REINFORCEMENT LEARNING WITH SUPERVISION BEYOND ENVIRONMENTAL REWARDS

BY

TANMAY GANGWANI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2021

Urbana, Illinois

Doctoral Committee:

Professor Jian Peng, Chair
Professor David Forsyth
Professor ChengXiang Zhai
Professor Saurabh Gupta
Dr. Katja Hofmann, Microsoft Research

ABSTRACT

Reinforcement Learning (RL) is an elegant approach to tackle sequential decision-making problems. In the standard setting, the task designer curates a reward function and the RL agent's objective is to take actions in the environment such that the long-term cumulative reward is maximized. Deep RL algorithms—that combine RL principles with deep neural networks—have been successfully used to learn behaviors in complex environments but are generally quite sensitive to the nature of the reward function. For a given RL problem, the environmental rewards could be sparse, delayed, misspecified, or unavailable (*i.e.,* impossible to define mathematically for the required behavior). These scenarios exacerbate the challenge of training a stable deep-RL agent in a sample-efficient manner.

In this thesis, we study methods that go beyond a direct reliance on the environmental rewards by generating additional information signals that the RL agent could incorporate for learning the desired skills. We start by investigating the performance bottlenecks in delayed reward environments and propose to address these by learning surrogate rewards. We include two methods to compute the surrogate rewards using the agent-environment interaction data. Then, we consider the imitation-learning (IL) setting where we don't have access to any rewards, but instead, are provided with a dataset of expert demonstrations that the RL agent must learn to reliably reproduce. We propose IL algorithms for partially observable environments and situations with discrepancies between the transition dynamics of the expert and the imitator. Next, we consider the benefits of learning an ensemble of RL agents with explicit diversity pressure. We show that diversity encourages exploration and facilitates the discovery of sparse environmental rewards. Finally, we analyze the concept of sharing knowledge between RL agents operating in different but related environments and show that the information transfer can accelerate learning.

To Mom and Dad.

For their endless love and support!

ACKNOWLEDGMENTS

This thesis would not have been possible without the support and impetus from numerous people. This section is devoted to *credit assignment in hindsight* – thanking and giving credit to the amazing folks who propelled me in my journey at the University of Illinois Urbana-Champaign (UIUC). First and foremost, I am grateful to my thesis advisor, Professor Jian Peng. The brainstorming sessions with Jian were crucial for germinating our research contributions. I also appreciate him for providing me with financial assistance and the freedom to explore diverse directions in reinforcement learning. Next, I thank my doctoral committee members—Professor David Forsyth, Professor ChengXiang Zhai, Professor Saurabh Gupta, and Dr. Katja Hofmann (Microsoft Research)—for their contribution in evaluating and strengthening the presentation of the thesis material. I started my Ph.D. program after completing a Masters in the area of Computer Architecture, under the guidance of Professor Josep Torrellas. Josep is a remarkably inspiring person with immense passion and dedication to his work. I learned the art of research and scientific discipline from Josep.

Several articles in this thesis were accomplished by joint efforts with my collaborators. I was fortunate to work with these wonderful people – Professor Adam Morrison, Professor Carl Gunter, Professor Yuan Zhou, Professor Qiang Liu, Joel Lehman (internship at Uber), Jason Rolfe (internship at D-Wave), Azin Heidarshenas, Serif Yesil, Yunhui Long, Haris Mughees, and Michael Wan. Michael was an undergraduate student at UIUC during our collaboration. He is incredibly smart, and I feel privileged to have had the opportunity to mentor him on a couple of research projects.

At UIUC, I met some beautiful people and built special memories that would stay with me forever. These folks are a wellspring of prudent advice and emotional comfort. Thank you, Pulkit Budhiraja, Aditi Khullar, Gaurav Singh, Anmol Kumaar, Murali Subramanian, Anand Ramachandran, Aditya Agarwal, Mengjia Yan, Bhargava Gopireddy, Dimitrios Skarlatos, Yasser Shalabi, Thomas Shull, Jiho Choi, Raghavendra Pothukuchi, Antonio Garcia, Wooil Kim, Apostolos Kokolis, Anand Bhattad, Unnat Jain, Ketan Mittal, Kanika Narang, Xiaoming Zhao, Kshitij Gupta, Sandeep Dasgupta, Haris Mughees – you all made the university a second home for me. A shout out to my friends in New Delhi – Ritwik Sethi and Mayank Goel – who have stood by my side since my high school days.

This thesis is dedicated to my parents. I am at a juncture in life where I am beginning to realize the trials and tribulations of navigating through the complex real-world (yes, graduate school was a cocoon). I have so much better understanding of the sacrifices you two have

made over many years to ensure that I have a rock-solid platform for growth. I look up to you, I try to emulate you – today and always.

There were plenty of celebratory events during the program – paper submissions, paper acceptances, conference parties, etc. However, they all pale in the light of the mega-event in 2020, when I got married to Jyoti Aneja. In Jyoti, I found a loving and caring partner who constantly pushed me to become a better version of myself. You have shaped me in delightfully positive ways. Danke, liebe!

TABLE OF CONTENTS

CHAPTER 1: THESIS OVERVIEW

In the Reinforcement Learning (RL) framework (Sutton & Barto, 2018), an *agent* sequentially interacts with an external *environment* in discrete timesteps, with the objective of learning a sequence of decisions or *actions* that lead to accumulation of the maximum amount of *rewards* from the environment. The RL framework is applicable to a wide array of decision-making problems in domains such as robotics (Peng et al., 2020), healthcare (Yu et al., 2019), industrial process control (Hein et al., 2017), hardware design (Khadka et al., 2020), and recommendation systems (Afsar et al., 2021), among others. Recent years have witnessed significant progress in the development of RL algorithms and their application to complex environments (Mnih et al., 2015; Lillicrap et al., 2015; Silver et al., 2017,; Dabney et al., 2018; Vinyals et al., 2019). This has largely been driven by the advancements in design and understanding of deep neural networks, that are used in the Deep-RL algorithms as function approximators for feature extraction, value estimation, and action selection.

Different from the supervised learning setup, an RL agent collects the training data via an online closed-loop interaction with the environment. The environmental reward provides the supervision to guide the agent towards learning the preferred long-term task behavior. The reward function is the most concise definition of a task (Abbeel & Ng, 2004). From a learning perspective, it is desirable that every action of the agent from a given state be quantifiable under the reward function. However, such *dense* rewards are seldom available for the environments of interest, since distilling the desired behavior for a task into a robust mathematical function is an extremely complicated proposition. Realistically, environmental rewards could be sparse, delayed, misspecified (deceptive), or even completely missing. This scarcity of supervision often renders deep-RL algorithms unstable and sample-inefficient. ***The central theme of this thesis is to study alternate sources of supervision, beyond the environmental rewards, that could accelerate deep-RL algorithms.***

1.1 MAIN CONTRIBUTIONS

In this thesis, we analyze four different sources of auxiliary supervision. These are surrogate (proxy) rewards, expert demonstrations, diversity pressure, and knowledge sharing.

1.1.1 Supervision via Surrogate Rewards

We consider *delayed*-reward environments which are characterized by a finite delay between the time the agent takes an action and when the corresponding reward from the environment

is provided to it (Bacchus et al., 1996). Deep-RL algorithms struggle to learn with delayed rewards due to the fundamental issue of long-term temporal credit assignment (Minsky, 1961; Sutton, 1984). Credit assignment refers to the ability of the agent to attribute actions to consequences that may occur after a long time interval. Several approaches have been proposed to improve learning under reward delays, including multi-step backups for fast reward propagation (Kozuno et al., 2021), gradient-free policy search methods (Salimans et al., 2017), and learning surrogate rewards (Singh et al., 2009; Sheikh et al., 2020; Zheng et al., 2020). When the environmental rewards are not conducive for direct use in the policy-gradient or the value-estimation objectives, surrogate (or intrinsic) reward functions that are discovered from the agent-environment interaction data are an appealing alternative.

We contribute two methods to learn dense surrogate rewards that boost the performance of a deep-RL agent (Gangwani et al., 2018, 2020). In our first approach, we train a parametric reward generator by building on the idea of self-imitation learning. The reward network and the policy are trained iteratively in a saddle point optimization similar to GANs. In the second method, we compute non-parametric rewards from the agent-environment data. We highlight that the non-parametric rewards could be intuitively interpreted as a uniform return redistribution mechanism, and learning with them is equivalent to optimizing a modified form of the RL objective with smoothing in the trajectory space.

1.1.2 Supervision via Expert Demonstrations

Humans learn by observation and imitation. It is, therefore, prudent to study the imitation-learning (IL) paradigm, where an expert illustrates how to solve the given task with some exemplar demonstrations. IL algorithms train the agent to mimic the demonstrated behavior, without access to any environmental rewards. A large volume of work has been done on IL resulting in approaches such as Behavioral Cloning (Pomerleau, 1991; Ross et al., 2011), that frames IL as supervised learning of expert actions, and Inverse-RL (Ng et al., 2000; Abbeel & Ng, 2004), that attempts to recover the reward function for which the expert demonstrations are optimal. Recent IL algorithms that build on the probabilistic maximum entropy inverse-RL idea (Ziebart et al., 2008) have been successfully applied to locomotion and robotic-manipulation tasks in high dimensions (Ho & Ermon, 2016; Finn et al., 2016).

The demonstrations in IL typically include observations (or states) and the corresponding expert actions. The more practical yet challenging setting is "IL from observations only" (abbreviated ILO), *i.e.,* without the expert actions. ILO algorithms are compelling since they enable imitation across environments with heterogeneity in either the action-space or the transition dynamics, and in scenarios where expert actions are difficult to measure. We

design two algorithms for ILO with model misspecification – where the transition dynamics model of the learner is a perturbed version of that of the expert. In the first method (Gangwani & Peng, 2020), we start with the principle of maximum entropy inverse-RL and use a variational approximation to derive a scalable algorithm. In the second approach (Gangwani et al., 2021), we use the demonstrated expert states to produce a set of state-action pairs in the learner's environment, such that this generated dataset is convenient for imitation learning via distribution matching. Our third contribution in this part is an algorithm for imitation in partially observable environments, where the input observations to the agent contain imperfect information about the world (Gangwani et al., 2019). Our proposed approach combines concepts from belief representation learning and adversarial imitation learning.

1.1.3 Supervision via Diversity Pressure

Humans possess the capability to discover different ways to solve a given task due to the innate diversity in their behavioral preferences. In RL environments, the reward function provided by the task-designer may permit more than one optimal policy. In such cases, it is useful to learn a diverse collection of behaviors that achieve high environmental returns. Some of the benefits include efficient exploration in large state spaces with sparse or deceptive rewards (Conti et al., 2017; Hong et al., 2018), learning on downstream tasks via skill-composition (Florensa et al., 2017), and learning of robust behaviors that transfer across environments (Cully et al., 2015). A suitable strategy to uncover diverse behaviors is to train an ensemble of RL agents with an explicit loss function that encourages diversity.

We contribute an algorithm for learning a population of agents where each member is optimized to simultaneously accumulate high task-returns and exhibit behavioral diversity vis-à-vis other members (Gangwani et al., 2020). We build on a recent kernel-based method for training such an ensemble with Stein variational gradient descent (Liu et al., 2017).

1.1.4 Supervision via Knowledge Sharing

An RL agent need not learn a new task in isolation, from scratch. Its learning process could be supplemented with knowledge from other agents' learning. Information sharing among agents could be facilitated by techniques like transfer-RL and meta-RL. The transfer-RL method typically assumes the presence of a teacher policy network that distills useful knowledge into a student policy network (Rusu et al., 2015; Parisotto et al., 2015). In meta-RL, the agent is trained on a distribution over tasks, with the goal of learning to learn on a new task from the same distribution. Training with several tasks presents the opportunity

to share data among the tasks, thereby improving the overall sample-efficiency.

Our first contribution in this part is an algorithm for transfer-RL when the teacher and the student environments can have arbitrarily different state- and action-spaces (Wan et al., 2020). To handle this mismatch, we learn embeddings that can systematically extract knowledge from the teacher policy and value networks, and blend it into the student networks. Our second contribution is a trajectory-relabeling method for meta-RL that enables data sharing between tasks during the meta-training phase (Wan et al., 2021). It is inspired by a similar approach for the multi-task setting (Eysenbach et al., 2020). We find that sharing the trajectory data helps with exploration and is, therefore, especially beneficial for meta-RL on tasks with sparse rewards.

1.2 THESIS ORGANIZATION

This thesis is organized in the following three parts:

Part I: Supervision via Surrogate Rewards consists of two chapters. Chapters 2 and 3 focus on learning surrogate rewards using self-imitation and uniform return redistribution, respectively.

Part II: Supervision via Imitation Learning consists of two chapters. Chapters 4 and 5 propose algorithms for imitation learning in partially-observable environments and under model misspecification, respectively.

Part III: Supervision via Diversity Pressure and Knowledge Sharing consists of three chapters. Chapter 6 considers learning an ensemble of agents with behavioral diversity. Chapter 7 analyzes a transfer-RL approach across environments with different state- and action-spaces. Chapter 8 includes a method for sharing the trajectory data among the different tasks in the meta-RL paradigm.

Authorship Remarks. The content included in this thesis has either been published in conference proceedings, or is under submission at the time of writing. Chapters 2–6 are based on works for which this author is the primary contributor. Chapters 7–8 comprise of works for which Michael Wan is the first author, while this author is the second (Chapter 7) and the last (Chapter 8) author on the list.

Excluded Research. The material from Gangwani & Peng (2017) and Gangwani & Peng (2020) is not included in this thesis but was completed during the doctoral program.

CHAPTER 2: SURROGATE REWARDS WITH SELF-IMITATION LEARNING

2.1 INTRODUCTION

The success of popular algorithms for deep RL, such as policy-gradients and Q-learning, relies heavily on the availability of informative environmental rewards at each timestep of the sequential decision-making process. When rewards are only sparsely available during an episode, or rewarding feedback is provided only at episode termination, these algorithms tend to perform sub-optimally. This observation is linked to one of the fundamental challenges in RL known as temporal credit assignment. The lack of dense supervision results in inaccurate value estimation (Q or V function), which in turn makes credit assignment difficult. Several real-world decision-making problem are of the form where rewards are either only sparsely available during an episode, or the rewards are episodic, meaning that the reward feedback is delayed till the end of the episode of interaction (Rahmandad et al., 2009; Hein et al., 2017; Olivecrona et al., 2017). Delayed rewards could also be non-Markovian (Bacchus et al., 1996), which further complicates the design of RL algorithms and architectures.

One class of policy search algorithms, which are particularly handy when rewards are delayed, is black-box stochastic-optimization; examples include CEM (Rubinstein & Kroese, 2016), ES (Salimans et al., 2017) and Deep-Neuroevolution (Such et al., 2017). They are invariant to delayed rewards since the trajectories are not decomposed into individual timesteps for learning; rather zeroth-order finite-difference or gradient-free methods are used to learn policies based *only* on the cumulative rewards of the entire trajectory. However, a major disadvantage is that discarding the temporal structure of the RL problem leads to inferior sample-efficiency when compared with policy-gradients.

We propose to improve credit-assignment by learning dense surrogate Markovian rewards that are correlated with the environmental rewards. We observe that computing the policy-gradient using the surrogate rewards exhibits much better sample efficiency and asymptotic performance compared to the baseline approaches in tasks with sparse and delayed rewards. Basing off the equivalence between the policy function and its occupancy measure (Puterman, 1994), we formulate policy optimization as a divergence minimization problem. We show that with the Jensen-Shannon divergence, this reduces to a policy-gradient algorithm with shaped surrogate rewards that are learned with a binary classification loss, given a replay buffer of high-return trajectories. This algorithm could be interpreted as *self-imitation learning*, since the replay trajectories are self-generated by the agent during the course of learning, rather than being provided by an external source. Our experiments are performed

on the continuous-control MuJoCo locomotion tasks with delayed rewards. Lastly, we discuss the shortcomings of the self-imitation algorithm and directions to mitigate them.

2.2 POLICY OPTIMIZATION WITH SELF-IMITATION

Our RL environment is modeled as an infinite-horizon, discrete-time Markov Decision Process (MDP), where the objective is to learn a policy $\pi_\theta(a|s)$ such that the expected discounted sum of rewards, $\eta(\pi_\theta) = \mathbb{E}_{p_0,p,\pi}\big[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\big]$, is maximized. Here, $p(s_{t+1}|s_t,a_t)$ is the transition dynamics, $p_0(s_0)$ is the initial state distribution, and $\gamma \in [0,1)$ is the discount factor. Although the reward function $r(s,a)$ of the MDP is Markovian, in the scenario of delayed rewards, the rewards perceived by the agent could be non-Markovian. The state-action value function of π measures the expected return obtained by following π from a given state and action, $Q(s_t,a_t;\pi) = \mathbb{E}_{p_0,p,\pi}\big[\sum_{t'=t}^{\infty}\gamma^{t'-t} r(s_{t'},a_{t'})\big]$. We define the unnormalized γ-discounted state-visitation distribution for a policy π by $\rho_\pi(s) = \sum_{t=0}^{\infty}\gamma^t P(s_t = s|\pi)$, where $P(s_t = s|\pi)$ is the probability of being in state s at time t, when following policy π starting from a state sampled from p_0. The expected policy return $\eta(\pi_\theta)$ can then be written as $\mathbb{E}_{\rho_\pi(s,a)}[r(s,a)]$, where $\rho_\pi(s,a) = \rho_\pi(s)\pi(a|s)$ is the state-action visitation distribution. The policy gradient theorem (Sutton et al., 2000) provides the gradient for updating the policy parameters, $\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\rho_\pi(s,a)}\big[\nabla_\theta \log \pi_\theta(a|s) Q(s,a;\pi)\big]$.

