + \gamma V^{\mathcal{P},\pi_{\theta_{old}}}(s')], \quad (15)$$

by noting that $V^{\mathcal{P},\pi'}(s)$ serves as a baseline and does not affect the policy gradient. Eq. (15) is very similar to Eq. (15) in (Schulman et al. 2015), but recall that $r(s,a,s') + \gamma V^{\mathcal{P},\pi_{\theta_{old}}}(s') \neq Q^{\mathcal{P},\pi_{\theta_{old}}}(s,a)$, because the transition $(s,a)\rightarrow s'$ takes place in $\mathcal{P}'$ instead of $\mathcal{P}$. Empirically, we consider a local approximation for Eq. (15) as

$$\mathbb{E}_{\substack{s\sim d^{\mathcal{P}',\pi_{\theta_{old}}},\\ a\sim\pi_{\theta_{old}},s'\sim\mathcal{P}}}\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}\cdot[r(s,a,s') + \gamma V^{\mathcal{P},\pi_{\theta_{old}}}(s')]$$
$$= \mathbb{E}_{\substack{s\sim d^{\mathcal{P}',\pi_{\theta_{old}}},\\ a\sim\pi_\theta,s'\sim\mathcal{P}}} Q^{\mathcal{P},\pi_{\theta_{old}}}(s,a), \quad (16)$$

by approximating only one step transition $(s,a)\rightarrow s'$ in $\mathcal{P}$. Note that Eq. (16) still differs from standard Q-value maximization that s is sampled from $\mathcal{P}'$ while the Q-value is computed in $\mathcal{P}$. Now, to optimize the policy through maximizing Eq. (16), we follow Eq. (7) in SAC by sampling s from $\mathcal{P}'$ while estimating the Q-value in $\mathcal{P}$. We refer to this procedure as *Relative Policy Optimization* (RPO).

Now, we still need to look into the tightness of the bound in Proposition 3. In this section, we have considered fixed and diverse $\mathcal{P}$ and $\mathcal{P}'$, and hence δ_1 is always a positive constant. Then, C_2 in Proposition 3 will never approach zero by noting that $C_2 > \frac{2\gamma\epsilon}{(1-\gamma)^2}\delta_1 > 0$ even if δ_2 is sufficiently small, because the dynamics discrepancy prevents this. With the existence of $\delta_1 > 0$, $J(\mathcal{P}',\pi_\theta)$ is improved over $J(\mathcal{P},\pi_{\theta_{old}})$ (a constant at the current step) only when we can improve the objective in Eq. (15) by at least $(1-\gamma)C_2$. Worse still, starting from a well trained policy π_{θ^*} in $\mathcal{P}$, as RPO updates θ, $\pi_{\theta_{old}}$ will be gradually far from π_{θ^*}, and therefore $J(\mathcal{P},\pi_{\theta_{old}})$ might decrease. If this happens,

maximizing the value gap $J(\mathcal{P}', \pi_\theta) - J(\mathcal{P}, \pi_{\theta_{old}})$ will not guarantee the increase of $J(\mathcal{P}', \pi_\theta)$. The ideal case is that RPO still keeps π_θ perform well in $\mathcal{E}$. As we can imagine, if this happens, π_θ will be updated towards a robust policy that performs well in both $\mathcal{E}'$ and $\mathcal{E}$. Indeed, as we will show in our experiments, as long as $\mathcal{P}'$ is not too far away from $\mathcal{P}$, optimizing RPO is able to obtain such a robust policy; however, once $\mathcal{P}'$ differs from $\mathcal{P}$ too much, RPO will fail to transfer the policy to the target environment. This can be remarked as a disadvantage of the RPO algorithm which requires a relatively small dynamics gap, i.e., δ_1. Overall, to guarantee the success of policy transfer, we need further to eliminate the dynamics discrepancy. This is possible when $\mathcal{P}$ (the dynamics in the source environment) is trainable, as considered in physical dynamics modeling (Hu et al. 2020; Freeman et al. 2021; Werling et al. 2021) and model-based RL methods (Janner et al. 2019). In the next section, we will consider a trainable $\mathcal{P}$.