Although the policy $\pi(a|s)$ is parameterized as a conditional distribution, an equivalent characterization of its behavior is provided by the corresponding state-action visitation distribution $\rho_\pi(s,a)$, which wraps the MDP dynamics and precisely defines the expected return of the policy in the MDP via $\eta(\pi) = \mathbb{E}_{\rho_\pi}[r(s,a)]$ (Puterman, 1994). Therefore, distance metrics on a policy π could be defined with respect to the visitation distribution ρ_π, and policy search could be viewed as finding a visitation distribution ρ_π that yields a high expected return. Suppose we have access to the visitation distribution ρ_{π^*} corresponding to an expert policy π^*. It is natural to consider learning a policy π such that its visitation ρ_π matches ρ_{π^*}. To do so, we could define a probability divergence measure $D(\rho_\pi, \rho_{\pi^*})$ that captures the similarity between two distributions, and minimize this divergence for policy improvement, i.e., $\boldsymbol{\min_\pi D(\rho_\pi, \rho_{\pi^*})}$. Ho & Ermon (2016) popularized this idea for imitation learning from external demonstrations. In this work, however, we do not assume access to any externally available expert policy or demonstration data. Rather, we propose to maintain a buffer $\mathcal{B}$ of high-return trajectories from the previous rollouts of the policy π, and optimize π to minimize the divergence between ρ_π and the empirical state-action pair distribution $\{(s_i,a_i)\}_\mathcal{B}$:

$$\min_\pi D(\rho_\pi, \rho_\mathcal{B}) \qquad ; \; \rho_\mathcal{B} \stackrel{\text{def}}{=} \{(s_i,a_i)\}_\mathcal{B} \tag{2.1}$$

Since the agent learns by considering its own past good (high-return) experiences as the *expert*, the algorithm can be viewed as *self-imitation* learning from the replay buffer $\mathcal{B}$. The buffer stores the top-k trajectories generated thus far during the training process, as measured by the trajectory return. It is modeled with a priority queue of fixed capacity, and the priority of a trajectory is its return value. Using the trajectory return (instead of the per-timestep reward) makes the approach invariant to the reward delays. Following prior literature (Goodfellow et al., 2014; Ho & Ermon, 2016), we use the Jensen-Shannon divergence as the distance measure D. It can be estimated (up to constant scaling and shift) with the variational form:

$$D_{JS}(\rho_\pi, \rho_\mathcal{B}) = \max_{d(s,a), d_\mathcal{B}(s,a)} \mathbb{E}_{\rho_\pi}[\log \frac{d(s,a)}{d(s,a) + d_\mathcal{B}(s,a)}] + \mathbb{E}_{\rho_\mathcal{B}}[\log \frac{d_\mathcal{B}(s,a)}{d(s,a) + d_\mathcal{B}(s,a)}] \tag{2.2}$$

where $d(s,a)$ and $d_\mathcal{B}(s,a)$ are learned functions. Let the policy π be parameterized with θ, and θ_{old} be the current policy iterate. At $\theta = \theta_{\text{old}}$, denote the *argmax* of Equation 2.2 with $d^*(s,a;\theta_{\text{old}}), d^*_\mathcal{B}(s,a;\theta_{\text{old}})$. The gradient of $D_{JS}(\rho_\pi, \rho_\mathcal{B})$ *w.r.t.* θ at the current iterate is obtained as:

$$\begin{aligned} &\nabla_\theta D_{JS}(\rho_\pi, \rho_\mathcal{B})\Big|_{\theta=\theta_{\text{old}}} = \mathbb{E}_{\rho_\pi(s,a)}\big[\nabla_\theta \log \pi_\theta(a|s) Q^d(s,a;\pi)\big], \\ &\text{where } Q^d(s_t,a_t;\pi) = \mathbb{E}_{\rho_\pi(s,a)}\big[\sum_{t'=t}^{\infty} \gamma^{t'-t} \underbrace{\log \frac{d^*(s_{t'},a_{t'};\theta_{\text{old}})}{d^*(s_{t'},a_{t'};\theta_{\text{old}}) + d^*_\mathcal{B}(s_{t'},a_{t'};\theta_{\text{old}})}}_{\text{(negative of) surrogate reward}}\big] \end{aligned} \tag{2.3}$$

2.2.1 Combining Surrogate and Environmental Rewards

The gradient in Equation 2.3 bears semblance to the standard policy-gradient (Sutton et al., 2000), but for replacing the environmental reward with a shaped per-timestep cost (negative reward, shown with an under-brace). This enables us to conveniently interpolate the gradient of D_{JS} with the policy-gradient from the environmental rewards. The combined gradient on the policy parameters is:

$$\nabla_\theta \eta(\pi_\theta) = (1-\nu)\mathbb{E}_{\rho_\pi(s,a)}\big[\nabla_\theta \log \pi_\theta(a|s) Q^r(s,a)\big] - \nu \nabla_\theta D_{JS}(\rho_\pi, \rho_\mathcal{B}), \tag{2.4}$$

where Q^r is the Q-function with the environmental rewards, and $\nu \in [0,1]$. Define $r^\phi(s,a) \stackrel{\text{def}}{=} d^*(s,a;\theta_{\text{old}})/(d^*(s,a;\theta_{\text{old}}) + d^*_\mathcal{B}(s,a;\theta_{\text{old}}))$. $r^\phi(s,a)$ could be approximated by using a parameterized network (ϕ) that is trained to solve the D_{JS} optimization (Equation 2.2) using the current policy rollouts and $\mathcal{B}$. Using Equation 2.3, the interpolated gradient could be

Algorithm 2.1:

θ (policy), ϕ (discriminator) $\sim$ initial parameters
$\mathcal{B} \leftarrow$ empty replay memory
for *each iteration* **do**
  Generate batch of trajectories $\{\tau\}_1^b$ with two rewards for each transition: $r_1 = r(s,a)$ from the environment, and $r_2 = -\log r^\phi(s,a)$
  Update $\mathcal{B}$ with $\{\tau\}_1^b$ (using trajectory return as the priority)
  /* Update policy θ */
  for *each minibatch* **do**
    Calculate $g_1 = \nabla_\theta \eta^{r_1}(\pi_\theta)$ with PPO using r_1 as reward
    Calculate $g_2 = \nabla_\theta \eta^{r_2}(\pi_\theta)$ with PPO using r_2 as reward
    Update θ with $(1-\nu)g_1 + \nu g_2$ using ADAM
  end
  /* Update self-imitation discriminator ϕ */
  for *each epoch* **do**
    $s_1 \leftarrow$ Sample mini-batch of (s,a) from $\mathcal{B}$
    $s_2 \leftarrow$ Sample mini-batch of (s,a) from $\{\tau\}_1^b$
    Update ϕ with log-loss objective using s_1, s_2 (Equation 2.2)
  end
end

rearranged as:

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\rho_\pi(s,a)}\Big[\nabla_\theta \log \pi_\theta(a|s)\big[\underbrace{(1-\nu)Q^r(s,a) + \nu Q^{r^\phi}(s,a)}_{\text{combined } Q}\big]\Big], \tag{2.5}$$

where $Q^{r^\phi}(s_t, a_t) = -\mathbb{E}_{p_0,p,\pi}\big[\sum_{t'=t}^{\infty} \gamma^{t'-t} \log r^\phi(s_{t'}, a_{t'})\big]$ is the Q-function calculated using $-\log r^\phi(s,a)$ as the surrogate reward. Intuitively, the surrogate reward is high for the state-action pairs that have a high density under the distribution of the buffer $\rho_\mathcal{B}$ (*i.e.,* expert-like pairs), and low otherwise. The combined Q-function in Equation 2.5 is, therefore, an interpolation between Q^r computed with the environmental rewards and Q^{r^ϕ} computed with the surrogate rewards that are implicitly shaped to guide the learner towards the behavior characterized by the (high-return) replay buffer. In environments with sparse or delayed rewards, where the signal from Q^r is weak or sub-optimal, a higher weight on Q^{r^ϕ} enables successful learning by self-imitation. We show this empirically in our experiments. We further observe that even in cases with dense environmental rewards, the two sources of reward signal can be readily combined. Algorithm 2.1 outlines the steps of our approach.

2.2.2 Limitations of Self-Imitation

As the replay buffer $\mathcal{B}$ contains a subset of the previously generated trajectories, the quality of the data in the buffer depends on the exploration proficiency of the agent. Inadequate exploration could lead to failure of self-imitation in a couple of ways. Consider a maze navigation task where the robot is only rewarded when it arrives at a goal $\mathcal{G}$ located in a far-off corner (zero reward for other timesteps). Unless the robot reaches $\mathcal{G}$ at least once, the trajectories in $\mathcal{B}$ always have a return of zero. There is no preference order among the zero-return trajectories and thus the learning signal from Q^{r^ϕ} is not useful. In a similar vein, self-imitation can lead to sub-optimal policies when there are local minima in the policy optimization landscape. For instance, assume the maze has a second goal $\mathcal{G}'$ in closer proximity to the starting point of the agent than $\mathcal{G}$, but reaching $\mathcal{G}'$ yields a lesser reward. With naïve exploration, the agent may fill $\mathcal{B}$ with below-par trajectories leading to $\mathcal{G}'$, and the reinforcement from Q^{r^ϕ} would drive it further to the deceptive target of $\mathcal{G}'$.

Stochasticity in the environment may also make it difficult to recover the optimal policy simply by imitating the past top-k rollouts. Consider a 2-armed bandit problem with reward distributions: Bernoulli$-p$ and Bernoulli$-(p+\epsilon)$, with a small ϵ. During training, rollouts from both the arms get conflated in $\mathcal{B}$ with a high probability, making it difficult to robustly (self-)imitate the action of picking the arm with the higher expected reward. One direction to mitigate some of the aforementioned shortcomings is to maintain sufficient *diversity* in the replay buffer and have an *ensemble* of agents that imitate all the different behaviors of interest, among those that exist in the buffer. We expand on this idea further in Chapter 6.

2.3 EXPERIMENTS

In this section, we quantitatively compare self-imitation with the standard policy-gradient with environmental rewards, under different types of reward distributions. We benchmark high-dimensional, continuous-control locomotion tasks based on the MuJoCo physics simulator (Todorov et al., 2012), by extending the OpenAI Baselines (Dhariwal et al., 2017) framework. The policy is modeled as a multivariate Gaussian distribution with standard deviation parameterized by a state-independent vector. We use clipped-ratio PPO algorithm (Schulman et al., 2017) as our base RL algorithm.

2.3.1 Self-Imitation with Different Reward Distributions

We consider the locomotion tasks in OpenAI Gym under 3 separate reward distributions: ***Dense*** refers to the default reward function in Gym, which provides a reward for each

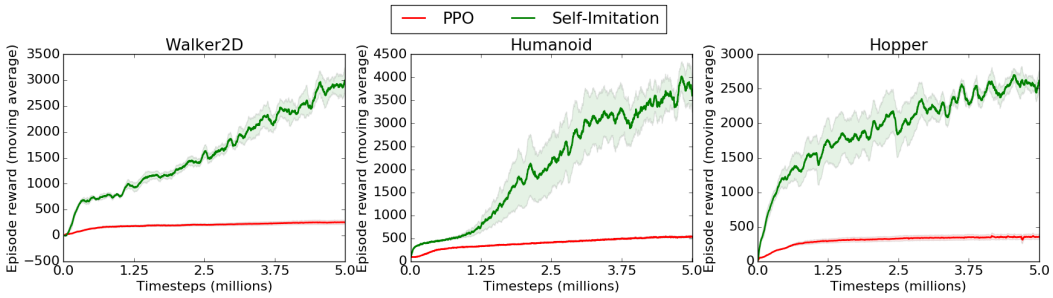


Figure 2.1: Learning curves for PPO and Self-Imitation on tasks with episodic rewards. Mean and standard-deviation over 5 random seeds is plotted.

simulation timestep. In ***episodic*** reward setting, rather than providing $r(s_t, a_t)$ at each timestep of an episode, we provide $\sum_t r(s_t, a_t)$ at the *last* timestep of the episode, and zero reward at other timesteps. In the ***noisy*** reward setting, we probabilistically mask out each out each per-timestep reward $r(s_t, a_t)$ in an episode. Reward masking is done independently for every new episode, and therefore, the agent receives non-zero feedback at different, albeit only few, timesteps in different episodes. The probability of masking-out or suppressing the rewards is denoted by p_m.

In Figure 2.1, we plot the learning curves on three locomotion tasks (Walker2d, Humanoid, Hopper) with episodic rewards. Recall that ν is the hyper-parameter controlling the weight distribution between gradients with environment rewards and the gradients with the surrogate rewards r^ϕ (Equation 2.5). The baseline PPO agents use $\nu = 0$, meaning that the entire learning signal comes from the environmental rewards. We compare them with the self-imitating (SI) agents that use a constant value $\nu = 0.8$. The capacity of the buffer $\mathcal{B}$ is fixed at 10 trajectories. We didn't observe our method to be particularly sensitive to the choice of ν and the capacity value; ablations are included in §2.3.2.

In Figure 2.1, we see that the PPO agents are unable to make any tangible progress on these tasks with episodic rewards, possibly due to difficulty in credit assignment – the lumped rewards at the end of the episode can't be properly attributed to the individual state-action pairs during the episode. In case of Self-Imitation, the algorithm has access to the surrogate rewards for each timestep, derived from the high-return trajectories in $\mathcal{B}$. This makes credit-assignment easier, leading to successful learning even for very high-dimensional control tasks such as Humanoid.

Table 2.1 summarizes the final performance, averaged over 5 runs with random seeds, under the various reward settings. For the noisy rewards, we compare performance with two different reward masking values - suppressing each reward $r(s_t, a_t)$ with 90% probability ($p_m = 0.9$), and with 50% probability ($p_m = 0.5$). The density of rewards increases across

	Episodic rewards				Noisy rewards Each r_t suppressed w/ 90% prob. ($p_m = 0.9$)		Noisy rewards Each r_t suppressed w/ 50% prob. ($p_m = 0.5$)		Dense rewards (Gym default)	
	$\nu = 0.8$ (SI)	$\nu = 0$ (PPO)	CEM	ES	$\nu = 0.8$ (SI)	$\nu = 0$ (PPO)	$\nu = 0.8$ (SI)	$\nu = 0$ (PPO)	$\nu = 0.8$ (SI)	$\nu = 0$ (PPO)
Walker	2996	252	205	≈1200	2276	2047	3049	3364	3263	3401
Humanoid	3602	532	426	-	4136	1159	4296	3145	3339	4149
H-Standup ($\times 10^4$)	18.1	4.4	9.6	-	14.3	11.4	16.3	9.8	17.2	10
Hopper	2618	354	97	≈1900	2381	2264	2137	2132	2700	2252
Half-Cheetah	3686	-1572	-	≈3200	3378	1670	4574	2374	4878	2422
Swimmer	173	21	17	-	52	37	127	56	106	68
Invd.Pendulum	8668	344	86	≈9000	8744	8826	8926	8968	8989	8694
Inv. Pendulum	977	53	-	-	993	999	978	988	969	992

Table 2.1: Performance of PPO and Self-Imitation (SI) on tasks with episodic rewards, noisy rewards with masking probability p_m, and dense rewards. All runs use 5M timesteps of interaction with the environment. ES performance at 5M timesteps is taken from (Salimans et al., 2017). Missing entry denotes that we were unable to obtain the 5M timestep performance from the source.

the reward settings from left to right in Table 2.1. We find that SI agents ($\nu = 0.8$) achieve higher average score than the baseline PPO agents ($\nu = 0$) in majority of the tasks for all the settings. This indicates that not only does self-imitation vastly help when the environment rewards are scant, it can readily be incorporated with the standard policy gradients via interpolation, for successful learning across various reward settings. For completion, we include performance of CEM and ES as these algorithms depend only on trajectory-return and don't exploit the temporal structure. CEM perform poorly in most of the cases. ES, while being able to solve the tasks, is sample-inefficient. We include ES performance from Salimans et al. (2017) after 5M timesteps of training for a fair comparison with our algorithm.

2.3.2 Ablation Studies

We measure the sensitivity of self-imitation to ν and the buffer capacity, denoted by $|\mathcal{B}|$ on the Humanoid and the Hopper tasks with episodic rewards. Table 2.2 reports the average performance over 5 random seeds. For ablation on ν, $|\mathcal{B}|$ is fixed at 10; for ablation on $|\mathcal{B}|$, ν is fixed at 0.8. With episodic rewards, a higher value of ν helps boost performance since the RL signal from the environment is weak. With $\nu = 0.8$, there isn't a single best choice for $|\mathcal{B}|$, though all values of $|\mathcal{B}|$ give better results than baseline PPO ($\nu = 0$).

	Humanoid	Hopper		Humanoid	Hopper
$\nu = 0$	532	354	$\lvert\mathcal{B}\rvert = 1$	2861	1736
$\nu = 0.2$	395	481	$\lvert\mathcal{B}\rvert = 5$	2946	2415
$\nu = 0.5$	810	645	$\lvert\mathcal{B}\rvert = 10$	3602	2618
$\nu = 0.8$	3602	2618	$\lvert\mathcal{B}\rvert = 25$	2667	1624
$\nu = 1$	3891	2633	$\lvert\mathcal{B}\rvert = 50$	4159	2301

Table 2.2: Ablations on the parameter ν and the buffer capacity.

2.4 RELATED WORK AND CONCLUSION

In recent work, Oh et al. (2018) propose exploiting past good trajectories to drive exploration. Their algorithm buffers (s, a) and the corresponding return value for each transition in generated trajectories, and reuses the data for training if the stored return value is higher than the current state-value estimate. Our approach presents a different objective for self-imitation based on divergence-minimization. With this view, we learn dense surrogate rewards which are then used for policy optimization. Reusing high-return trajectories has also been explored for program synthesis and semantic parsing tasks (Liang et al., 2016, 2018; Abolafia et al., 2018). Our approach is also related to prior works on divergence minimization and imitation learning. We mention some of them below.

Divergence minimization for policy-search. Relative Entropy Policy Search (REPS) (Peters et al., 2010) restricts the loss of information between policy updates by constraining the KL-divergence between the state-action distribution of old and new policy. Policy search can also be formulated as an EM problem, leading to several interesting algorithms, such as RWR (Peters & Schaal, 2007) and PoWER (Kober & Peters, 2009). Here the M-step minimizes a KL-divergence between trajectory distributions, leading to an update rule which resembles return-weighted imitation learning. Please refer to Deisenroth et al. (2013) for a comprehensive exposition. MATL (Wulfmeier et al., 2017) uses adversarial training to bring state occupancy from a real and simulated agent close to each other for efficient transfer learning. In Guided Policy Search (GPS, Levine & Koltun (2013)), a parameterized policy is trained by constraining the divergence between the current policy and a controller learnt via trajectory optimization.

Learning from Demonstrations (LfD). The objective in LfD, or imitation learning, is to train a control policy to produce a trajectory distribution similar to the demonstrator.

Approaches for self-driving cars (Bojarski et al., 2016) and drone manipulation (Ross et al., 2013) have used human-expert data, along with Behavioral Cloning algorithm to learn good control policies. Deep Q-learning has been combined with human demonstrations to achieve performance gains in Atari (Hester et al., 2017) and robotics tasks (Večerík et al., 2017; Nair et al., 2017). Human data has also been used in the maximum entropy IRL framework to learn cost functions under which the demonstrations are optimal (Finn et al., 2016). Ho & Ermon (2016) use the same framework to derive an imitation-learning algorithm (GAIL) which is motivated by minimizing the divergence between agent's rollouts and external expert demonstrations. Besides humans, other sources of expert supervision include planning-based approaches such as iLQR (Levine et al., 2016) and MCTS (Silver et al., 2016). Our algorithm departs from prior work in forgoing external supervision, and instead using the past experiences of the learner itself as the demonstration data.

2.4.1 Conclusion

We approached policy optimization for deep-RL from the perspective of Jensen–Shannon divergence minimization between state-action distributions of a policy and its own past high-return rollouts. This leads to a self-imitation algorithm that is invariant to reward delays and thus improves upon standard RL from environmental rewards, especially when the rewards are episodic and noisy. The improvement is facilitated by a policy-gradient term that is computed using dense surrogate rewards. The surrogate rewards are parametric—they are obtained from a discriminator trained with the binary classification loss. We observe substantial performance gains over the baselines for high-dimensional, continuous-control tasks under different reward distributions. Lastly, we consider some limitations of our data-driven surrogate reward learning method and suggest using an ensemble of diverse agents to mitigate them (Chapter 6 has further details).

CHAPTER 3: SURROGATE REWARDS WITH UNIFORM REWARD REDISTRIBUTION

3.1 INTRODUCTION

Prevalent algorithms for deep-RL typically need to estimate the expected future rewards after taking an action in a particular state – Actor-critic and Q-learning involve computing the Q-value, while policy-gradient methods tend to be more stable when using the advantage function. The value estimation is performed using temporal difference (TD) or Monte-Carlo (MC) learning. Although deep-RL algorithms have achieved remarkable results on a wide array of tasks, their performance crucially depends on the meticulously designed reward function, which provides a dense per-timestep learning signal and facilitates value estimation. In real-world sequential decision-making problems, however, the rewards are often sparse or delayed. Examples include, to name a few, industrial process control (Hein et al., 2017), molecular design (Olivecrona et al., 2017), and resource allocation (Rahmandad et al., 2009). Delayed rewards introduce high bias in TD-learning and high variance in MC-learning (Arjona-Medina et al., 2019), leading to poor value estimates. This impedes long-term *temporal credit assignment* (Minsky, 1961; Sutton, 1984), which refers to the ability of the agent to attribute actions to consequences that may occur after a long time interval. Also, the delayed rewards could be non-Markovian (Bacchus et al., 1996), further complicating the design of RL algorithms and architectures. As a motivating example in a simulated domain, Figure 3.1 shows the performance with Soft-Actor-Critic (SAC) (Haarnoja et al., 2018), an off-policy RL method, on two MuJoCo locomotion tasks from the Gym suite. For *delay=k*, the agent receives no reward for $(k-1)$ timesteps and is then provided the accumulated rewards at the k^{th} timestep. Increasing the delay leads to progressively worse performance.

Policy search algorithms based on black-box stochastic optimization, such as Evolution Strategies (Salimans et al., 2017) and Deep-Neuroevolution (Such et al., 2017), are useful when rewards are delayed. They are invariant to delayed rewards since the trajectories are not decomposed into individual timesteps for learning; rather zeroth-order finite-difference or gradient-free methods are used to learn policies based only on the trajectory returns (or aggregated rewards). However, one downside is that discarding the temporal structure of the RL problem leads to inferior sample-efficiency when compared with the standard RL algorithms. Our goal in this work is to design an approach that easily integrates into the existing RL algorithms, thus enjoying the sample-efficiency benefits, while being invariant to delayed rewards. To achieve this, we introduce a modified RL objective that involves smoothing in the trajectory-space and arrive at a new algorithm for learning surrogate

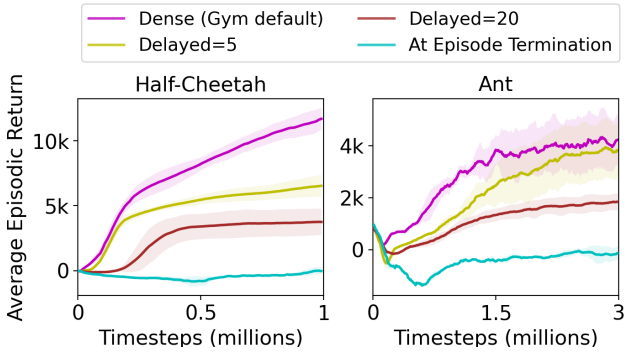


Figure 3.1: Effect of delayed rewards

rewards, which we refer to as *guidance* rewards. The guidance rewards are computed in a non-parametric way using only the trajectory-return values. They are Markovian and easy to infer for a given state-action tuple. The dense supervision from the guidance rewards makes value estimation and credit assignment easier, substantially accelerating learning when the original environmental rewards are sparse or delayed.

We provide an intuitive understanding for the guidance rewards in terms of *uniform* credit assignment – they characterize a uniform redistribution of the trajectory return to each constituent state-action pair. A favorable property of our approach is that no additional neural networks need to be trained to obtain the guidance rewards, unlike recent works that also consider the delayed rewards setting (*cf.* Section 3.6). For quantitative evaluation, we combine the guidance rewards with a variety of RL algorithms and environments. These include single-agent tasks: Q-learning (Watkins & Dayan, 1992) in a discrete grid-world and SAC on continuous control locomotion; and multi-agent tasks: TD3 (Fujimoto et al., 2018) and Distributional-RL (Bellemare et al., 2017) in multi-particle cooperative environments.

3.2 PRELIMINARIES

Our RL environment is modeled as an infinite-horizon, discrete-time Markov Decision Process (MDP). The MDP is characterized by the tuple $(\mathcal{S}, \mathcal{A}, r, p, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state- and action-space, respectively, and $\gamma \in [0, 1)$ is the discount factor. Given an action a_t, the next state is sampled from the transition dynamics distribution, $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$, and a reward $r(s_t, a_t)$ is generated using the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Although this reward function is Markovian for an MDP, in the delayed-reward setting, the rewards perceived by the agent could be non-Markovian. A stochastic policy $\pi(a_t|s_t)$ defines the state-conditioned distribution over actions. τ denotes a trajectory $\{s_0, a_0, s_1, a_1, \dots\}$ and

$R(\tau)$ is the sum of discounted rewards over the trajectory, $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$. The RL objective is to learn π that maximizes the expected $R(\tau)$, $\eta(\pi) = \mathbb{E}_{p,\pi}[R(\tau)]$.

Actor-critic Algorithms. These methods use a critic for value function estimation and an actor that is updated based on the information provided by the critic. The critic is trained with TD-learning in a policy-evaluation step; then the actor is updated with an approximate gradient in the direction of policy improvement. Under certain conditions, their repeated application converges to an optimal policy (Sutton & Barto, 2018). We briefly outline two model-free off-policy actor-critic RL algorithms that perform well on high-dimensional tasks and are used in this work – TD3 and SAC. TD3 is a deterministic policy gradient algorithm (DPG) (Silver et al., 2014). It uses a deterministic policy μ_θ that is updated with the policy gradient: $\nabla_\theta \mathbb{E}_{s\sim\rho^\beta}[Q^\mu(s, \mu_\theta(s))]$, where ρ^β is the state distribution of a behavioral policy β, and Q^μ is the state-action value trained with the Bellman error. TD3 alleviates the Q-function overestimation bias in DPG by using Clipped Double Q-learning when calculating the Bellman target. Differently, SAC optimizes for the maximum entropy RL objective, $\mathbb{E}_\pi[\sum_t \gamma^t (r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)))]$, where $\mathcal{H}$ and α are the policy entropy and the temperature, respectively. SAC alternates between soft policy evaluation, which estimates the soft Q-function using a modified Bellman operator, and soft policy improvement, which updates the actor by minimizing the Kullback-Leibler divergence between the policy distribution and exponential form of the soft Q-function. The loss functions for the critic (Q_ϕ), the actor (π_θ) and the temperature (α) are:

$$J_Q(\phi; r) = \mathbb{E}_{\substack{(s,a,s')\sim\mathcal{D} \\ a'\sim\pi_\theta(\cdot|s')}} \Big[\frac{1}{2}\big(Q_\phi(s,a) - (r(s,a) + \gamma(Q_{\bar{\phi}}(s',a') - \alpha \log \pi_\theta(a'|s')))\big)^2\Big] \tag{3.1}$$

$$J_\pi(\theta) = \mathbb{E}_{\substack{s\sim\mathcal{D} \\ a\sim\pi_\theta(\cdot|s)}} \big[\alpha \log(\pi_\theta(a|s)) - Q_\phi(s,a)\big]; \quad J(\alpha) = \mathbb{E}_{\substack{s\sim\mathcal{D} \\ a\sim\pi_\theta(\cdot|s)}} \big[-\alpha \log \pi_\theta(a|s) - \alpha\bar{\mathcal{H}}\big] \tag{3.2}$$

where $\mathcal{D}$ is the replay buffer, $Q_{\bar{\phi}}$ is the target critic, and $\bar{\mathcal{H}}$ is the expected target entropy.

3.3 GUIDANCE REWARDS: DEFINITION AND INTUITION

This section begins with the definition of our modified RL objective that involves smoothing in the trajectory-space, following which we make design choices that result in guidance rewards. Given a policy π_θ, the standard RL objective is: $\arg\max_\theta \mathbb{E}_{\tau\sim\pi(\theta)}[R(\tau)]$. As motivated before, with delayed environmental rewards, directly optimizing this objective hinders learning due to difficulty in temporal credit assignment caused by value estimation errors. Objective function smoothing has long been studied in the stochastic optimization literature.

In the context of RL, Salimans et al. (2017) proposed Evolution Strategies (ES) for policy search. ES creates a smoothed version of the standard RL objective using *parameter-level* smoothing (usually Gaussian blurring):

$$\eta^{\text{ES}}(\pi_\theta) = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\mathbb{E}_{\tau\sim\pi(\theta+\sigma\cdot\epsilon)}[R(\tau)] \tag{3.3}$$

where σ controls the level of smoothing. Although ES is invariant to delayed rewards, eschewing the temporal structure of the RL problem often results in low sample efficiency. Following the broad principle of using a smoothed objective to obtain effective gradient signals, we consider explicit smoothing in the *trajectory-space*, rather than the parameter-space. We define our maximization objective as:

$$\eta_{\text{smooth}}(\pi_\theta) = \mathbb{E}_{\hat{\tau}\sim\pi(\theta)}\big[\mathbb{E}_{\tau\sim M_{\hat{\tau}}}[R(\tau)]\big] \tag{3.4}$$

where $M_{\hat{\tau}}(\tau)$ is the smoothing distribution over trajectories τ that is parameterized by the reference trajectory $\hat{\tau}$. When M is a delta distribution, *i.e.*, $M_{\hat{\tau}}(\tau) = \delta(\tau = \hat{\tau})$, the original RL objective is recovered. We wish to design a smoothing distribution M that helps with credit assignment. Let $\beta(a|s)$ denote a behavioral policy and the trajectory distribution induced by β in the MDP be $p_\beta(\tau) = p(s_0)\prod_{t=0}^{\infty} p(s_{t+1}|s_t, a_t)\beta(a_t|s_t)$. Further, we introduce $p_\beta(\tau; s, a)$ as the distribution over the β-induced trajectories which *include* the state-action pair (s, a):

$$p_\beta(\tau; s, a) \stackrel{\text{def}}{=} \frac{p_\beta(\tau)\mathbb{1}[(s,a)\in\tau]}{\int_\tau p_\beta(\tau)\mathbb{1}[(s,a)\in\tau]\,\mathrm{d}\tau} \tag{3.5}$$

where $\mathbb{1}$ is the indicator function. For consistency, we require that the normalization constant be positive $\forall(s, a)$. Let $\{\hat{s}_t, \hat{a}_t\}$ be the reference state-action pairs in the reference trajectory $\hat{\tau}$. We propose the following infinite mixture model for the smoothing distribution $M_{\hat{\tau}}(\tau)$:

$$M_{\hat{\tau},\beta}(\tau) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t p_\beta(\tau; \hat{s}_t, \hat{a}_t) \tag{3.6}$$

Given a reference trajectory $\hat{\tau}$, this distribution samples trajectories from the behavioral policy β that *intersect* or overlap with the reference trajectory, with intersections at later timesteps discounted exponentially with the factor γ. Inserting this in Equation 3.4, rearranging and ignoring constants, the smoothed objective to maximize becomes:

$$\eta_{\text{smooth}}(\pi_\theta) = \mathbb{E}_{\hat{\tau}\sim\pi(\theta)}\big[\sum_{t=0}^{\infty}\gamma^t \underbrace{\int_\tau p_\beta(\tau; \hat{s}_t, \hat{a}_t)R(\tau)\,\mathrm{d}\tau}_{r_{\text{g}}(\hat{s}_t,\hat{a}_t)}\big] \tag{3.7}$$

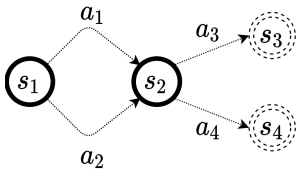


Figure 3.2: MDP with 4 states and 4 actions

This is equivalent to the standard RL objective, albeit with a different reward function than the environmental reward. We define this as the guidance reward function, $r_{\mathrm{g}}(s, a; \beta) = \mathbb{E}_{\tau \sim p_\beta(\tau; s, a)}[R(\tau)]$. The guidance reward apportioned to each state-action pair is the expected value (under p_β) of the returns of the trajectories which *include* that state-action pair. Useful features of r_{g} are that it provides a dense reward signal, and is invariant to delays in the environmental rewards since it depends on the trajectory return. Thus, it potentially promotes better value estimation and credit assignment.

3.3.1 Interpretation as Uniform Credit Assignment

Temporal Credit assignment deals with the question: *"given a final outcome (e.g. trajectory returns), how relevant was each state-action pair in that trajectory towards achieving the return?"*. Prior work has proposed learning estimators that explicitly model the relevance of an action to future returns, or using contribution analysis methods to redistribute rewards to the individual timesteps (*cf.* Section 3.6). Our method could be viewed as performing a simple redistribution – it *uniformly* distributes the trajectory return among the state-action pairs in that trajectory. [1] This non-committal or maximum entropy credit assignment is natural to consider in the absence of any prior structure or information. The guidance reward for each state-action pair is then obtained as the expected value (under p_β) of the uniform credit it receives from the different trajectory returns. For clarity of exposition, the next subsection demonstrates the guidance rewards using some elementary MDPs and p_β.

3.3.2 Illustration of Guidance Rewards with Simple MDP and p_β

Consider the MDP in Figure 3.2 with the states $\{s_1, s_2, s_3, s_4\}$, s_1 is the start state, $\{s_3, s_4\}$ are the terminal states. $\{a_1, a_2\}$ are the possible actions from s_1; $\{a_3, a_4\}$ are the possible

[1] In Equation 3.7, we excluded the constant $(1 - \gamma)$ from the smoothing distribution $M_{\hat{\tau}}(\tau)$ to reduce clutter. Since $1/(1 - \gamma)$ is the effective horizon, $(1 - \gamma)R(\tau)$ represents a uniform redistribution of the trajectory return to each constituent state-action pair.

actions from s_2. There are 4 possible trajectories. Let the return associated with each trajectory be the following:

- $\tau_1 : \{s_1 a_1 s_2 a_3\}; R(\tau_1) = 1$
- $\tau_2 : \{s_1 a_1 s_2 a_4\}; R(\tau_2) = 3$
- $\tau_3 : \{s_1 a_2 s_2 a_3\}; R(\tau_3) = 1$
- $\tau_4 : \{s_1 a_2 s_2 a_4\}; R(\tau_4) = 1$

The guidance reward is given by:

$$r_{\mathrm{g}}(s, a; \beta) = \mathbb{E}_{\tau \sim p_\beta(\tau; s, a)}[R(\tau)] \quad , \quad p_\beta(\tau; s, a) \stackrel{\text{def}}{=} \frac{p_\beta(\tau) \mathbb{1}[(s, a) \in \tau]}{\int_\tau p_\beta(\tau) \mathbb{1}[(s, a) \in \tau] \, \mathrm{d}\tau} \tag{3.8}$$

We compute the guidance rewards for the above MDP for two different p_β distributions - uniform and exponential.

With Uniform p_β

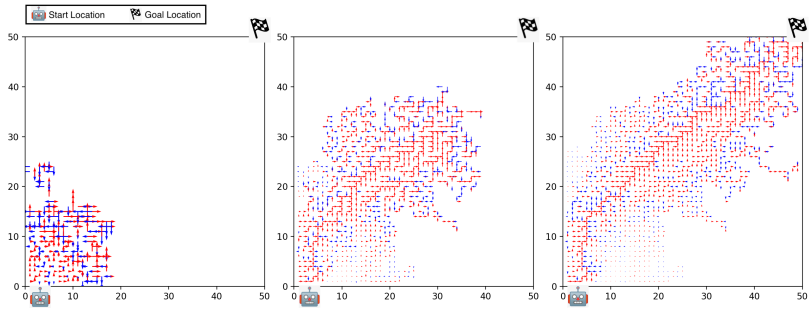
If p_β is uniform, $p_\beta(\tau_1) = p_\beta(\tau_2) = p_\beta(\tau_3) = p_\beta(\tau_4) = 0.25$. From this, we obtain:

- $p_\beta(\tau; s_1, a_1) = \frac{1}{2}\delta(\tau = \tau_1) + \frac{1}{2}\delta(\tau = \tau_2); \quad r_{\mathrm{g}}(s_1, a_1; \beta) = 2$
- $p_\beta(\tau; s_1, a_2) = \frac{1}{2}\delta(\tau = \tau_3) + \frac{1}{2}\delta(\tau = \tau_4); \quad r_{\mathrm{g}}(s_1, a_2; \beta) = 1$
- $p_\beta(\tau; s_2, a_3) = \frac{1}{2}\delta(\tau = \tau_1) + \frac{1}{2}\delta(\tau = \tau_3); \quad r_{\mathrm{g}}(s_2, a_3; \beta) = 1$
- $p_\beta(\tau; s_2, a_4) = \frac{1}{2}\delta(\tau = \tau_2) + \frac{1}{2}\delta(\tau = \tau_4); \quad r_{\mathrm{g}}(s_2, a_4; \beta) = 2$

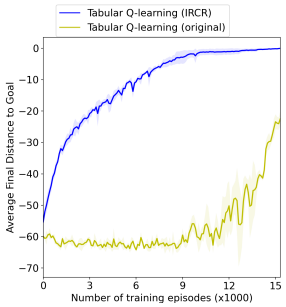
With Exponential p_β

If p_β is exponential, *i.e.* $p_\beta(\tau) \propto \exp(R(\tau))$, $p_\beta(\tau_1) = p_\beta(\tau_3) = p_\beta(\tau_4) = 0.1$; $p_\beta(\tau_2) = 0.7$ (rounded off to 1 decimal). From this, we obtain:

- $p_\beta(\tau; s_1, a_1) = \frac{1}{8}\delta(\tau = \tau_1) + \frac{7}{8}\delta(\tau = \tau_2); \quad r_{\mathrm{g}}(s_1, a_1; \beta) = 2.75$
- $p_\beta(\tau; s_1, a_2) = \frac{1}{2}\delta(\tau = \tau_3) + \frac{1}{2}\delta(\tau = \tau_4); \quad r_{\mathrm{g}}(s_1, a_2; \beta) = 1$
- $p_\beta(\tau; s_2, a_3) = \frac{1}{2}\delta(\tau = \tau_1) + \frac{1}{2}\delta(\tau = \tau_3); \quad r_{\mathrm{g}}(s_2, a_3; \beta) = 1$
- $p_\beta(\tau; s_2, a_4) = \frac{7}{8}\delta(\tau = \tau_2) + \frac{1}{8}\delta(\tau = \tau_4); \quad r_{\mathrm{g}}(s_2, a_4; \beta) = 2.75$



(a) Quiver plots during training



(b) Performance with guidance rewards

Figure 3.3: We consider a 50×50 grid with the start and goal locations marked in the image. The rewards are episodic – a non-zero reward is only provided at the end of the episode (horizon=150 steps) and is equal to the negative of the distance of the final position to the goal. We run for 15k episodes. The three quiver plots (left to right) are taken after 100 episodes, 2k episodes and 15k episodes, respectively. In each quiver plot, an arrow in a state represents the guidance reward: the direction denotes the action with *maximum* guidance reward, *i.e.,* $\arg\max_a r_{\mathrm{g}}(s,a)$, and the length denotes its magnitude in $[0,1]$. A state with no arrow mean that the guidance reward is 0 for all actions in that state. For ease of exposition, we have colored all arrows pointing up/right with red and all arrows pointing down/left with blue. We note that over time, reasonable guidance rewards emerge along the diagonal path from the start location to the goal. Although the guidance rewards in the top-left and bottom-right regions of the grid are imprecise, they are not critical for learning the optimal policy to achieve the task. Figure (right) compares tabular Q-learning with environmental rewards and our guidance rewards. *Quiver plots best viewed when digitally zoomed.*