Relative Transition Optimization (RTO)

In this section, given fixed π and $\mathcal{P}'$, we consider a trainable $\mathcal{P}$. Suppose $\mathcal{P}_\phi(s'|s,a)$ is parameterized by ϕ for any transition $(s,a) \to s'$. In the following theorem, we will import three dynamics quantities $\mathcal{P}'$, $\mathcal{P}_\phi$ and $\mathcal{P}_{\phi'}$, where $\mathcal{P}'$ can be treated as the dynamics in the target environment, and $\mathcal{P}_\phi$ and $\mathcal{P}_{\phi'}$ are two variant dynamics functions parameterized by ϕ and ϕ', respectively.

Theorem 4. *With the definitions in Theorem 2, introduce the following function*

$$L_{\phi'}(\phi) = \sum_{t=0}^{\infty} \gamma^t \, \mathbb{E}_{\tau\sim\mathcal{P}',\pi}\Big[\mathbb{E}_{s_{t+1}\sim\mathcal{P}'}\big[r_t + \gamma V^{\mathcal{P}_{\phi'},\pi}(s_{t+1})\big] - \mathbb{E}_{s_{t+1}\sim\mathcal{P}_\phi}\big[r_t + \gamma V^{\mathcal{P}_{\phi'},\pi}(s_{t+1})\big]\Big], \tag{17}$$

as an approximation of $\Delta^{\mathcal{P}',\mathcal{P}_\phi}(\pi)$ by evaluating the value V using $\mathcal{P}_{\phi'}$ instead of $\mathcal{P}_\phi$. Then, we have

$$|\Delta^{\mathcal{P}',\mathcal{P}_\phi}(\pi)| \le |L_{\phi'}(\phi)| + \frac{4\gamma\delta_1 r_{max}}{(1-\gamma)^2} \min\left(\frac{\delta_2(\gamma^2+1)}{1-\gamma}, 1\right). \tag{18}$$

In order to reduce the dynamics-induced gap by updating ϕ, we have to minimize $|\Delta^{\mathcal{P}',\mathcal{P}_\phi}(\pi)|$. Based on Theorem 4, we can alternatively minimize $|L_{\phi'}(\phi)|$. Empirically, if we consider $\mathcal{P}_\phi$ as a probability function, then by noting that

$$L_{\phi'}(\phi) = \sum_{t=0}^{\infty} \gamma^t \, \mathbb{E}_{\tau\sim\mathcal{P}',\pi} \sum_{s'} \big(\mathcal{P}'(s'|s,a) - \mathcal{P}_\phi(s'|s,a)\big) \cdot \big(r + \gamma V^{\mathcal{P}_{\phi'},\pi}(s')\big), \tag{19}$$

and taking ϕ' as the old version of ϕ since the last update, we consider a square loss on minimizing $|L_{\phi'}(\phi)|$ as

$$\text{minimize}_\phi \, \mathbb{E}_{\tau\sim\mathcal{P}',\pi} \sum_{s'} \big(\mathcal{P}'(s'|s,a) - \mathcal{P}_\phi(s'|s,a)\big)^2 \cdot \big(r + \gamma V^{\mathcal{P}_{\phi'},\pi}(s')\big)^2. \tag{20}$$

That is, we sample (s,a) in $\mathcal{E}'$ using π and optimize Eq. (20). We refer to the above optimization as *Relative Transition Optimization* (RTO). It is easy to see that RTO implies the standard model-based RL methods, which directly train ϕ using supervised learning, i.e.,

$$\text{minimize}_\phi \, \mathbb{E}_{\tau\sim\mathcal{P}',\pi} \sum_{s'} \big(\mathcal{P}'(s'|s,a) - \mathcal{P}_\phi(s'|s,a)\big)^2. \tag{21}$$

Indeed, the objective of RTO in Eq. (20) can be viewed as a weighted form of supervised learning, with the weight $(r + \gamma V^{\mathcal{P}_{\phi'},\pi}(s'))^2$ showing that transitions with larger values evaluated by $\mathcal{P}_{\phi'}$ and π should be optimized more aggressively. This is reasonable from the perspective of fitting state values, instead of purely fitting the transition dynamics. Therefore, RTO provides a theoretical explanation for the standard model-based RL methods by reducing the dynamics-induced value gap, and absorbs standard model-based RL as a special case in RTO.