3.3.3 Monte-Carlo Estimate of the Guidance Rewards

Without access to the true reward function of the MDP, it is infeasible to solve for the guidance rewards exactly. We resort to a Monte-Carlo (MC) estimation for the expectation, $r_{\mathrm{g}}(s,a;\beta) = \mathbb{E}_{\tau\sim p_\beta(\tau;s,a)}[R(\tau)]$. Let $\rho_\pi(s,a)$ denote the unnormalized discounted state-action visitation distribution for π. The smoothed RL objective (Equation 3.7) can be written as:

$$\eta_{\mathrm{smooth}}(\pi_\theta) = \mathbb{E}_{(s,a)\sim\rho_\pi}\mathbb{E}_{\tau\sim p_\beta(\tau;s,a)}[R(\tau)] \tag{3.9}$$

Plugging in the definition of $p_\beta(\tau;s,a)$ and using linearity of expectations:

$$\eta_{\mathrm{smooth}}(\pi_\theta) = \mathbb{E}_{\tau\sim p_\beta(\tau)}\mathbb{E}_{(s,a)\sim\rho_\pi}\left[\frac{R(\tau)\mathbb{1}[(s,a)\in\tau]}{\int_\tau p_\beta(\tau)\mathbb{1}[(s,a)\in\tau]\,\mathrm{d}\tau}\right] \tag{3.10}$$

Let $\mathbf{\Gamma}$ denote a set of N trajectories generated in the MDP using β, and $N(s,a)$ be the count of the trajectories in $\mathbf{\Gamma}$ which include the tuple (s,a). The MC estimate of $\eta_{\mathrm{smooth}}(\pi_\theta)$ is:

Algorithm 3.1: Tabular Q-learning with IRCR

Initialize Q(s,a) $\leftarrow 0$
$R_{\max} \leftarrow -\infty$; $R_{\min} \leftarrow \infty$ ▷ Maximum/Minimum return thus far
$\mathcal{B}(s,a) \leftarrow \emptyset \ \forall \ (s,a)$ ▷ Buffer that stores for each (s,a), a list of returns of trajectories that include that (s,a)

Function `GetGuidanceReward`(s,a)**:**
    /* Get normalized returns; return 0 if $\mathcal{B}(s,a) = \emptyset$ */
    return $\mathbb{E}_{R_i \sim \mathcal{B}(s,a)}[\frac{R_i - R_{\min}}{R_{\max} - R_{\min}}]$

for *each episode* **do**
    $R_e \leftarrow 0$ ▷ Accumulates rewards for current episode
    $\tau_e \leftarrow \emptyset$ ▷ Stores state-action pairs for current episode
    for *each step in* $\{1, \ldots, T\}$ **do**
        Choose $\mathbf{a}$ from $\mathbf{s}$ using policy derived from Q (ϵ-greedy)
        Take action $\mathbf{a}$ and observe $\mathbf{r}, \mathbf{s}'$ ▷ Sample transition from the environment
        $\tau_e \leftarrow \tau_e \cup \{(\mathbf{s}, \mathbf{a})\}$; $R_e \leftarrow R_e + \mathbf{r}$
        $\mathbf{r}_g(\mathbf{s}, \mathbf{a}) \leftarrow$ GetGuidanceReward$(\mathbf{s}, \mathbf{a})$
        $Q(\mathbf{s}, \mathbf{a}) \leftarrow Q(\mathbf{s}, \mathbf{a}) + \alpha \ [\mathbf{r}_g(\mathbf{s}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}') - Q(\mathbf{s}, \mathbf{a})]$
    end
    for *each* (s,a) *in* τ_e **do**
        $\mathcal{B}(s,a) \leftarrow \mathcal{B}(s,a) \cup \{R_e\}$ ▷ Update $\mathcal{B}$ for (s,a) along the collected trajectory
    end
    $R_{\max} \leftarrow \max(R_{\max}, R_e)$; $R_{\min} \leftarrow \min(R_{\min}, R_e)$ ▷ Update $R_{\max}, R_{\min}$
end

$$\hat{\eta}_{\text{smooth}}(\pi_\theta) = \frac{1}{N} \sum_{\tau \in \mathbf{\Gamma}} \mathbb{E}_{(s,a)\sim\rho_\pi} \left[\frac{R(\tau) \mathbb{1}[(s,a) \in \tau]}{N(s,a)/N} \right] = \mathbb{E}_{(s,a)\sim\rho_\pi} \underbrace{\sum_{\tau \in \mathbf{\Gamma}} \left[\frac{R(\tau) \mathbb{1}[(s,a) \in \tau]}{N(s,a)} \right]}_{r_g(s,a)} \tag{3.11}$$

The MC estimate of the guidance rewards is: $r_g(s,a) \triangleq (1/N(s,a)) \sum_{\tau \in \mathbf{\Gamma}} \left[R(\tau) \mathbb{1}[(s,a) \in \tau] \right]$. We further define $r_g(s,a) = 0$ if $N(s,a) = 0$.

3.4 INTEGRATING GUIDANCE REWARDS WITH DEEP-RL

It is possible to deploy an exploratory behavioral policy β to obtain the trajectory set $\mathbf{\Gamma}$ in a pre-training phase. Following that, the *stationary* guidance rewards computed from $\mathbf{\Gamma}$ could be used to learn a new policy with any of the standard RL methods. One issue with this sequential approach is that it is challenging to design β such that it achieves adequate

state-action-space coverage in high dimensions. Perhaps more importantly, it is typically unnecessary to have good estimates for the guidance rewards for the *entire* state-action-space. For instance, in a grid-world, if the goal location is always to the right of the starting position of the agent, it is acceptable to have an imprecise guidance reward in the left half of the grid, as long as the agent is discouraged from venturing to the left. Therefore, we propose an iterative approach where the experience gathered thus far by the agent is used to build the guidance rewards, *i.e.*, $r_{\mathrm{g}}(s, a)$ is the expected value of the uniform credit received by (s, a) from the trajectories already rolled out in the MDP. Simultaneously, a policy π (or Q-function) is learned using these *non-stationary* rewards. With this procedure, β could be thought of as being implicitly defined as a mixture of current and past policies π. The scale of the credit apportioned to a state-action pair from a trajectory depends on the scale of its return value. For the guidance rewards to be effective, it is sufficient that the *relative* values of the rewards be properly aligned to solve the task. Therefore, when assigning credits, we normalize the trajectory returns to the range $[0, 1]$ using min-max normalization. We refer to our approach for producing guidance rewards as **Iterative Relative Credit Refinement (IRCR)**. The guidance rewards are adapted over time as the average credit assigned to each state-action pair is refined by the information (return value) from newly sampled trajectories.

Any standard RL algorithm could be modified by replacing the environmental rewards with the guidance rewards. In Algorithm 3.1, we outline this for tabular Q-learning with small state and action spaces. The notable change from the standard Q-learning is the use of r_{g} in Line 14, instead of the environmental reward. To compute r_{g}, we maintain a buffer $\mathcal{B}(s, a)$ for each state-action pair that stores the returns of the past trajectories which include that state-action pair (Line 17). The guidance rewards evolve over time since the average credit allotted to a state-action pair changes as more experience is gathered in the MDP. To illustrate this, we run Algorithm 3.1 in a 50×50 grid-world with episodic environmental rewards. Figure 3.3a provides some insights on the structure of the guidance rewards assigned to the different regions of the grid as training progresses; the description of the episodic rewards and the arrows in the quiver plots is provided in the figure caption. In Figure 3.3b, we show the performance gains compared with tabular Q-learning using environmental rewards.

3.4.1 Scaling to High-dimensional Continuous Spaces

Actor-critic algorithms that scale to more complex environments (*e.g.* TD3, SAC) maintain an experience replay buffer (Lin, 1992) that stores $\{s, a, s', r\}$ tuples. These algorithms can be readily tailored to use guidance rewards. In Algorithm 3.2, we summarize SAC with

Algorithm 3.2: Soft Actor-Critic with IRCR

1 Initialize ϕ, $\bar{\phi}$, θ ▷ Policy and critic parameters
2 $R_{\max} \leftarrow -\infty$; $R_{\min} \leftarrow \infty$ ▷ Maximum/Minimum return thus far
3 $\mathcal{D} \leftarrow \emptyset$ ▷ Empty replay buffer
4 **for** *each episode* **do**
5 $\quad R_e \leftarrow 0$ ▷ Accumulates rewards for current episode
6 $\quad \mathcal{D}_e \leftarrow \emptyset$ ▷ Stores transitions for current episode
7 $\quad$ **for** *each step in* $\{1, \ldots, T\}$ **do**
8 $\quad\quad \mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$
9 $\quad\quad$ Take action $\mathbf{a}$ and observe $\mathbf{r}, \mathbf{s}'$ ▷ Sample transition from the environment
10 $\quad\quad \mathcal{D}_e \leftarrow \mathcal{D}_e \cup \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')\}$; $R_e \leftarrow R_e + \mathbf{r}$
11 $\quad$ **end**
12 $\quad$ **for** *each* $(s, a, s') \in \mathcal{D}_e$ **do**
13 $\quad\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{(s, a, s', R_e)\}$ ▷ Append each transition with R_e and add to replay buffer
14 $\quad$ **end**
15 $\quad R_{\max} \leftarrow \max(R_{\max}, R_e)$; $R_{\min} \leftarrow \min(R_{\min}, R_e)$ ▷ Update $R_{\max}, R_{\min}$
16 $\quad$ **for** *each gradient step* **do**
17 $\quad\quad \{s^{(k)}, a^{(k)}, s'^{(k)}, R^{(k)}\}_{k \in \mathbb{N}^+} \sim \mathcal{D}$ ▷ Sample batch
18 $\quad\quad \mathbf{r_g^{(k)}} \leftarrow \frac{R^{(k)} - R_{\min}}{R_{\max} - R_{\min}}$ ▷ Get guidance reward by normalizing return
19 $\quad\quad \phi \leftarrow \phi - \lambda \nabla_\phi J_Q(\phi; \mathbf{r_g})$ ▷ Update Q-function using guidance rewards, (Equation 3.1)
20 $\quad\quad \theta \leftarrow \theta - \lambda \nabla_\theta J_\pi(\theta)$; $\alpha \leftarrow \alpha - \lambda \nabla_\alpha J(\alpha)$ ▷ Update policy and temperature, (Equation 3.2)
21 $\quad$ **end**
22 **end**

IRCR. The environmental reward in the experience replay tuple is replaced with the return of the trajectory which produced that tuple (Line 13). When computing the soft Bellman error for learning the soft Q-function, the guidance reward is calculated by normalizing this return value (Lines 18-19). Mathematically, this is equivalent to the MC estimation of the guidance reward using a single trajectory, rather than the expected credit from a trajectory distribution. This is not an issue in practice if the soft Q-function, which is learned with these guidance rewards, is parameterized by a deep neural network that tends to generalize well in the vicinity of the input data. Indeed, as our experiments will show, Algorithm 3.2 achieves reliable performance in high-dimensional tasks. Other actor-critic algorithms could be modified analogously to incorporate the guidance rewards.

3.4.2 Discussion on Convergence

Some comments are in order concerning the convergence of our iterative approach. We provide a qualitative analysis by drawing an analogy with the Cross Entropy (CE) method (Ru-

binstein & Kroese, 2013; Mannor et al., 2003). For policy search, CE uses a multivariate Gaussian distribution to represent a population of policies. In each iteration, individuals π_k are drawn from this distribution, their fitness, $\mathbb{E}_{\tau\sim\pi_k}[R(\tau)]$, is evaluated, and a fixed number of fittest individuals determine the new mean and variance of the population. This fitness-based selection ensures steady policy improvement. In IRCR, the trajectories generated by a mixture of the current and past policies ($\pi_{0:i}$) are used to compute the guidance rewards (r_{g}); π_{i+1} is then obtained by a policy optimization step with these rewards. Since r_{g} is positively correlated with the environmental returns $R(\tau)$, maximizing for a discounted sum of r_{g} tends to seek out a policy that attains higher $R(\tau)$ compared to $\pi_{0:i}$, on average. Consequently, this optimization step facilitates policy improvement in the same spirit as the CE method. The next section provides empirical evidence that the policy behavior improves over iterations of IRCR. We consider the theoretical study of convergence as an important future work.

3.4.3 Limitations of IRCR and the Role of Exploration

Exploration and credit assignment are two distinct fundamental problems in RL. The former deals with the *discovery* of new useful information, the latter is about efficiently incorporating this information for learning a robust policy. In hard exploration problems, an agent typically obtains zero rewards in each episode unless an exploration impetus is given, whereas in our setting, a reward signal is readily provided to the agent at the end of *every* episode. The focus of this work, therefore, is to train effectively from this delayed feedback and improve upon the credit assignment.

Since the reward that the agent optimizes for is different from the original task reward and is coupled to a behavioral policy β, for a given MDP, it might be possible to design an adversarial β such that optimizing for the resultant guidance rewards leads to unintended behaviors (as per the task rewards). Although not visible in our empirical evaluation, a limitation of IRCR is that a careful adaptation of β could be crucial in some domains to avoid this. For instance, if β gets stuck in some region of the state-action-space, the learning agent may also get trapped in a local optimum due to *deceptive* guidance rewards. Combining IRCR with methods that explicitly incentivize exploration is a promising approach.

3.5 EXPERIMENTS

This section evaluates our approach on various single-agent and multi-agent RL tasks to quantify the benefits of using the guidance rewards in place of the environmental rewards,

when the latter are sparse or delayed.

3.5.1 Single-agent Environments and Baselines

We benchmark high-dimensional, continuous-control locomotion tasks based on the MuJoCo physics simulator, provided in OpenAI Gym (Brockman et al., 2016). We compare SAC (IRCR) outlined in Algorithm 3.2 with the following baselines:

- SAC with environmental rewards. It uses the same hyperparameters as SAC (IRCR).
- Generative Adversarial Self-imitation Learning (*GASIL*), which represents the method proposed in Guo et al. (2018); Gangwani et al. (2018). A buffer stores the top-k trajectories according to the return. A discriminator network, which is a binary classifier that distinguishes the buffer data from data generated by the current policy, acts as a source of the guidance rewards.
- *Reward Regression*, which typifies the approaches presented in Arjona-Medina et al. (2019); Liu et al. (2019). They formulate a regression task that predicts the return given the entire trajectory. A network trained with this regression loss helps to decompose the trajectory return back to the constituent state-action pairs, and provides the guidance rewards. We include the results from Liu et al. (2019) using the Transformer architecture (data obtained from authors).

We modify the reward function in Gym to design tasks with episodic rewards – the agent collects a zero reward for all timesteps *except* the last one, at which the accumulated original reward is given. With reference to Figure 3.1 in the Introduction, this expresses the maximum possible delay in rewards. Figure 3.4 plots the learning curves for all the algorithms with episodic rewards. We observe SAC (IRCR) to be the most sample-efficient across all tasks. SAC (original) is unable to learn any useful behavior since the value estimation errors due to delayed environmental rewards impede temporal credit assignment. GASIL and Reward-regression [2] are invariant to delayed rewards, but the learning is much more sluggish than our approach, possibly due to the instability in training the additional neural network for reward function estimation (discriminator for GASIL and Transformer for Reward-regression). In contrast, IRCR does not require any auxiliary networks and simply defines the guidance rewards as the expected value of the credit apportioned to (s, a) from past trajectories.

[2]We were unable to obtain the data for *InvertedDoublePendulum*, *Ant* and *Half-Cheetah* for Reward-regression from the authors of Liu et al. (2019) since they do not evaluate on these three tasks.

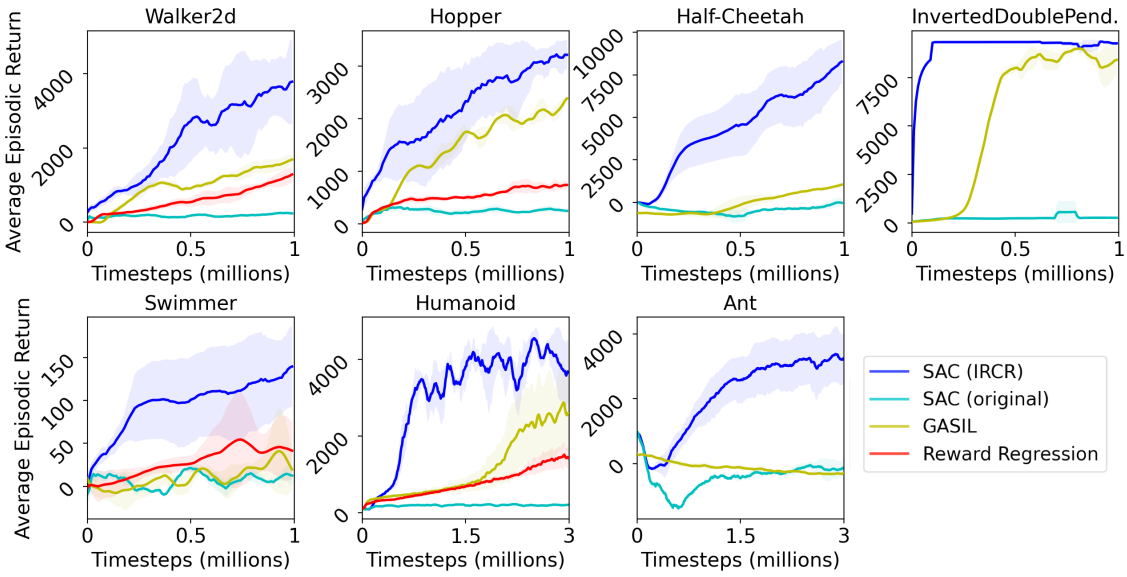
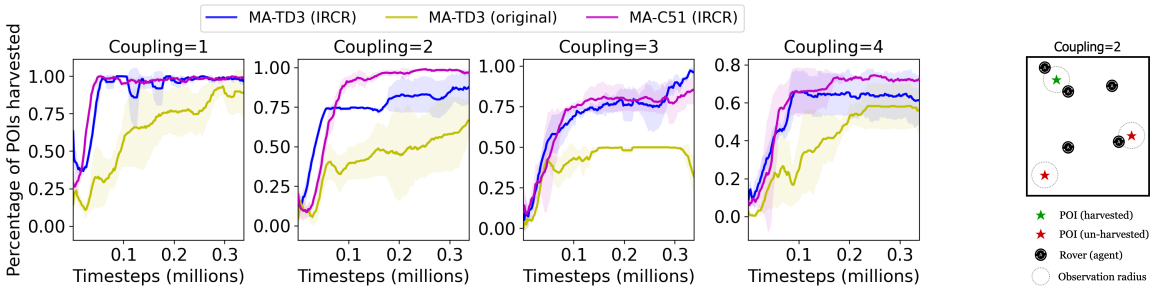


Figure 3.4: Learning curves for the MuJoCo locomotion tasks with **episodic rewards**. The mean and standard deviation over 5 random seeds are plotted. Reward-regression baseline has some missing data[2].

3.5.2 Multi-agent Environments and Baselines

We evaluate IRCR in a sparse-reward cooperative multi-agent RL (MARL) environment. This setting involves multiple agents that execute actions that jointly influence the environment; the agents receive local observations and a shared (sparse) reward. We adopt the *Rover Domain* from Rahmattalabi et al. (2016) in which agents navigate in a two-dimensional world with continuous states and actions. There are N rovers (agents) and K Points-of-interests (POIs). A global reward is achieved when any POI is *harvested*. For harvesting a POI, a certain minimum number of rovers—determined by a *Coupling* parameter—need to be simultaneously within a small observation radius around that POI; higher couplings require greater coordination. Figure 3.5b illustrates a scenario with coupling=2. Each rover has sensors to detect other rovers and POI around it using a mechanism similar to a LIDAR. A rover within the observation radius of an un-harvested POI receives a small local reward.

Our baseline algorithm is MA-TD3, a multi-agent extension of TD3 with the critic network shared among all agents. This is compared with two methods that employ guidance rewards – MA-TD3 (IRCR), which replaces the environmental rewards with the guidance rewards (similar to Algorithm 3.2), but uses the same underlying update rules; and MA-C51 (IRCR), which replaces the Clipped Double Q-learning in MA-TD3 with the C51 distributional-RL algorithm (Bellemare et al., 2017). Our C51 variant includes other minor alternations detailed in the next subsection. We experiment with different values for N, K, and the



(a) Learning curves (b) Rover Domain

Figure 3.5: (a) Learning curves for the Rover domain with different coupling factors. The mean and standard deviation over 5 random seeds are plotted.; (b) Rover Domain (illustrated with coupling=2).

coupling factor. Figure 3.5a plots, for coupling factors 1 to 4, the percentage of the POIs that are harvested at the end of an episode vs. the number of training timesteps. We note that MA-TD3 (IRCR) is more sample-efficient and achieves higher scores compared to MA-TD3 with environmental rewards, that are sparse since the agent collects a zero reward if it is outside the observation radius of every POI. Finally, the good performance of MA-C51 (IRCR) suggests that guidance rewards can be effectively used with distributional-RL also.

3.5.3 C51 Distributional-RL with Guidance Rewards

Background. Distributional-RL models the full distribution of the returns, the expectation of which is the Q-function. We use the C51 algorithm introduced by Bellemare et al. (2017) which represents the return distribution with learned probabilities on a fixed support; several other representation methods have also been proposed. Let $Z^\pi(s,a)$ be the random variable denoting the sum of discounted rewards along a trajectory starting with the state-action pair (s,a). The value function is $Q^\pi(s,a) = \mathbb{E}Z^\pi(s,a)$. $Z^\pi(s,a)$ is obtained by the repeated application of the distributional Bellman operator $\mathcal{T}^\pi$ defined as:

$$\mathcal{T}^\pi Z(s,a) \overset{D}{=} R(s,a) + \gamma Z(s',a') \quad s' \sim p(\cdot|s,a), a' \sim \pi(\cdot|s') \tag{3.12}$$

C51 models the value distribution with a discrete distribution on a fixed support $\{z_i\}_{i=1}^N$, referred to as a set of *atoms*. The atom probabilities are given by a learned parametric model $f_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^N$:

$$Z_\theta(s,a) = z_i \quad w.p. \quad p_\theta^i(s,a) = \text{softmax}(f_\theta(s,a))_i \tag{3.13}$$

Atom Support and Guidance Rewards in Log-space. In Bellemare et al. (2017), the support of the atoms ranges from $V_{\min}$ to $V_{\max}$, which are environment-specific variables defining the limits of the returns possible in that environment. To make the support range environment-agnostic, we define it in the log-space: $w_i = \{1/N, 2/N, \ldots, 1\}$ and $z_i = \log w_i$. Thus, the Q-function is written as $Q_\theta(s,a) = \sum_i p_\theta^i(s,a) \log w_i$.

We further define guidance rewards modified with a log function, $r_{\text{Lg}}(s,a) = \log r_{\text{g}}(s,a)$. Recall that $r_{\text{g}} \in [0,1]$ due to the min-max normalization; hence the application of log is proper (*expect at 0 where a small ϵ should be added*). Although this transformation changes the magnitude, the relative ordering of the guidance rewards is preserved due to the monotonicity of the log. The parametric model f_θ is optimized with TD-learning. With the distributional Bellman equation, this is equivalent to a distribution matching problem. Given a training tuple $(s, a, r_{\text{Lg}}, s')$ from the replay buffer, the discrete target distribution is:

$$r_{\text{Lg}} + \gamma z_i \quad w.p. \quad p_\theta^i = \text{softmax}(f_\theta(s', a'))_i \tag{3.14}$$

Using the log-space atom support and definition of r_{Lg}, we can rewrite this as:

$$\log \left[r_{\text{g}}.{w_i}^\gamma \right] \quad w.p. \quad p_\theta^i = \text{softmax}(f_\theta(s', a'))_i \tag{3.15}$$

Similarly, the discrete source distribution is:

$$\log w_i \quad w.p. \quad p_\theta^i = \text{softmax}(f_\theta(s, a))_i \tag{3.16}$$

The source and the target distributions have a support interval (-∞, 0]. In principle, any f-divergence metric on them could be minimized. An alternative is to induce a transformation before the divergence minimization. This is justified by the following theorem from Qiao & Minematsu (2010): *"The f-divergence between two distributions is invariant under differentiable and invertible transformation"*. With an exponential transformation, we get the following distributions that are now shaped to have a support interval (0,1]:

$$\begin{aligned} r_{\text{g}}.{w_i}^\gamma \quad w.p. \quad p_\theta^i &= \text{softmax}(f_\theta(s', a'))_i \\ w_i \quad w.p. \quad p_\theta^i &= \text{softmax}(f_\theta(s, a))_i \end{aligned} \tag{3.17}$$

Following (Bellemare et al., 2017), we minimize the KL-divergence between these distributions using a *projection* step to account for the mismatch in the atom positions between the source and the target.

3.5.4 Robotic Manipulator Arm Environment

Figure 3.6 shows the robotic arm that models a 7 degree-of-freedom Sawyer robot, inspired by Chen et al. (2018). The task is to insert a cylindrical peg (held in the end-effector attached to the arm) into a hole some distance away on the table. A non-zero reward is provided only at the end of every episode and is equal to exponential of negative L_2 distance between the final position of the peg and the hole. In Table 3.1, we compare the final performance of the SAC algorithm with environmental and guidance rewards, and note that the latter is considerably better.

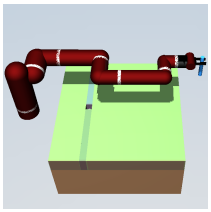


Figure 3.6: MuJoCo model of a 7 DoF arm based on the Sawyer robot.

SAC (env. rewards)	SAC (IRCR)	Random Policy
$104_{\pm 4}$	$160_{\pm 7}$	$90_{\pm 11}$

Table 3.1: SAC performance on the peg-insertion task with environmental and guidance rewards. Mean and standard deviation over 5 random seeds are reported.

3.6 RELATED WORK AND CONCLUSION

Reward shaping. Designing reward functions that modify or substitute the original rewards is a popular method for improving the learning efficiency of RL agents. Ng et al. (1999) developed potential-based shaping functions that guarantee the preservation of the optimal policy. Providing bonus rewards inspired by intrinsic motivation ideas such as curiosity (Schmidhuber, 1991, 1999) has been shown to aid exploration. The optimal reward problem (ORP) (Singh et al., 2009, 2010) introduces the idea that the original reward (that captures the *designer's* intent) could be decoupled from the rewards used by the agent in the RL algorithm (guidance rewards), even though the agent is finally evaluated on the designer's intent. Using the guidance rewards in place of the original rewards can substantially accelerate learning, especially if the agent is bounded by computational or knowledge constraints (Sorg et al., 2010, 2011). IRCR could be interpreted as an instance of ORP. It provides dense guidance rewards that improve value estimation when the original environmental rewards are sparse or delayed, thus enabling faster learning.

Credit Assignment. A variety of methods have been proposed to improve temporal credit assignment. Hindsight Credit Assignment (Harutyunyan et al., 2019) learns a model that quantifies the relevance of an action to a future outcome, such as the total return following

that action. An interesting feature of the model is that the Q-function estimate for *all* the actions could be improved using the returns sampled from a certain starting action. Temporal Value Transport (Hung et al., 2019) uses an attention-based memory module that links past events (at time t') to the present time t. The state-value at t is then "transported" to t', to be used as an additional bootstrap (or fictitious reward) in the TD error at t'. This helps in efficiently propagating credit backward in time. Neural Episodic Control (Pritzel et al., 2017) and Episodic Backward Update (Lee et al., 2019) enable faster propagation of sparse or delayed rewards from the entire episode through all the transitions of the episode. In RUDDER (Arjona-Medina et al., 2019), an LSTM network is trained to predict the trajectory return at every time-step. The guidance reward is then obtained as the difference of consecutive predictions. Liu et al. (2019) use a Transformer with masked multi-head self-attention as the learned reward function. It is trained by regressing on the trajectory-return and helps to decompose the return back to each time-step in the trajectory. Adversarial self-imitation approaches (Guo et al., 2018; Gangwani et al., 2018) use a min-max objective to train a discriminator and a policy iteratively. The discriminator is learned with a binary classification loss and provides guidance rewards for policy optimization. In contrast with these, our computation of the guidance rewards does not require training auxiliary networks and could be viewed as a simple uniform return decomposition.

3.6.1 Conclusion

In this work, we introduce a surrogate RL objective with smoothing in the trajectory-space. We show that our choice of the smoothing distribution makes this objective equivalent to standard RL, albeit with the guidance reward function instead of the environmental reward. The guidance rewards are easily measurable for any state-action pair as the expected return of the past trajectories which include that pair. The dense supervision afforded by them makes value estimation and temporal credit assignment easier. Our method is invariant to delayed rewards, does not require training auxiliary networks, and integrates well with existing RL algorithms. Experimental results across a variety of RL algorithms with single- and multi-agent tasks highlight the contribution of the guidance rewards in improving the sample-efficiency, especially when the environmental rewards are sparse or delayed.

CHAPTER 4: BELIEF REPRESENTATIONS FOR IMITATION LEARNING IN POMDPS

4.1 INTRODUCTION

Recent advances in RL have found successful applications in solving complex problems, including robotics, games, dialogue systems, and recommendation systems, among others. Despite such notable success, the application of RL is still quite limited to problems where the observation-space is rich in information and data generation is inexpensive. On the other hand, the environments in real-world problems, such as autonomous driving and robotics, are typically stochastic, complex and partially observable. To achieve robust and practical performance, RL algorithms should adapt to situations where the agent is being fed noisy and incomplete sensory information. To model these types of environments, partially observable Markov decision processes (POMDPs; Aström (1965)) have been proposed and widely studied. In a POMDP, since the current observation alone is insufficient for choosing optimal actions, the agent's history (its past observations and actions) is encoded into a *belief state*, which is defined as the distribution (representing the agent's beliefs) over the current latent state. Although belief states can be used to derive optimal policies (Kaelbling et al., 1998; Hauskrecht, 2000), maintaining and updating them requires knowledge of the transition and observation models of the POMDP, and is prohibitively expensive for high-dimensional spaces. To overcome this difficulty, several algorithms have been proposed that perform approximate inference of the belief state representation from raw observations, using recurrent neural networks (Guo et al., 2018), variational autoencoders (Igl et al., 2018; Gregor et al., 2018), and Predictive State Representations (Venkatraman et al., 2017). After the belief model has been learned, a policy optimization algorithm is then applied to the belief representation to optimize a predefined reward signal.

As an alternative to RL from predefined rewards, imitation learning often provides a fast and efficient way for training an agent to complete tasks. Expert demonstrations are provided to guide a learner agent to mimic the actions of the expert without the need to specify a reward function. A large volume of work has been done over the past decades on imitation learning for fully observable MDPs, including the seminal work on generative adversarial imitation learning (GAIL, Ho & Ermon (2016)), but there has been little focus on applying these ideas to partially observable environments.

In this work, we study the problem of imitation learning for POMDPs. Specifically, we introduce a new belief representation learning approach for generative adversarial imitation learning in POMDPs. Different from previous approaches, where the belief state represen-

tation and the policy are trained in a decoupled manner, we learn the belief module jointly with the policy, using a task-aware imitation loss which helps to align the belief representation with the policy's objective. To avoid potential belief degeneration, we introduce several informative belief regularization techniques, including auxiliary losses of predicting multi-step past/future observations and action-sequences, which improve the robustness of the belief representation. Evaluated on various partially observable continuous-control locomotion tasks built from MuJoCo, our belief-module imitation learning approach (BMIL) substantially outperforms several baselines, including the original GAIL algorithm and the task-agnostic belief learning algorithm. Extensive ablation analysis indicates the effectiveness of task-aware belief learning and belief regularization.

4.1.1 Background and Notations

RL in POMDPs. We consider the RL setting where the environment is modeled as a partially-observable Markov decision process (POMDP). A POMDP is characterized by the tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{O}$, $\mathcal{R}$, $\mathcal{T}$, $\mathcal{U}$, , $p(s_0)$, γ), where $\mathcal{S}$ is the state-space, $\mathcal{A}$ is the action-space, and $\mathcal{O}$ is the observation-space. The true environment states $s_t \in \mathcal{S}$ are latent or unobserved to the agent. Given an action $a_t \in \mathcal{A}$, the next state is governed by the transition dynamics $s_{t+1} \sim \mathcal{T}(s_{t+1}|s_t, a_t)$, an observation is generated as $o_{t+1} \sim \mathcal{U}(o_{t+1}|s_{t+1})$, and reward is computed as $r_t = \mathcal{R}(r_t|s_t, a_t)$. The RL objective involves maximization of the expected discounted sum of rewards, $\eta(\pi_\theta) = \mathbb{E}_{p_0,\mathcal{T},\pi}\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\big]$, where $\gamma \in [0, 1)$ is the discount factor, and $p(s_0)$ is the initial state distribution. The action-value function is $Q^\pi(s_t, a_t) = \mathbb{E}_{p_0,\mathcal{T},\pi}\big[\sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'})\big]$. We define the unnormalized γ-discounted state-visitation distribution for a policy π by $\rho_\pi(s) = \sum_{t=0}^{\infty} \gamma^t P(s_t{=}s|\pi)$, where $P(s_t{=}s|\pi)$ is the probability of being in state s at time t, when following policy π and starting state $s_0 \sim p_0$. The expected policy return $\eta(\pi_\theta)$ can then be written as $\mathbb{E}_{\rho_\pi(s,a)}[r(s, a)]$, where $\rho_\pi(s, a) = \rho_\pi(s)\pi(a|s)$ is the state-action visitation distribution (also referred to as the occupancy measure). For any policy π, there is a one-to-one correspondence between π and its occupancy measure (Puterman, 1994). Using the policy gradient theorem (Sutton et al., 2000), the gradient of the RL objective can be obtained as $\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\rho_\pi(s,a)}\big[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s, a)\big]$.

Imitation Learning. Learning in popular RL algorithms (such as policy-gradients and Q-learning) is sensitive to the quality of the reward function. In many practical scenarios, the rewards are either unavailable or extremely sparse, leading to difficulty in temporal credit assignment (Sutton, 1984). In the absence of explicit environmental rewards, a promising approach is to leverage *demonstrations* of the completed task by experts, and learn to imitate

their behavior. Behavioral cloning (BC; Pomerleau (1991)) poses imitation as a supervised-learning problem, and learns a policy by maximizing the likelihood of expert-actions in the states visited by the expert. The policies produced with BC are generally not very robust due to the issue of compounding errors; several approaches have been proposed to remedy this (Ross et al., 2011; Ross & Bagnell, 2014). Inverse Reinforcement Learning (IRL) presents a more principled approach to imitation by attempting to recover the cost function under which the expert demonstrations are optimal (Ng et al., 2000; Ziebart et al., 2008). Most IRL algorithms, however, are difficult to scale up computationally because they require solving an RL problem in their inner loop. Recently, Ho & Ermon (2016) proposed framing imitation learning as an occupancy-measure matching (or divergence minimization) problem. Their architecture (GAIL) forgoes learning the optimal cost function in order to achieve computational tractability and sample-efficiency (in terms of the number of expert demonstrations needed). In detail, if $\rho_\pi(s,a)$ and $\rho_E(s,a)$ represent the state-action visitation distributions of the policy and the expert, respectively, then minimizing the Jenson-Shanon divergence $\min_\pi D_{JS}[\rho_\pi(s,a) \,||\, \rho_E(s,a)]$ helps to recover a policy with a similar trajectory distribution as the expert. GAIL iteratively trains a policy (π_θ) and a discriminator (D_ω) to optimize the min-max objective:

$$\min_\theta \max_\omega \; \mathbb{E}_{(s,a)\sim\pi,\mathcal{T}}\big[\log(1 - D_\omega(s,a))\big] + \mathbb{E}_{(s,a)\sim\mathcal{M}_E}\big[\log D_\omega(s,a)\big] \tag{4.1}$$

where $D_\omega : \mathcal{S} \times \mathcal{A} \to (0,1)$, $\mathcal{M}_E$ is the buffer with expert demonstrations, and $\mathcal{T}$ is the transition dynamics.

4.2 THE POLICY AND BELIEF MODULES

In a POMDP, the observations are by definition non-Markovian. A policy $\pi(a_t|o_t)$ that chooses actions based on current observations performs sub-optimally, since o_t does not contain sufficient information about the true state of the world. It is useful to infer a distribution on the true states based on the experiences thus far. This is referred to as the *belief state*, and is formally defined as the filtering distribution: $p(s_t|o_{\leq t}, a_{<t})$. It combines the memory of past experiences with uncertainty about unobserved aspects of the world. Let $h_t := (o_{\leq t}, a_{<t})$ denote the observation-action history, and $b_t := \phi(h_t)$ be a function of h_t. If b_t is learned such that it forms the sufficient statistics of the filtering posterior over states, i.e., $p(s_t|o_{\leq t}, a_{<t}) \approx p(s_t|b_t)$, then b_t could be used as a surrogate code (or representation) for the belief state, and be used to train agents in POMDPs. Henceforth, with slight abuse of notation, we would refer to b_t as the *belief*, although it is a high-dimensional representation

rather than an explicit distribution over states.

An intuitive way to obtain this belief is by combining the observation-action history using aggregator functions such as recurrent or convolution networks. For instance, the intermediate hidden states in a recurrent network could represent b_t. In the RL setting with environmental rewards, the representation could be trained by conditioning the policy on it, and back-propagating the RL (e.g. policy gradient) loss. However, the RL signal is generally too weak to learn a rich representation b_t that provides sufficient statistics for the filtering posterior over states. Moreno et al. (2018) provide empirical evidence of this claim by training oracle models where representation learning is supervised with privileged information in form of the (unknown) environment states, and comparing them with learning solely using the RL loss. The problem is only exacerbated when the environmental rewards are extremely sparse. In our imitation learning setup, the belief update is incorporated into the mini-max objective for adversarial imitation of expert trajectories, and hence the representation is learned with a potentially stronger signal (Section 4.2.2). Prior work has shown that representations can be improved by using auxiliary losses such as reward-prediction (Jaderberg et al., 2016), depth-prediction (Mirowski et al., 2016), and prediction of future sensory data (Dosovitskiy & Koltun, 2016; Oh et al., 2015). Inspired by this, in Section 4.3, we regularize the representation with various prediction losses.