In this work, we learn the soft value function by following SAC (Haarnoja et al. 2018). Then, we give the final objective for minimizing the dynamics-induced gap as

$$\mathbb{E}_{\tau\sim\mathcal{P}',\pi} \sum_{s'} \big(\mathcal{P}'(s'|s,a) - \mathcal{P}_\phi(s'|s,a)\big)^2 w(r,s') \quad \text{with } w(r,s') = \big(r + \gamma(Q^{\mathcal{P}_{\phi'},\pi}(s',\pi(s')) - \log\pi(s'))\big)^2, \tag{22}$$

where the Q-value will be optimized to minimize the soft Bellman residual. In our implementation, we normalize the dynamics weight w in the replay buffer $\mathcal{D}_{source}$ to guarantee numerical stability by $\hat{w} = \frac{w - w_{min}}{w_{max} - w_{min}} + \epsilon$, where w_{min} and w_{max} are the smallest and largest weights in the buffer $\mathcal{D}_{source}$, and ϵ controls the minimum weight for updating.

In the experiments, we try both probabilistic and deterministic dynamics models. For the latter case, we adopt a deterministic physical dynamics model, similar to recently proposed differentiable simulators (Hu et al. 2020; Freeman et al. 2021; Werling et al. 2021), which take advantage of the inherent physical processes. Such a model is much more efficient than a pure neural network based dynamics model, because only a few scalars such as mass, length, gravity, etc., are trainable parameters in the physical systems. However, differentiable simulator requires specific engineering on the considered environments and is not general for broad usage.

Relative Policy-Transition Optimization (RPTO) Algorithm

Combining RPO and RTO, we finally obtain the complete Relative Policy-Transition Optimization (RPTO) algorithm. We provide the overall algorithm of RPTO in Algorithm 1.

As we observe in Algorithm 1, in the main loop of RPTO, the policy π_θ interacts with the two environments simultaneously and pushes the data into two buffers separately. In step 5, we sample a mini-batch from $\mathcal{D}_{source}$ to update the soft Q-value in Eq. (6). In step 6, we sample a mini-batch from $\mathcal{D}_{target}$ to update the policy parameter θ according to Eq. (15) and Eq. (7) in RPO, and also update the dynamics ϕ according to Eq. (22) from RTO. Therefore, RPTO combines data collection from two environments, RPO and RTO in a closed loop, and this offers a principled learning framework for policy transfer. Indeed, steps 5 and 6 can be parallelized, as long as the Q-value function $Q^{\mathcal{P}_\phi^{source},\pi_\theta}$ (suppose

Algorithm 1: Relative Policy-Transition Optimization.

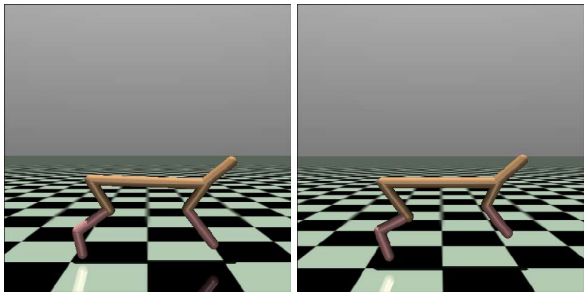
Input: The source and target environments $\mathcal{E}^{source}$ and $\mathcal{E}^{target}$, and their dynamics $\mathcal{P}^{source}_{\phi_0}$ and $\mathcal{P}^{target}$, where ϕ_0 can accurately describe the initial source dynamics; a well-trained policy π_{θ_0} in $\mathcal{E}^{source}_{\phi_0}$;
1. Create two empty replay buffers $\mathcal{D}_{source}$ and $\mathcal{D}_{target}$;
2. Initialize $\theta = \theta_0$ and $\phi = \phi_0$;
repeat
3. Using π_θ to interact with $\mathcal{E}^{source}_{\phi}$ and push the generated trajectories into $\mathcal{D}_{source}$;
4. Using π_θ to interact with $\mathcal{E}^{target}$ and push the generated trajectories into $\mathcal{D}_{target}$;
5. Sample a mini-batch $\{(s, a, s')\}_{source} \sim \mathcal{D}_{source}$, then update $Q^{\mathcal{P}^{source}_{\phi}, \pi_\theta}$ by minimizing the soft Bellman residual;
6. Sample a mini-batch $\{(s, a, s')\}_{target} \sim \mathcal{D}_{target}$, then apply RPO and RTO to update π_θ and $\mathcal{P}^{source}_{\phi}$ respectively;
until Some convergence criteria is satisfied.