Recently, Ha & Schmidhuber (2018) proposed an architecture (World-Models) that decouples model-learning from policy-optimization. In the model-learning phase, a variational auto-encoder compresses the raw observations to latent-space vectors, which are then temporally integrated using an RNN, combined with a mixture density network. In the policy-optimization phase, a policy conditioned on the RNN hidden-states is learned to maximize the rewards. We follow a similar *separation-of-concerns* principle, and divide the architecture into two modules: 1) a **policy module** $\pi_\theta(a_t|b_t)$ which learns a distribution over actions, conditioned on the belief; and 2) a **belief module** B_ϕ which learns a good representation of the belief $b_t := B_\phi(h_t)$, from the history of observations and actions, $h_t := (o_{\leq t}, a_{<t})$. While the policy module is trained with imitation learning, the belief module can be trained in a task-agnostic manner (like in World-Models), or in a task-aware manner.

4.2.1 Policy Module

The goal of our agent is to learn a policy by imitating a few expert demonstration trajectories of the form $\{o_i, a_i\}_{i=0}^{|\tau|}$. Similar to the objective in GAIL, we hope to minimize the Jenson-Shanon divergence between the state-action visitation distributions of the policy and the expert: $\min_\pi D_{JS}[\rho_\pi(s,a) \;||\; \rho_E(s,a)]$. However, since the true environment state s_t is

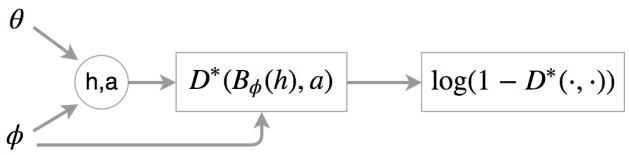


Figure 4.1: Stochastic computation graph for the expectation: $\widetilde{\mathbb{E}}_{(b,a)\sim\pi,\mathcal{T}}\big[\log(1-D^*(b,a))\big]$. Both the policy ($\theta$) and belief module ($\phi$) parameters influence the generation of observation-action sequences, $(h_t, a_t) := \{o_{\leq t}, a_{\leq t}\}$ through environment interaction. $b = B_\phi(h)$ is the belief. Circles represent stochastic nodes; rectangles are deterministic nodes.

unobserved in POMDPs, we modify the objective to involve the belief representation b_t instead, since it characterizes the posterior over s_t via the generative process $p(s_t|b_t)$. Defining the *belief-visitation* distribution $\rho_\pi(b)$ for a policy analogously to the state-visitation distribution, the data processing inequality for f-divergences provides that: $D_{JS}[\rho_\pi(s) \,||\, \rho_E(s)] \leq D_{JS}[\rho_\pi(b) \,||\, \rho_E(b)]$. The objective $\min_\pi D_{JS}[\rho_\pi(b) \,||\, \rho_E(b)]$ thus minimizes an upper bound on the D_{JS} between the state-visitation distributions of the expert and the policy. Further relaxation of this objective allows us to explicitly include the belief-conditioned policy $\pi(a|b)$ into the divergence minimization objective: $D_{JS}[\rho_\pi(b) \,||\, \rho_E(b)] \leq D_{JS}[\rho_\pi(b,a) \,||\, \rho_E(b,a)]$, where $\rho_\pi(b,a) = \rho_\pi(b)\pi(a|b)$ is the belief-action visitation.

Minimizing $\mathbf{D_{JS}}[\rho_\pi(\mathbf{b},\mathbf{a}) \,||\, \rho_\mathbf{E}(\mathbf{b},\mathbf{a})]$. Although explicitly formulating these visitation distributions is difficult, it is possible to obtain an empirical distribution of $\rho_\pi(b,a)$ by rolling out trajectories $(o_1, a_1, \dots)$ from π, and using our belief module to produce samples of belief-action tuples (b_t, a_t), where $b_t := B_\phi(h_t), h_t := (o_{\leq t}, a_{<t})$. Similarly, the expert demonstrations buffer $\mathcal{M}_E$ contains observation-actions sequences, and can be used as an estimate of $\rho_E(b,a)$. To reduce clutter, we shorthand $D_{JS}[\rho_\pi(b,a) \,||\, \rho_E(b,a)]$ with just D_{JS} for the remainder of this chapter. D_{JS} can be approximated (up to a constant scale and shift) with a binary classification problem as exploited in GANs (Goodfellow et al., 2014):

$$D_{JS}(\theta;\phi) \approx \max_\omega \widetilde{\mathbb{E}}_{(b,a)\sim\mathcal{M}_E}\big[\log D_\omega(b,a)\big] + \widetilde{\mathbb{E}}_{(b,a)\sim\pi,\mathcal{T}}\big[\log(1 - D_\omega(b,a))\big] \tag{4.2}$$

where θ are the parameters for the policy $\pi_\theta(a|b)$, D_ω is the discriminator, and $\mathcal{T}$ is the transition dynamics. It should be noted that D_{JS} is a function of the belief module parameters ϕ through its dependence on the belief states. The imitation learning objective for optimizing the policy is then obtained as:

$$\min_\theta D_{JS}(\theta;\phi) \approx \min_\theta \max_\omega \widetilde{\mathbb{E}}_{(b,a)\sim\mathcal{M}_E}\big[\log D_\omega(b,a)\big] + \widetilde{\mathbb{E}}_{(b,a)\sim\pi,\mathcal{T}}\big[\log(1 - D_\omega(b,a))\big] \tag{4.3}$$

In Equation 4.2, denoting the functional maximum over D_ω by D^*, the gradient for policy optimization is: $\nabla_\theta D_{JS}(\theta;\phi) \approx \nabla_\theta \widetilde{\mathbb{E}}_{(b,a)\sim\pi,\mathcal{T}}\big[\log(1-D^*(b,a))\big]$. Figure 4.1 shows the stochastic computation graph (Schulman et al., 2015) for this expectation term, where the stochastic nodes are represented by circles, deterministic nodes by rectangles, and we have written belief as a function of the history. Given fixed belief module parameters (ϕ), the gradient *w.r.t.* θ is obtained using the policy gradient theorem (Sutton et al., 2000):

$$\nabla_\theta D_{JS}(\theta;\phi) \approx \nabla_\theta \widetilde{\mathbb{E}}_{(b,a)\sim\pi,\mathcal{T}}\big[\log(1-D^*(b,a))\big] = \widetilde{\mathbb{E}}_{(b,a)\sim\pi,\mathcal{T}}\big[\nabla_\theta \log \pi_\theta(a|b)\hat{Q}^\pi(b,a)\big]$$
$$\text{where, } \hat{Q}^\pi(b_t,a_t) = \widetilde{\mathbb{E}}_{(b,a)\sim\pi,\mathcal{T}}\big[\sum_{t'=t}^{\infty}\gamma^{t'-t}\log(1-D^*(b_{t'},a_{t'}))\big] \tag{4.4}$$

Therefore, updating the policy to minimize D_{JS} is approximately the same as applying the standard policy-gradient using the rewards obtained from a learned discriminator, $r(b,a) = -\log(1-D^*(b,a))$. As is standard practice, we do not train the discriminator to optimality, but rather jointly train the policy and discriminator using iterative gradient updates. The discriminator is updated using the gradient from Equation 4.2, while the policy is updated with gradient from Equation 4.4. We now detail the update rule for ϕ.

4.2.2 Belief Module

This module transforms the history $(o_{\leq t}, a_{<t})$ into a belief representation. Various approaches could be used to aggregate historical context, such as RNNs, masked convolutions (Gehring et al., 2017) and attention-based methods (Vaswani et al., 2017). In our implementation, we model the belief module B_ϕ with an RNN, such that $b_t = B_\phi(b_{t-1}, o_t, a_{t-1})$. We use GRUs (Cho et al., 2014) as they have been demonstrated to have good empirical performance. We denote by $\mathcal{R}$, a replay-buffer which stores observation-action sequences (current and past) from the agent. As stated before, the belief module could be learnt in a task-agnostic manner (similar to Ha & Schmidhuber (2018)), or with task-awareness.

Task-agnostic learning (separately from policy). An unsupervised approach to learning ϕ without accounting for the agent's objective, is to maximize the joint likelihood of the observation sequence, conditioned on the actions, $\log p(o_{\leq T}|a_{<T})$. This decomposes autoregressively as $\sum_t \log p(o_t|o_{<t}, a_{<t})$. The objective can be optimized by conditioning a generative model for o_t on the RNN hidden state b^ϕ_{t-1} and action a_{t-1}, and using MLE. Using a unimodal Gaussian generative model (learned function g for the mean, and fixed variance),

the autoregressive loss to minimize is:

$$\mathcal{L}^{AR}(\phi) = \mathbb{E}_{\mathcal{R}}||o_t - g(b^{\phi}_{t-1}, a_{t-1})||^2_2 \tag{4.5}$$

Task-aware learning (jointly with policy). Since the policy is conditioned on the belief, an intuitive way to improve the agent's performance is to learn the belief with an objective more aligned with policy-learning. Since the agent minimizes $D_{JS}(\theta, \phi)$, as defined in Equation 4.2, the same imitation learning objective naturally can also be used for learning ϕ:

$$\mathcal{L}^{IM}(\phi) := D_{JS}(\theta, \phi) \approx \widetilde{\mathbb{E}}_{(h,a)\sim\mathcal{M}_E}\big[\log D^*(B_\phi(h), a)\big] + \widetilde{\mathbb{E}}_{\substack{(h,a)\sim \\ \pi_\theta(a|B_\phi(h)),\mathcal{T}}}\big[\log(1 - D^*(B_\phi(h), a))\big] \tag{4.6}$$

which is the same as Equation 4.2 except for the use of the optimal discriminator (D^*), and that we have written the belief in terms of history $b := B_\phi(h)$ to explicitly bring out the dependence on ϕ. The gradient of the first expectation term *w.r.t.* ϕ is straightforward; the gradient of the second expectation term *w.r.t.* ϕ (for given fixed parameters θ) comprises of a policy-gradient term and a pathwise-derivative term (Figure 4.1). Therefore, $\nabla_\phi D_{JS}(\theta; \phi)$ can be approximated with:

$$\begin{aligned}\widetilde{\mathbb{E}}_{(h,a)\sim\mathcal{M}_E}\big[\nabla_\phi \log D^*(B_\phi(h), a)\big] &+ \underbrace{\widetilde{\mathbb{E}}_{(h,a)\sim\pi_\theta(a|B_\phi(h)),\mathcal{T}}\big[\nabla_\phi \log \pi_\theta(a|B_\phi(h))\hat{Q}^\pi\big]}_{\text{policy-gradient term}} \\ &+ \underbrace{\widetilde{\mathbb{E}}_{(h,a)\sim\pi_\theta(a|B_\phi(h)),\mathcal{T}}\big[\nabla_\phi \log(1 - D^*(B_\phi(h), a))\big]}_{\text{pathwise-derivate term}}\end{aligned} \tag{4.7}$$

where $\hat{Q}$ is as defined in Equation 4.4. The overall min-max objective for jointly training the policy, belief and discriminator is:

$$\min_{\phi,\theta}\max_{\omega}\ \widetilde{\mathbb{E}}_{(b,a)\sim\mathcal{M}_E}\big[\log D_\omega(b, a)\big] + \widetilde{\mathbb{E}}_{(b,a)\sim\pi,\mathcal{T}}\big[\log(1 - D_\omega(b, a))\big] \tag{4.8}$$

4.3 BELIEF REGULARIZATION

With the min-max objective (Equation 4.8), it may be possible that the belief parameters (ϕ) are driven towards a degenerate solution that ignores the history ($o_{\leq t}, a_{<t}$), thereby producing constant (or similar) beliefs for policy and expert trajectories. Indeed, if we omit the actions (a) in the discriminator $D_\omega(b, a)$, a constant belief output is an optimal solution for Equation 4.8. To learn a belief representation that captures relevant historical

context and is useful for deriving optimal policies, we add forward-, inverse- and action-regularization to the belief module. We define and motivate them from the perspective of mutual information maximization.

Notation. For two continuous random variables X, Y, mutual information is defined as $I(X;Y) := H(X) - H(X|Y)$, where H denotes the differential entropy. Intuitively, $I(X;Y)$ measures the dependence between X and Y. Conditional mutual information is defined as $I(X;Y|Z) := \mathbb{E}_z[(I(X;Y)|z]$. Given Y, if X and Z are independent ($X \perp Z|Y$), then X, Y, Z form a Markov Chain ($X \to Y \to Z$), and the data processing inequality for mutual information states that $I(X;Z) \leq I(X;Y)$.

Forward regularization. As discussed in Section 4.2, an ideal belief representation completely characterizes the posterior over the true environment states $p(s_t|b_t)$. Therefore, it ought to be correlated with future true states (s_{t+k}), conditioned on the intervening future actions ($a_{t:t+k-1}$). We frame this objective as maximization of the following conditional mutual information: $I(b_t; s_{t+k}|a_{t:t+k-1})$. Since $o_{t+k} \perp b_t|s_{t+k}$ because of the observation generation process in a POMDP, we get the following after using the data processing inequality for mutual information:

$$\begin{aligned} I(b_t; s_{t+k}|a_{t:t+k-1}) &\geq I(b_t; o_{t+k}|a_{t:t+k-1}) \\ &= \mathbb{E}_{a_{t:t+k-1}}\big[H(o_{t+k}|a_{t:t+k-1}) - H(o_{t+k}|b_t; a_{t:t+k-1})\big] \\ &\geq \mathbb{E}_{a_{t:t+k-1}}\Big[H(o_{t+k}|a_{t:t+k-1})\mathbb{E}_{o_{t+k},b_t}\big[\log q(o_{t+k}|b_t; a_{t:t+k-1})\big]\Big] \end{aligned} \tag{4.9}$$

where the final inequality follows because we can lower bound the mutual information using a variational approximation q, similar to the variational information maximization algorithm (Agakov & Barber, 2004). Therefore, we maximize a lower bound to the mutual information $I(b_t; s_{t+k}|a_{t:t+k-1})$ with the surrogate objective:

$$\max_{\phi,q} \mathbb{E}_{\substack{o_{t+k},b_t,\\ a_{t:t+k-1}}}\big[\log q(o_{t+k}|b_t^\phi; a_{t:t+k-1})\big] \tag{4.10}$$

With the choice of a unimodal Gaussian (learned function g for the mean, and fixed variance) for the variational distribution q, the loss function for forward regularization of the belief module is:

$$\mathcal{L}^f(\phi) = \mathbb{E}_{\mathcal{R}}||o_{t+k} - g(b_t^\phi, a_{t:t+k-1})||_2^2 \tag{4.11}$$

where the expectation is over trajectories $(o_1, a_1, \dots)$ sampled from the replay buffer $\mathcal{R}$.

Inverse regularization. It is desirable that the belief at time t is correlated with the past true states (s_{t-k}), conditioned on the intervening past actions ($a_{t-k:t-1}$). This should improve the belief representation by helping to capture long-range dependencies. Proceeding in a manner similar to above, the conditional mutual information between these signals, $I(s_{t-k}; b_t | a_{t-k:t-1})$, can be lower bounded by $I(o_{t-k}; b_t | a_{t-k:t-1})$ using the data processing inequality. As before, this can be further lower bounded using a variational distribution q for generating past observation o_{t-k}. A unimodal Gaussian (mean function g) for q yields the following loss, that is optimized using trajectories from the replay $\mathcal{R}$:

$$\mathcal{L}^i(\phi) = \mathbb{E}_{\mathcal{R}} || o_{t-k} - g(b_t^\phi, a_{t-k:t-1}) ||_2^2 \tag{4.12}$$

Action regularization. We maximize $I(a_{t:t+k-1}; s_{t+k} \mid b_t)$ for the reason that, conditioned on the current belief b_t, a sequence of k subsequent actions ($a_{t:t+k-1}$) should provide information about the resulting true future state (s_{t+k}). Similar lower bounding and use of a variational distribution with mean function g for generating action-sequences gives the loss:

$$\mathcal{L}^a(\phi) = \mathbb{E}_{\mathcal{R}} || (a_{t:t+k-1}) - g(b_t^\phi, o_{t+k}) ||_2^2 \tag{4.13}$$

The complete loss function for training the belief module results from a weighted combination of the imitation-loss and regularization terms. Imitation-loss uses on-policy data and expert demonstrations ($\mathcal{M}_E$), while the regularization losses are computed with on-policy and off-policy data, as well as $\mathcal{M}_E$.

$$\mathcal{L}(\phi) = \mathcal{L}^{IM} + \lambda_1 \mathcal{L}^f + \lambda_2 \mathcal{L}^i + \lambda_3 \mathcal{L}^a \tag{4.14}$$

We derive our final expressions for $(\mathcal{L}^f, \mathcal{L}^i, \mathcal{L}^a)$ by modeling the respective variational distributions (q) as fixed-variance, unimodal Gaussians. We later show that using this simple model results in appreciable performance benefits for imitation learning. Other expressive model classes, such as mixture density networks and flow-based models (Rezende & Mohamed, 2015), can be readily used as well, to learn complex and multi-modal distributions over the future observations (o_{t+k}), past observations (o_{t-k}) and action-sequences ($a_{t:t+k-1}$).

4.4 OVERALL ARCHITECTURE AND ALGORITHM

Figure 4.2 shows the schematic diagram of our complete architecture, including an overview of implemented neural networks. In Algorithm 4.1, we outline the major steps of the train-

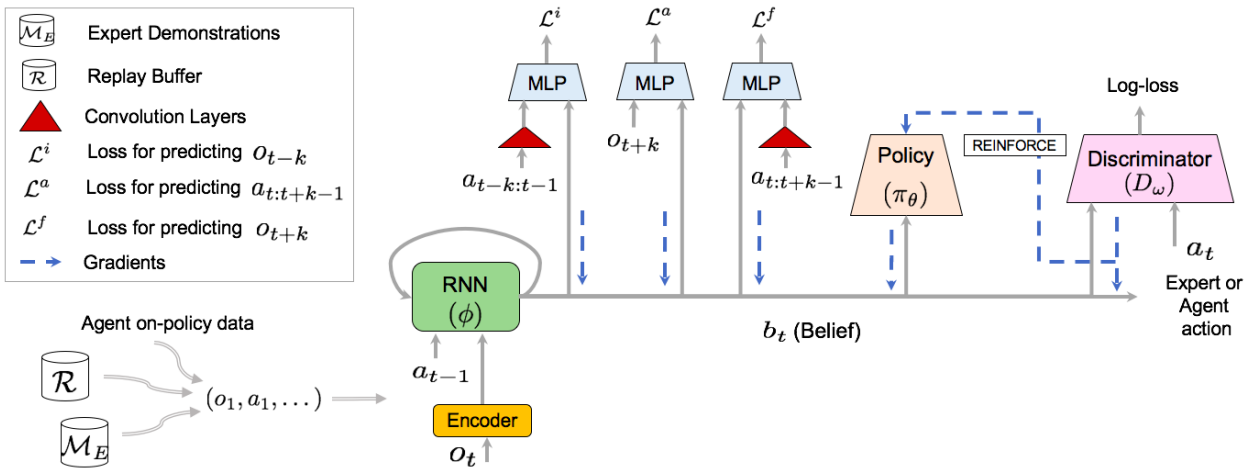


Figure 4.2: Schematic diagram of our complete architecture. The belief module B_ϕ is a recurrent network with GRU cells, and encodes trajectories $(o_1, a_1, \dots)$ from agent (on-policy data), replay buffer $\mathcal{R}$ (off-policy data) and expert demonstrations memory $\mathcal{M}_E$ into belief representations (b_t). B_ϕ is updated with imitation-loss (Equation 4.6) computed from the current policy and discriminator networks. It is further regularized with forward-, inverse- and action-regularization using MLPs (colored in blue in the figure). Convolution layers (colored in red) encode the past actions $(a_{t-k:t-1})$ and future actions $(a_{t:t+k-1})$ into compact representations, which are then fed into the MLPs. The policy $\pi_\theta(a_t|b_t^\phi)$ is conditioned on the belief, and updated using imitation learning (Equation 4.4). The discriminator $D_\omega(b_t^\phi, a_t)$ is a binary classifier trained on tuples from the agent and expert demonstrations (Equation 4.2).

ing procedure. In each iteration, we run the policy for a few steps and obtain shaped rewards from the current discriminator (Line 6). The policy parameters are then updated using A2C, which is the synchronous adaptation of asynchronous advantage actor-critic (A3C; Mnih et al. (2016)), as the policy-gradient algorithm (Line 10). Other RL algorithms, such as those based on trust-regions methods (Schulman et al., 2015) could also be readily used. Similar to the policy (actor), the baseline (critic) used for reducing variance of the stochastic gradient-estimation is also conditioned on the belief. To further reduce variance, Generalized Advantage Estimation (GAE; Schulman et al. (2015)) is used to compute the advantage. Apart from the policy-gradient, on-policy data also enables computing the gradient for the discriminator network (Line 13) and the belief module (Line 14). The belief is further refined by minimizing the regularization losses on off-policy data from the replay buffer $\mathcal{R}$ (Line 15).

The regularization losses $(\mathcal{L}^f, \mathcal{L}^i, \mathcal{L}^a)$ described in Section 4.3 include a hyperparameter k that controls the temporal offset of the predictions. For instance, for $\mathcal{L}^i(\phi; k)$, the larger the k, the farther back in time the observation predictions are made, conditioned on the cur-

Algorithm 4.1: Belief-module Imitation Learning (BMIL)

for *each iteration* **do**
  $d_\pi = \{\}$, $d_E = \{\}$
  /* Rollout c steps from policy */
  repeat
    Get observation o_t from environment
    $a_t \sim \pi_\theta(a_t|b_t)$, where $b_t = B_\phi(o_{\leq t}, a_{<t})$
    $r_t = -\log(1 - D_\omega(b_t, a_t))$
    $d_\pi \leftarrow d_\pi \cup (b_t, a_t, r_t)$
    If o_t is terminal, add rollout $\{o_i, a_i\}_{i=0}^{|\tau|}$ to $\mathcal{R}$
  until $|d_\pi| == c$;
  /* Update Policy */
  Update θ with policy-gradient (Equation 4.4)
  /* Update discriminator ω */
  Fetch $(o_t, a_t, \dots)$ of length c from $\mathcal{M}_E$
  Generate belief-action tuples $d_E = \{(b_i, a_i)\}_{i=t}^{t+c-1}$
  Update ω with log-loss objective using d_π and d_E
  /* Update Belief Module ϕ */
  Update ϕ with $\nabla_\phi \mathcal{L}(\phi)$ using d_π and d_E (Equation 4.14)
  /* Off-policy Updates */
  for *few update steps* **do**
    Fetch $(o_t, a_t, \dots)$ of length c from $\mathcal{R}$
    Update ϕ with $\nabla_\phi(\lambda_1 \mathcal{L}^f + \lambda_2 \mathcal{L}^i + \lambda_3 \mathcal{L}^a)$
  end
end

rent belief and past actions. The temporal abstractions provided by multi-step predictions (k>1) should help to extract more global information from the input stream into the belief representation. Our ablations (Section 4.5.3) show the performance benefit of including multi-step losses. Various strategies for selecting k are possible, such as uniform sampling from a range (Guo et al., 2018) and adaptive selection based on a curriculum (Oh et al., 2015). For simplicity, we choose fixed values, and leave the exploration of the more sophisticated approaches to future work. Hence, our total regularization loss comprises of single-step (k=1) and multi-step (k=5) forward-, inverse-, and action-prediction losses. For encoding a sequence of past or future actions into a compact representation, we use multi-layer convolution networks (Figure 4.2).

4.5 EXPERIMENTS

The goal in this section is to evaluate and analyze the performance of our proposed architecture for imitation learning in partially-observable environments, given some expert demonstrations. Herein, we describe our environments, provide comparisons with GAIL, and perform ablations to study the importance of the design decisions that motivate our architecture.

Partially-observable locomotion tasks. We benchmark high-dimensional, continuous-control locomotion environments based on the MuJoCo physics simulator, available in OpenAI Gym (Brockman et al., 2016). The standard Gym MuJoCo library of tasks, however, consists of MDPs (and not POMDPs), since observations in such tasks contain sufficient state-information to learn an optimal policy conditioned on only the current observation. As such, it has been extensively used to evaluate performance of reinforcement-learning and imitation-learning algorithms in the MDP setting (Schulman et al., 2017; Ho & Ermon, 2016). To transform these tasks into POMDPs, we follow an approach similar to Duan et al. (2016), and redact some sensory data from the observations. Specifically, from the default observations, we remove measurements for the translation and angular velocities of the torso, and also the velocities for all the link joints. We denote the original (MDP) observations by $s \in \mathcal{S}$, and the curtailed (POMDP) observations by $o \in \mathcal{O}$.

For all experiments, we assume access to 50 expert demonstrations of the type $\{o_i, a_i\}_{i=0}^{|\tau|}$, for each of the tasks. The policy and discriminator networks are feed-forward MLPs with two 64-unit layers. The policy network outputs include the action mean and per-action variances (i.e. actions are assumed to have an independent Gaussian distribution). In the belief module, the dimension of the GRU cell is 256, while the MLPs used for regularization have two 64-unit feed-forward layers.

4.5.1 Comparison to GAIL

Our first baseline is modeled after the architecture used in the original GAIL approach (Ho & Ermon, 2016). It consists of feed-forward policy and discriminator networks, without the recurrent belief module. The policy is conditioned on o_t, and the discriminator performs binary classification on (o_t, a_t) tuples. The update rules for the policy and discriminator are obtained in similar way as Equation 4.4 and Equation 4.2, respectively, by replacing the belief b_t with observation o_t. The next baseline, referred to as GAIL+Obs. stack, augments GAIL by concatenating 3 previous observations to each o_t, and feeding the entire

stack as input to the policy and discriminator networks. This approach has been found to extract useful historical context for a better state-representation (Mnih et al., 2015). We abbreviate our complete proposed architecture (Figure 4.2) by BMIL, short for Belief-Module Imitation Learning. BMIL jointly trains the policy, belief and discriminator networks using a mini-max objective (Equation 4.8), and additionally regularizes the belief with multi-step predictions. Table 4.1 compares the performance of different designs on POMDP MuJoCo. We shows the mean episode-returns, averaged over 5 runs with random seeds, after 10M timesteps of interaction with the environment. We observe that GAIL—both with and without observation stacking—is unable to successfully imitate the expert behavior. Since the observation o_t alone does not contain adequate information, the policy conditioned on it performs sub-optimally. Also, the discriminator trained on (o_t, a_t) tuples does not provide robust shaped rewards. Using the full state s_t instead of o_t in our experiments leads to successful imitation with GAIL, suggesting that the performance drop in Table 4.1 is due to partial observability, rather than other artifacts such as insufficient network capacity, or lack of algorithmic or hyperparameter tuning. Further, we see that techniques such as stacking past observations provide only a marginal improvement in some of the tasks. In contrast, in BMIL, the belief module curates a belief representation from the history $(o_{\leq t}, a_{<t})$, which is used both for discriminator training, and to condition the action-distribution (policy). BMIL achieves scores very close to those of the expert.

	GAIL	GAIL + Obs. stack	BMIL (Ours)	Expert ($\approx$ Avg.)
Inv.DoublePend.	109	1351	**9104**	9300
Hopper	157	517	**2665**	3200
Ant	895	1056	**1832**	2400
Walker	357	562	**4038**	4500
Humanoid	1686	1284	**4382**	4800
Half-cheetah	205	-948	**5860**	6500

Table 4.1: Mean episode-returns after 10M timesteps in POMDP MuJoCo.

4.5.2 Comparison to GAIL with Recurrent Networks

For our next two baselines, we replace the feed-forward networks in GAIL with GRUs. GAIL-RF uses a recurrent policy and a feed-forward discriminator, while in GAIL-RR, both the policy and the discriminator are recurrent. In both these baselines, the belief is created internally in the recurrent policy module. Importantly, unlike BMIL, the belief is not shared between the policy and the discriminator. The average final performance of GAIL-

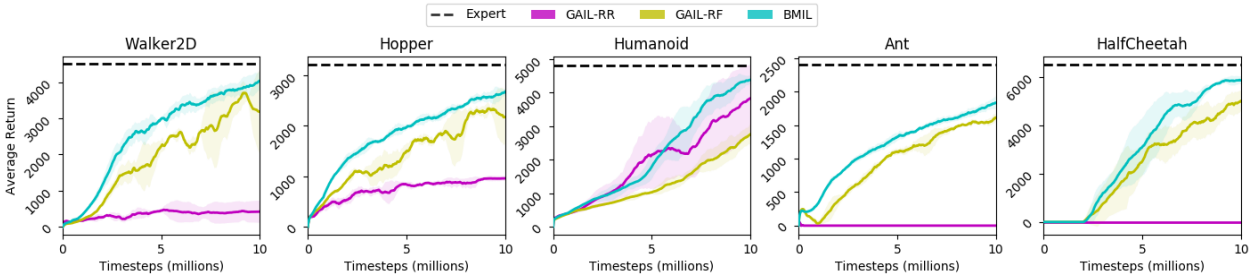


Figure 4.3: Mean episode-returns vs. timesteps of environment interaction. BMIL is our proposed architecture (Figure 4.2); GAIL-RF uses a recurrent policy and a feed-forward discriminator, while in GAIL-RR, both the policy and the discriminator are recurrent.

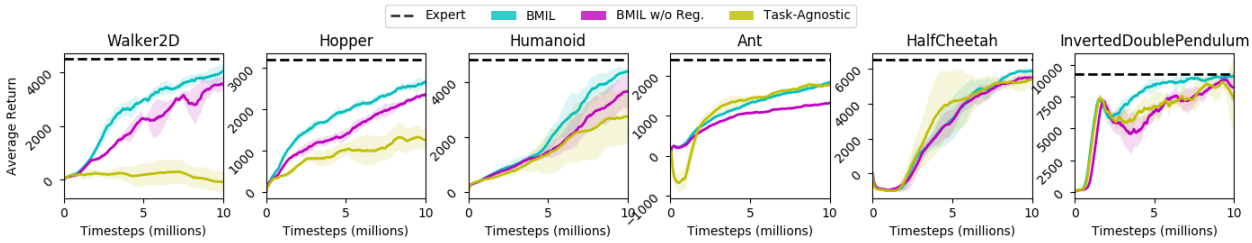


Figure 4.4: Mean episode-returns vs. timesteps of environment interaction. BMIL is our proposed architecture (Figure 4.2); BMIL w/o Reg excludes the various regularization terms (Section 4.3) from this design; *Task-Agnostic* learns the belief module separately from the policy using a task-agnostic loss ($\mathcal{L}^{AR}$, Section 4.2.2).

RF and GAIL-RR in our POMDP environments is shown in Table 4.2. We observe that GAIL-RR does not perform well on most of the tasks. A plausible explanation for this is that using the adversarial binary classification loss for training the discriminator parameters does not induce a sufficient representation of the belief state in its recurrent network. The other baseline, GAIL-RF, avoids this issue with a feed-forward discriminator trained on (o_t, a_t) tuples from the expert and the policy. This leads to much better performance. However, BMIL consistently outperforms GAIL-RF, most significantly in *Humanoid* (1.6× higher score), indicating the effectiveness of the decoupled architecture and other design decisions that motivate BMIL. Figure 4.3 plots the learning curves for these experiments.

4.5.3 Ablation Studies

How crucial is belief regularization? We compare the performance of our architecture with and without belief regularization (BMIL vs. BMIL w/o Reg.). Figure 4.4 plots the mean episode-returns vs. timesteps of environment interaction. We observe that including

	GAIL-RR	GAIL-RF	BMIL (Ours)
Inv.DoublePend.	8965	9103	**9104**
Hopper	955	2164	**2665**
Ant	-533	1612	**1832**
Walker	400	3188	**4038**
Humanoid	3829	2761	**4382**
Half-cheetah	-922	5011	**5860**

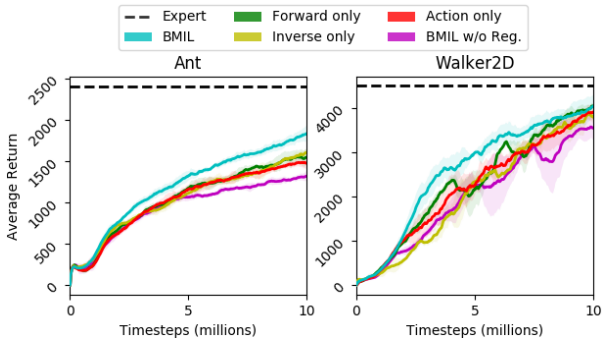
Table 4.2: Mean episode-returns after 10M timesteps in POMDP MuJoCo.

regularization leads to better episode-returns and sample-complexity for most of the tasks considered, indicating that it is useful for shaping the belief representations.

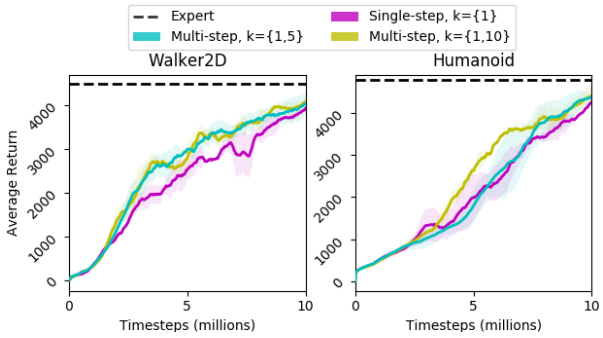
Task-aware vs. Task-agnostic belief learning. Next, we compare with a design in which the belief module is trained separately from the policy, using a task-agnostic loss ($\mathcal{L}^{AR}$, Section 4.2.2). This echos the philosophy used in World-Models (Ha & Schmidhuber, 2018). As Figure 4.4 shows, this results in mixed success for imitation in POMDPs. While the agent achieves good scores in tasks such as Ant and HalfCheetah, the performance in Walker and Hopper is unsatisfactory. In contrast, BMIL, which uses a task-aware imitation-loss for the belief module, is consistently better.

Are all of $\mathcal{L}^f, \mathcal{L}^i, \mathcal{L}^a$ useful? We introduced 3 different regularization terms for the belief module – forward ($\mathcal{L}^f$), inverse ($\mathcal{L}^i$) and action ($\mathcal{L}^a$). To assess their benefit individually, in Figure 4.5a, we plot learning curves for two tasks, with each of the regularizations applied in isolation. We compare them with BMIL, which includes all of them, and BMIL w/o Reg., which excludes all of them (no regularization). For the Ant task, we notice that each of $\{\mathcal{L}^f, \mathcal{L}^i, \mathcal{L}^a\}$ provides performance improvement over the no-regularization baseline, and combining them (BMIL) performs the best. With the Walker task, we see better mean episode-returns at the end of training with each of $\{\mathcal{L}^f, \mathcal{L}^i, \mathcal{L}^a\}$, compared to no-regularization; BMIL attains the best sample-complexity.

Are multi-step predictions useful? As argued before, making coarse-scale, multi-step (k>1) predictions for forward, inverse observations and action-sequences could improve representations by providing temporal abstractions. In Figure 4.5b, we plot BMIL, which uses single- and multi-step losses k=$\{1, 5\}$, and compare it with two versions: first that uses a different temporal offset k=$\{1, 10\}$, and second that predicts only at the single-step granularity k=$\{1\}$. For both tasks, we get better sample-complexity and higher final average returns with multi-step, suggesting its positive contribution to representation learning.



(a) Ablation on components of belief regularization. Forward-, Inverse-, Action-only correspond to using $\mathcal{L}^f$, $\mathcal{L}^i$, $\mathcal{L}^a$, respectively, in isolation, without the other two.



(b) Ablation on hyperparameter k in the regularization terms. Multi-step design builds over single-step by adding predictions at different temporal offsets, k=5 and k=10.

Figure 4.5: Ablation studies.

4.6 RELATED WORK AND CONCLUSION

While we cannot do full justice to the extensive literature on RL-algorithms for POMDPs, we here mention some recent related work. Most prior algorithms for POMDPs assume access to a predefined reward function. These include approaches based on Q-learning (DRQN; Hausknecht & Stone (2015)), policy-gradients (Igl et al., 2018), partially observed guided policy search (Zhang et al., 2016), and planning methods (Silver & Veness, 2010; Ross et al., 2008; Pineau et al., 2003). In contrast, we propose to adapt ideas from generative adversarial imitation learning to learn policies in POMDPs without environmental rewards.

Learning belief states from history $(o_{\leq t}, a_{<t})$ was recently explored in Guo et al. (2018). The authors show that training the belief representation with a Contrastive Predictive Coding (CPC, Oord et al. (2018)) loss on future observations, conditioned on future actions, helps to infer knowledge about the underlying state of the environment. Predictive State Representations (PSRs) offer another approach to modeling the belief state in terms of observable data (Littman & Sutton, 2002). The assumption in PSRs is that the filtering distribution can be approximated with a distribution over the k future observations, conditioned on future actions, $p(s_t|o_{\leq t}, a_{<t}) \approx p(o_{t+1:t+k}|o_{\leq t}, a_{<t+k})$. PSRs combined with RNNs have been shown to improve representations by predicting future observations (Venkatraman et al., 2017; Hefny et al., 2018). While we also make future predictions, a key difference compared to aforementioned methods is that our belief representation is additionally regularized by predictions in the past, and in action-space, which we later show benefits our approach.

State-space models (SSMs; Fraccaro et al. (2016); Goyal et al. (2017); Buesing et al. (2018)), which represent the unobserved environment states with latent variables, have also

been used to obtain belief states. Igl et al. (2018) use a particle-filtering method to train a VAE, and represent the belief state with a weighted collection of particles. The model is also updated with the RL-loss using a belief-conditioned policy. Gregor et al. (2018) proposed TD-VAE, which explicitly connects belief distributions at two distant timesteps, and enforces consistency between them using a transition distribution and smoothing posterior. Although we use a deterministic model for our belief module (B_ϕ), our methods apply straightforwardly to SSMs as well.

4.6.1 Conclusion

In this work, we study imitation learning for POMDPs, which has been considerably less explored compared to imitation learning for MDPs, and learning in POMDPs with predefined reward functions. We introduce a framework comprised of a belief module, and policy and discriminator networks conditioned on the generated belief. Crucially, all networks are trained jointly with a min-max objective for adversarial imitation of expert trajectories.

Within this flexible setup, many instantiations are possible, depending on the choice of networks. Both feed-forward and recurrent networks can be used for the policy and the discriminator, while for the belief module there is an expansive set of options based on the rich literature on representation learning, such as CPC (Guo et al., 2018) and using auxiliary tasks (Jaderberg et al., 2016). Many more methods based on state-space latent-variable models are also applicable. In our instantiation of the belief module, we use the task-based imitation loss (Equation 4.6), and improve robustness of representations by regularizing with multi-step prediction of past/future observations and action-sequences. One benefit of our proposed framework is that in future work, it would be straightforward to substitute other methods for learning belief representations for the one we arrived at in our work. Similarly, recent advancements in GAN and RL literature could guide the development of better discriminator and policy networks for imitation learning in POMDPs. Exploring these ranges, as well as their interplay, are important directions.