it is parameterized by μ) does not share parameters with π_θ, i.e., μ and θ are independent. Also, μ can be updated more frequently because it only requires data generated from the source environment, and the more accurate the Q-value function in the source environment is, the more accurate the relative policy gradient in step 6 will be estimated. For the dynamics parameter ϕ, the case turns out to be different, and it is often not a good choice to update ϕ as fast as we can. To understand this, we need to explain how RPTO transfers the policy to the target environment in advance.

As we can imagine, as $\mathcal{P}^{source}_{\phi}$ approaches $\mathcal{P}^{target}$ gradually by updating ϕ, the policy π_θ is able to interact with a sequence of smoothly varying 'source' environments. During this training period, the policy observes much more diverse transitions that lie between the initial source environment (with dynamics $\mathcal{P}^{source}_{\phi_0}$) and the target environment. These diverse transitions are very helpful to encourage the agent to explore a robust policy. On the other hand, since the policy is initialized with a well-trained θ_0 in $\mathcal{P}^{source}_{\phi_0}$, smoothly varying $\mathcal{P}^{source}_{\phi_0}$ to reach $\mathcal{P}^{target}$ generates a sequence of environments that naturally provide a curriculum learning scheme, and this is very similar to what is considered in the recently emerged Environment Design (Dennis et al. 2020). Now, we could answer the question in the last paragraph that why ϕ should not be updated aggressively: a slowly and smoothly varying ϕ provides more chances for the agent to see diverse transitions. Actually, if $\mathcal{P}^{source}_{\phi}$ approaches $\mathcal{P}_{target}$ too fast, RPTO will be similar to directly training SAC in the target environment with a warm start. In our implementation, we apply fewer dynamics updates than the original MBPO (Janner et al. 2019).

Experiments

In this section, we perform comprehensive experiments across diverse control tasks to assess the efficacy of our RPTO. Additionally, we analyze how RPT, RTO, and RPTO individually contribute to learning.



(a) Source (b) Target

Figure 1: An illustration of the source and target environments on HalfCheetah.

Environment Settings. We experiment on a set of MuJoCo continuous control tasks with the standard neural network (NN) based probabilistic dynamics model, including Ant, Hopper, HalfCheetah, Walker2d and Swimmer. We first modify the default observation space and action space in some tasks as the source environments, while for each task, we arbitrarily modify some of its physical factors to create the corresponding target environments. For MuJoCo tasks, we randomly add a $\pm 20\%$ variation to the length of all joints of the robot and the friction of the environment. We use the default setting in OpenAI gym as our source environment. Figure 1 shows an illustration of the source and target environments. All joints in the target environment have different lengths than in the source environment. Please see Appendix G for detailed settings on environments.

Baselines. We first pre-train a converged policy and dynamics using the pure model-based algorithm in the source environment, and the pre-trained policy and dynamics will be used as a warm start when transferring to the target environment. For the policy transfer stage, SAC (Haarnoja et al. 2018) and TRPO (Schulman et al. 2015) with the pre-trained policy as initialization, denoted as SAC-warm and TRPO-warm respectively, will be used as the baseline methods. We also involve the state-of-the-art model-based methods MBPO (Janner et al. 2019), SLBO (Luo et al. 2018), PTP (Wu et al. 2022) and PDML (Wang et al. 2023), as the baseline methods. Both of them use their pre-trained model from the source environment as initialization to achieve fair comparison in transfer tasks. All the compared algorithms are open-sourced, thus we can use their original implementations to avoid biased evaluation.

Practical Implementation. We build the neural dynamics model by following MBPO, and use an ensemble network to learn the dynamics. We set the ensemble size to 7 and each ensemble network has 4 fully connected layers with 400 units. Each head of dynamics model is a probabilistic neural network which outputs Gaussian distribution with diagonal covariance: $p^i_\phi(s_{t+1}, r_t|, s_t, a_t) = \mathcal{N}\big(\mu^i_\phi(s_t, a_t), \sum^i_\phi(s_t, a_t)\big)$. We set the model horizon to 1 and the replay ratio of dynamics to 1 for all environments. Other implementation details are provided in Appendix G.

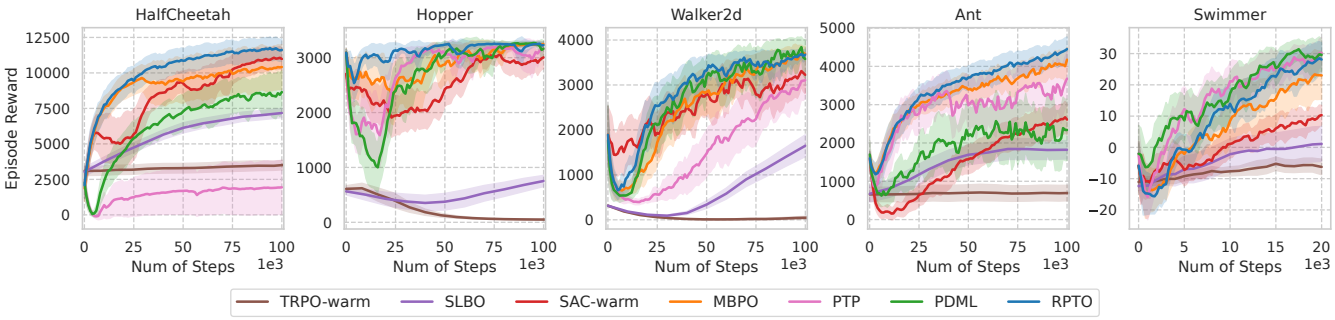


Figure 2: Overall performance on MuJoCo tasks.

For every environment, we create 4 different target environments, and we repeat 4 times for each target environment. Therefore, we totally repeat 16 times for each environment to compute the mean curve with 95% confidence interval.

Benchmark Results. We show the results of all methods on the MuJoCo tasks in Figure 2. In these environments, we first pre-train the policy and the neural network (NN) dynamics model with MBPO and SLBO. Then, we use them as warm start in the RPTO, MBPO and SLBO methods. Moreover, we use the pre-trained policy in MBPO and SLBO as warm start for SAC-warm and TRPO-warm, respectively. As we can see, the model-based method MBPO can serve as a strong baseline for policy transfer. However, in all the tasks, the RPTO methods demonstrate superiority over the other baselines in terms of both fast policy transfer and even better asymptotic convergence, because RPTO sees much more diverse dynamics that promote exploration, as explained at the end of the RPTO description section.

Didactic Example. We then use CartPole as an illustrative environment to show how the proposed algorithms are practically useful. In this experiment, we evaluate the RPO, RTO and RPTO with deterministic physical dynamics model, because the physical systems are explicitly coded in CartPole, allowing us to conveniently build physical dynamics model with only a few trainable factors. In this didactic example, only the pole length is treated as the trainable parameter, i.e., ϕ in CartPole only contains one free parameter. Similarly, we also first pre-train a converged policy in the source environment, but we don't need to pre-train the dynamics model this time. We use deterministic physical dynamics to minimize the dynamics gap between the source and the target environment.

Figure 3 reports the evaluation results of our RPO, RTO and RPTO. Figure 3(a) demonstrates that RPO (without RTO) is sufficient to successfully transfer the policy to the target environment with a pole length of 1.2, and RPO transfers the policy faster than SAC-warm. In Figure 3(b), by varying the pole length to test more target environments, we find that when the pole length difference becomes larger, RPO fails to obtain a stable policy in the target environment. These results are consistent with our analysis at the RPO section. Figure 3(c) evaluates the performance of RTO. As we can observe, RTO can optimize ϕ to converge to the true pole length in the target environment. Finally, Figure 3(d)

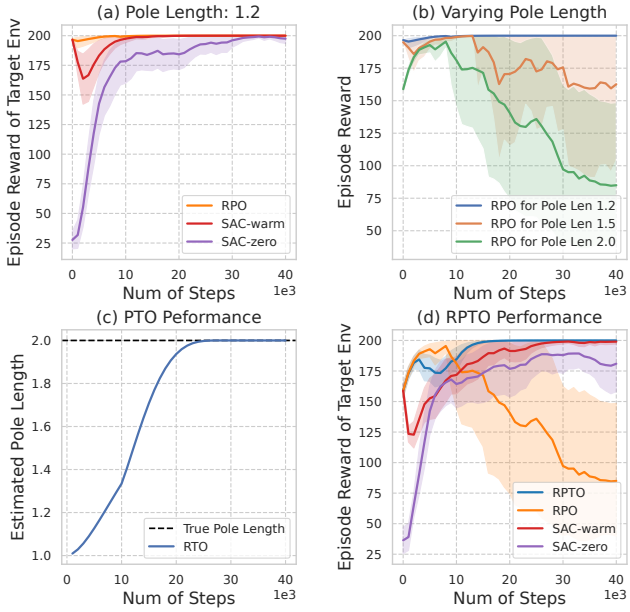


Figure 3: Illustrative experiments in CartPole. The length of pole in source environment is 1.0.

reports the RPTO's performance on the tagret environment with the pole length of 2.0. As we can observe, RPTO transfers the policy much faster than all the other methods.

Future Work

Our work is related to model-based RL, environment design and sim2real problems. There are some future directions worth investigating. For example, as we have discussed before, controlling the update step size or frequency of RTO can provide better curriculum learning or environment design (although in this paper we simply fix the learning rate of RTO). Connecting this with meta-RL and current environment design algorithms is a promising future direction. Moreover, the theory and algorithms in this paper are orthogonal to other techniques commonly used in sim2real tasks, such as domain adaptation, domain randomization, augmented observation, etc., and all of these methods can be integrated together into sim2real applications.

Acknowledgments

Baoxiang Wang is partially supported by National Natural Science Foundation of China (62106213, 72150002, 72394361) and Shenzhen Science and Technology Program (RCBS20210609104356063, JCYJ20210324120011032).

References

Bakker, B. 2001. Reinforcement learning with long short-term memory. *Advances in Neural Information Processing Systems (NeurIPS)*, 14.

Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Debiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Delecki, H.; Corso, A.; and Kochenderfer, M. 2023. Model-based Validation as Probabilistic Inference. In *Learning for Dynamics and Control Conference*, 825–837. PMLR.

Dennis, M.; Jaques, N.; Vinitsky, E.; Bayen, A.; Russell, S.; Critch, A.; and Levine, S. 2020. Emergent Complexity and Zero-shot Transfer via Unsupervised Environment Design. *Advances in Neural Information Processing Systems (NeurIPS)*, 33: 13049–13061.

Freeman, C. D.; Frey, E.; Raichuk, A.; Girgin, S.; Mordatch, I.; and Bachem, O. 2021. Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation.

Fujimoto, S.; Meger, D.; and Precup, D. 2019. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning (ICML)*, 2052–2062.

Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Han, L.; Xiong, J.; Sun, P.; Sun, X.; Fang, M.; Guo, Q.; Chen, Q.; Shi, T.; Yu, H.; Wu, X.; et al. 2020. Tstarbot-x: An open-sourced and comprehensive study for efficient league training in starcraft ii full game. *arXiv preprint arXiv:2011.13729*.

Han, L.; Zhu, Q.; Sheng, J.; Zhang, C.; Li, T.; Zhang, Y.; Zhang, H.; Liu, Y.; Zhou, C.; Zhao, R.; et al. 2023. Lifelike agility and play on quadrupedal robots using reinforcement learning and generative pre-trained models. *arXiv preprint arXiv:2308.15143*.

Hu, Y.; Anderson, L.; Li, T.-M.; Sun, Q.; Carr, N.; Ragan-Kelley, J.; and Durand, F. 2020. DiffTaichi: Differentiable Programming for Physical Simulation. In *International Conference on Learning Representations (ICLR)*.

Janner, M.; Fu, J.; Zhang, M.; and Levine, S. 2019. When to Trust Your Model: Model-Based Policy Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 32: 12519–12530.

Kaiser, L.; Babaeizadeh, M.; Miłos, P.; Osiński, B.; Campbell, R. H.; Czechowski, K.; Erhan, D.; Finn, C.; Kozakowski, P.; Levine, S.; et al. 2019. Model Based Reinforcement Learning for Atari. In *International Conference on Learning Representations (ICLR)*.

Kakade, S.; and Langford, J. 2002. Approximately optimal ap- proximate reinforcement learning. In *International Conference on Machine Learning (ICML)*, volume 2, 267–274.

Li, S.; Xu, J.; Dong, H.; Yang, Y.; Yuan, C.; Sun, P.; and Han, L. 2023. The Fittest Wins: a Multi-Stage Framework Achieving New SOTA in ViZDoom Competition. *IEEE Transactions on Games*.

Luo, Y.; Xu, H.; Li, Y.; Tian, Y.; Darrell, T.; and Ma, T. 2018. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. *arXiv preprint arXiv:1807.03858*.

Lyu, J.; Ma, X.; Li, X.; and Lu, Z. 2022. Mildly conservative Q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 1711–1724.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; and Ostrovski, G. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529.

Rosete-Beas, E.; Mees, O.; Kalweit, G.; Boedecker, J.; and Burgard, W. 2023. Latent plans for task-agnostic offline reinforcement learning. In *Conference on Robot Learning*, 1838–1849. PMLR.

Schaefer, A. M.; Udluft, S.; and Zimmermann, H.-G. 2007. A recurrent control neural network for data efficient reinforcement learning. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 151–157. IEEE.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 1889–1897.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Siegel, N. Y.; Springenberg, J. T.; Berkenkamp, F.; Abdolmaleki, A.; Neunert, M.; Lampe, T.; Hafner, R.; Heess, N.; and Riedmiller, M. 2020. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; and Lanctot, M. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587): 484.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; and Bolton, A. 2017. Mastering the game of Go without human knowledge. *Nature*, 550(7676): 354.

Taylor, M. E.; and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7).

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster Level in StarCraft II using Multi-Agent Reinforcement Learning. *Nature*, 575(7782): 350–354.

Wang, X.; Wongkamjan, W.; Jia, R.; and Huang, F. 2023. Live in the moment: Learning dynamics model adapted to evolving policy. In *International Conference on Machine Learning*, 36470–36493. PMLR.

Werling, K.; Omens, D.; Lee, J.; Exarchos, I.; and Liu, C. K. 2021. Fast and Feature-Complete Differentiable Physics for Articulated Rigid Bodies with Contact. *arXiv preprint arXiv:2103.16021*.

Wu, Z.; Yu, C.; Chen, C.; Hao, J.; and Zhuo, H. H. 2022. Plan To Predict: Learning an Uncertainty-Foreseeing Model For Model-Based Reinforcement Learning. *Advances in Neural Information Processing Systems*, 35: 15849–15861.

Xu, J.; Li, S.; Yang, R.; Yuan, C.; and Han, L. 2023. Efficient Multi-Goal Reinforcement Learning via Value Consistency Prioritization. *Journal of Artificial Intelligence Research*, 77: 355–376.

Zhan, W.; Huang, B.; Huang, A.; Jiang, N.; and Lee, J. 2022. Offline reinforcement learning with realizability and single-policy concentrability. In *Conference on Learning Theory*, 2730–2775. PMLR.

Zhu, Z.; Lin, K.; Jain, A. K.; and Zhou, J. 2023. Transfer learning in deep reinforcement learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.