CHAPTER 5: IMITATION LEARNING FROM OBSERVATIONS UNDER TRANSITION MODEL DISPARITY

5.1 INTRODUCTION

Imitation Learning (IL) is a framework that trains an agent to perform desired skills by leveraging expert demonstrations of those skills. Compared to the standard Reinforcement Learning (RL) approach, IL offers the benefit of not requiring a reward function, that can be difficult to specify for complicated objectives. Recent IL methods that integrate efficiency with deep-RL and are performant in high-dimensional state-action spaces include behavioral-cloning-based algorithms (Ross et al., 2011; Brantley et al., 2019), and adversarial IL algorithms inspired by maximum entropy inverse-RL (Ho & Ermon, 2016; Finn et al., 2016). Imitation Learning from Observations (ILO) refers to the setting where the expert demonstrations consist of only observations (or states), while the expert actions are unavailable. ILO is beneficial when the measurement of the expert action is difficult, *e.g.*, in kinesthetic teaching in robotics or when learning with motion capture datasets.

Adversarial IL methods frame the problem as the minimization of an f-divergence between the state-action visitation distributions of the expert and the learner (Ke et al., 2019). Since expert actions are absent in ILO, the analogous methodology here is to minimize the f-divergence between the state-transition distributions of the expert and the learner (Arumugam et al., 2020). A state-transition distribution for a policy is the joint distribution over the current state and the next state, and is defined formally in Section 5.2. Choosing the f-divergence to be the Jensen–Shannon (JS) divergence has enabled successful imitation using algorithms such as GAIL (Ho & Ermon, 2016) and GAIfO (Torabi et al., 2018), for IL and ILO, respectively.

The transition dynamics model of an environment governs the distribution over the next state, given the current state and action. In this work, we focus on ILO in the scenario of a transition dynamics mismatch between the expert and the learner environments. Such discrepancy could manifest in real-world applications of imitation learning when there are subtle differences in the physical attributes of the system used to collect the demonstrations and the system where the learner policy is run. Adversarial ILO methods such as GAIfO, that attempt to train the learner by matching its state-transition distribution with that of the expert, perform very well when the expert and learner operate in a shared environment, under the same dynamics. When the dynamics differ, however, matching the state-transition distributions becomes challenging since the state transitions provided in the expert demonstrations could be infeasible under the dynamics in the learner's environment. To form some

intuition, consider ILO in a 2D grid with discrete states, where the path demonstrated by the expert follows the main diagonal from the bottom-left grid-cell to the top-right grid-cell. Suppose that the transition dynamics for the learner environment are such that only horizontal and vertical moves are permitted on the grid, *i.e.,* no diagonal motion for the learner. In this case, an optimal learner's movement is still in the same general direction as the expert, and it covers all the expert states (along with some adjacent states). However, matching the state-transition distributions is an ineffective strategy since the learner cannot reproduce the (one-step) state transitions of the expert.

To alleviate the challenges of ILO under dynamics mismatch, we propose an algorithm that trains an intermediary policy in the learner environment, and hope to use it as a surrogate expert for training the learner (imitator). We refer to this policy as the *advisor*. For the advisor to be effective, the state transitions generated by it in the learner environment should be as close as possible to the state transitions in the expert dataset. We formalize this concept in terms of the cross-entropy distance between state-conditional next-state distributions of the expert and the advisor. To convert this into a practical and scalable algorithm for training the advisor, we incorporate ideas from distribution support estimation (Wang et al., 2019). Simultaneous to the advisor training, the learner agent is updated to imitate the advisor. Crucially, the advisor operates in the same environment as the learner, making the distribution-matching IL objective amenable.

We evaluate the efficacy of our ILO algorithm using five locomotion control tasks from OpenAI Gym where we introduce a mismatch between the dynamics of the expert and the learner by changing different configuration parameters. We demonstrate that our approach compares favorably to the baseline ILO algorithms in many of the considered scenarios.

5.2 PRELIMINARIES

We model the RL environment as a discounted, infinite horizon Markov Decision Process (MDP). At every discrete timestep, the agent observes a state ($s \in \mathcal{S}$), generates an action ($a \in \mathcal{A}$) from a stochastic policy $\pi(a|s)$, receives a scalar reward $r(s,a)$, and transitions to the next state (s') sampled from the transition dynamics model $p(s'|s,a)$. In the infinite horizon setting, the future rewards are discounted by a factor of $\gamma \in [0,1)$. Let $d^t_\pi(s)$ denote the distribution induced by π over the state-space at a particular timestep t. The stationary discounted state distribution of π is then defined as $\rho_\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t d^t_\pi(s)$. The RL objective of maximizing the cumulative discounted sum of rewards can be framed as $\max_\pi \mathbb{E}_{\rho_\pi(s,a)}[r(s,a)]$, where $\rho_\pi(s,a) = \rho_\pi(s)\pi(a|s)$ is the state-action distribution (also known as the occupancy measure). Lastly, we define the state-transition distribution for a

policy as $\rho_\pi(s, s') = \sum_a p(s'|s, a)\rho_\pi(s, a)$ [1].

5.2.1 GAIL and GAIfO

Generative Adversarial Imitation Learning (Ho & Ermon, 2016) is a widely popular model-free IL method that builds on the Maximum Causal Entropy Inverse-RL (MaxEnt-IRL) framework (Ziebart, 2010). MaxEnt-IRL models the expert behavior with a policy that maximizes its γ-discounted causal entropy, $\mathcal{H}(\pi) = \mathbb{E}_\pi[-\log \pi(a|s)]$, while satisfying a feature matching constraint. GAIL considers a regularized version of the dual to this primal problem. It shows that RL with the reward function recovered as the solution of the regularized dual is equivalent to directly learning a policy whose state-action distribution is similar to that of the expert. For a specific choice of the regularizer, this similarity is quantified by the JS divergence between the two state-action distributions, $D_{JS}[\rho_\pi(s, a) \;||\; \rho_{\pi_e}(s, a)]$. Based on these ideas, GAIL seeks to learn a policy with the objective:

$$\min_\pi \max_D \; \mathbb{E}_{\rho_\pi}\big[\log D(s, a)\big] + \mathbb{E}_{\rho_{\pi_e}}\big[\log(1 - D(s, a))\big] - \lambda\mathcal{H}(\pi) \tag{5.1}$$

where $D : \mathcal{S} \times \mathcal{A} \to (0, 1)$ is the discriminator that provides the rewards for training the learner policy π, and the inner maximization over D approximates $D_{JS}(\cdot)$ similar to GANs (Goodfellow et al., 2014). To empirically estimate the expectation under $\rho_{\pi_e}(s, a)$, state-action pairs are sampled from the available expert demonstrations. In the ILO setting, however, expert actions are not included in the demonstrations. To mitigate this challenge, Torabi et al. (2018) propose GAIfO, which adapts to ILO by modifying the GAIL objective to match the *state-transition* distributions of the expert and the learner, *i.e.*, $D_{JS}[\rho_\pi(s, s') \;||\; \rho_{\pi_e}(s, s')]$. Correspondingly, the discriminator in GAIfO is a function of the state transitions $D(s, s')$.

5.2.2 Support Estimation via RED

We summarize a recently proposed method for estimating the support of a distribution in high dimensions (RED; Wang et al. (2019)), since it forms a core ingredient of our final algorithm. Let $\mathcal{X}$ denote a set and p be a probability distribution on $\mathcal{X}$. Denote by $\text{supp}(p) = \{x \in \mathcal{X} \mid p(x) \neq 0\}$, the support of the distribution p. Given any $x \in \mathcal{X}$, the task is to know if $x \in \text{supp}(p)$. Towards this goal, Wang et al. (2019) combine ideas from kernel-based

[1] With slight abuse of notation, we use the symbol ρ_π for state, state-action, and state-transition distributions of a policy π. We would provide context around their usage to avoid any confusion.

support estimation (De Vito et al., 2014) and RND (Burda et al., 2018), and consider the following objective:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{x\sim p(x)} \|f_\theta(x) - f_{\tilde{\theta}}(x)\|_2^2 \tag{5.2}$$

where $f_\theta : \mathcal{X} \to \mathbb{R}^K$ is a trainable function parameterized by θ, while $f_{\tilde{\theta}}$ is a *fixed* function with randomly initialized parameters $\tilde{\theta}$. Define the score function as (with constant positive scalar λ):

$$r_{\text{RED}}(x) = \exp(-\lambda \|f_{\theta^*}(x) - f_{\tilde{\theta}}(x)\|_2^2) \tag{5.3}$$

Wang et al. (2019) conclude that the score $r_{\text{RED}}(x)$ is high if $x \in \text{supp}(p)$, and is low otherwise. With neural network function approximators, we thus obtain a smooth metric whose value decreases (increases) as we move farther from (closer to) the support of the distribution p.

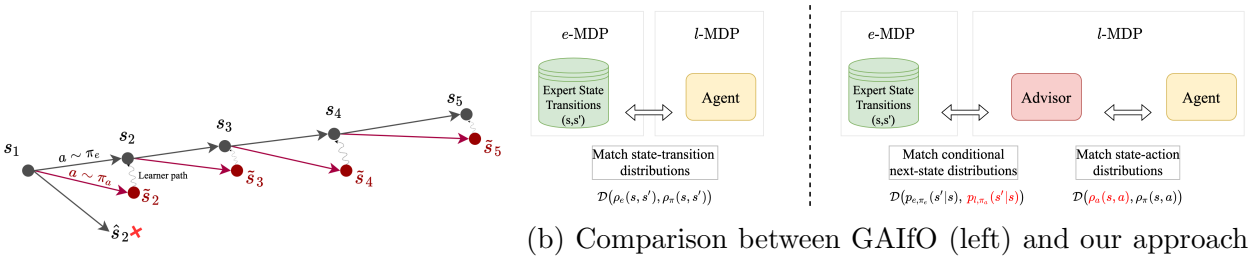
5.3 ADVISOR-AUGMENTED IMITATION LEARNING FROM OBSERVATIONS

We begin by defining some notations for our setup. We denote the MDP in which the expert policy (π_e) operates as the e-MDP, while the learner (or imitator) policy is run in the l-MDP. The two MDPs share all the attributes, except for the transition dynamics function, which we symbolize with $p_e(s'|s,a)$ and $p_l(s'|s,a)$, for the e-MDP and l-MDP, respectively. $\rho_e(s)$, $\rho_e(s,a)$, and $\rho_e(s,s')$ are the state, state-action, and state-transition distribution for the expert policy. These distributions depend on the e-MDP dynamics $p_e(s'|s,a)$. The corresponding distributions for the learner are denoted by $\rho_\pi(\cdot)$ and they depend on the l-MDP dynamics $p_l(s'|s,a)$.

Adversarial ILO methods, such as GAIfO, that learn the imitator policy by matching the state-transition distributions of the expert and learner aim to solve the following primal problem:

$$\max_{\pi} \mathcal{H}(\pi) \quad \text{s.t.} \quad \rho_e(s,s') = \rho_\pi(s,s') \quad \forall\, (s,s') \in \mathcal{S} \times \mathcal{S} \tag{5.4}$$

If the expert and the learner operate under different transition dynamics, depending on the extent of the mismatch, it is possible that some (or all) of the one-step state transitions of the expert are infeasible under the dynamics function in l-MDP. Said differently, given a (s_e, s'_e) pair sampled from the expert demonstrations, there could be no action in the l-MDP from the state s_e that results in s'_e as the next state, as per the dynamics $p_l(s'|s,a)$. This renders the state-transition matching objective hard to optimize in practice. For instance, in the practical implementation of GAIfO, the rewards for the imitator policy are computed from a discriminator that is trained with binary classification on (s,s') pairs from the expert



(a) State transitions

(b) Comparison between GAIfO (left) and our approach (right)

Figure 5.1: (a) A sequence of states s_i from the expert dataset. The states $\tilde{s}_i$ are reached by sampling an action from the advisor policy π_a from every expert state. State $\hat{s}_2$ is less desirable than $\tilde{s}_2$ since the latter is closer to the expert state s_2. The squiggly lines show the path that a learner, that is optimized to match the state-action distribution of the advisor, may take. (b) A high-level overview of our approach. While GAIfO directly matches the state-transition distributions of the expert and the learner, we learn an intermediary policy (advisor) in the l-MDP that acts as the surrogate expert for the learner. $\mathcal{D}$ is used to denote the corresponding distance metric.

and the learner. If the expert transitions can't be generated in l-MDP by the learner, then a high-capacity discriminator could achieve perfect accuracy and thus fail to provide informative rewards for imitation.

Consider a sequence of states $\{s_1, s_2, \dots\}$ generated by the expert policy in the e-MDP, as shown in Figure 5.1a. Given an expert state s_i, it may not be possible to reach s_{i+1} in a single timestep in the l-MDP. Instead, we would like to find an alternative state $\tilde{s}_{i+1}$ that *is* reachable from s_i in one step (*i.e.,* feasible under the dynamics p_l) and is close to the desired destination s_{i+1}. For instance, in Figure 5.1a, starting from the expert state s_1, $\tilde{s}_2$ is a more desirable next state compared to $\hat{s}_2$.

5.3.1 Guidance via an Advisor Policy

To discover such feasible states $\tilde{s}_i$, we introduce an advisor policy π_a that operates in the l-MDP. π_a is invoked only for action selection on the expert states, rather than being run in a closed feedback loop in the learner environment. In this way, π_a is akin to a contextual bandit policy. The goal with π_a is to produce an action $a_i \sim \pi_a(\cdot|s_i)$ from the expert state s_i such that the next state in the l-MDP, $\tilde{s}_{i+1} \sim p_l(\cdot|s_i, a_i)$, is close to the next state s_{i+1} in the e-MDP under the expert policy. We expand on the measure of *closeness* and the methodology to train the advisor in the next subsection.

When the expert and learner dynamics are the same, the optimal advisor would take the same actions as the expert policy. In the presence of a dynamics mismatch, however, the

advisor actions provide reasonably good guidance on how to stay close to the expert's state trajectory (Figure 5.1a). Therefore, the advisor π_a could act as a *surrogate expert* for the learner and the learner's imitation learning objective could be suitably modified. The main advantage of the learner and the advisor operating in the same l-MDP is that we could now attempt to match their visitation distributions, which is much more structured than matching distributions across dynamics (Equation 5.4). We consider the IL objective:

$$\max_{\pi} \mathcal{H}(\pi) \quad \text{s.t.} \quad \rho_{\pi_a}(s,a) = \rho_{\pi}(s,a) \quad \forall\, (s,a) \in \mathcal{S} \times \mathcal{A} \tag{5.5}$$

where $\rho_{\pi_a}(s,a)$ is the state-action distribution of π_a. Since the advisor is invoked only on the expert states, effectively, $\rho_{\pi_a}(s,a) = \rho_e(s)\pi_a(a|s)$. We can convert the above objective to an unconstrained optimization by using a parametric function $f_\omega(s,a)$ as the Lagrange multiplier:

$$\min_{\omega} \max_{\pi} \mathcal{J}(\pi,\omega) \coloneqq \mathcal{H}(\pi) + \mathbb{E}_{\rho_e(s)\pi_a(a|s)}[f_\omega(s,a)] - \mathbb{E}_{\rho_\pi(s,a)}[f_\omega(s,a)] \tag{5.6}$$

We note that this objective bears resemblance to entropy-regularized apprenticeship learning (Abbeel & Ng, 2004; Syed et al., 2008; Syed & Schapire, 2008), modulo the use of an advisor policy that contributes the favorable actions, instead of the expert policy.

5.3.2 Training the Advisor

The advisor policy generates an action (deterministic π_a), or a distribution over actions (stochastic π_a), given an expert state sampled from the demonstration data. One suitable objective for training the advisor is to minimize the dissimilarity between the destination state achieved with the expert policy in e-MDP and that achieved with the advisor in l-MDP. For instance, consider the scenario where the dynamics functions are deterministic and known, denoted by f_e and f_l for the e-MDP and l-MDP, respectively. Also, the policies π_e and π_a are deterministic. Then, the advisor could be optimized with the following loss:

$$\min_{\pi_a} \mathcal{J}(\pi_a) \coloneqq \mathbb{E}_{s\sim\rho_e(s)}\Big[\mathcal{D}\big[f_e(s,\pi_e(s)), f_l(s,\pi_a(s))\big]\Big] \tag{5.7}$$

where $\mathcal{D}$ is a distance measure, *e.g.,* the L_2-norm in the state space. However, we are interested in the setup with stochastic, unknown dynamics functions and stochastic policies. To compute a distance metric in this setting, we first define the state-conditional next-state

distributions for the expert and the advisor, by marginalizing over the actions:

$$p_{e,\pi_e}(s'|s) = \sum_a p_e(s'|s,a)\pi_e(a|s) \quad ; \quad p_{l,\pi_a}(s'|s) = \sum_a p_l(s'|s,a)\pi_a(a|s) \tag{5.8}$$

We then use the cross-entropy distance ($\mathbb{H}$) between these distributions as a measure of closeness between the expert and the advisor:

$$\begin{aligned}
\pi_a^* &= \arg\min_{\pi_a} \mathcal{J}(\pi_a) \coloneqq \mathbb{E}_{s\sim\rho_e(s)}\Big[\mathbb{H}\big[p_{l,\pi_a}(s'|s),\ p_{e,\pi_e}(s'|s)\big]\Big] \\
&= \arg\max_{\pi_a} \mathcal{J}(\pi_a) \coloneqq \mathbb{E}_{s\sim\rho_e(s)}\mathbb{E}_{a\sim\pi_a(a|s)}\mathbb{E}_{s'\sim p_l(s'|s,a)}\big[\log p_{e,\pi_e}(s'|s)\big] \\
&= \arg\max_{\pi_a} \mathcal{J}(\pi_a) \coloneqq \mathbb{E}_{s\sim\rho_e(s)}\mathbb{E}_{a\sim\pi_a(a|s)}\mathbb{E}_{s'\sim p_l(s'|s,a)}\big[\log \rho_e(s,s')\big]
\end{aligned} \tag{5.9}$$

where, in the last equation, we have multiplied the term inside the log with $\rho_e(s)$, a quantity independent of π_a. Figure 5.1b provides a high-level overview of our approach.

5.3.3 An Approximation Based on RED

The objective in Equation 5.9 presents a couple of challenges. Firstly, it is infeasible to evaluate the density of any state transition (s, s') under the expert's state-transition distribution $\rho_e(s, s')$, since this distribution is unknown and only a few samples from this distribution are available to us in the form of the expert demonstrations. Secondly, and more importantly, note that the state transition that ought to be evaluated under $\rho_e(s, s')$ is generated by the advisor policy in the l-MDP. If no advisor policy can replicate the state transition behavior of the expert, as is likely when there is a dynamics mismatch, then the objective becomes degenerate with an optimal value of $-\infty$.

Our approach to mitigate these issues is to replace the log-density term in Equation 5.9 with an estimated value that quantifies the *proximity* of a given state transition (s, s') to the manifold of the expert's support in this space, *i.e.,* $\rho_e(s, s')$. To get this value, we leverage RED (Wang et al., 2019), which pre-trains a deep neural network using data samples from the distribution of interest. Then, given a test sample, the network outputs a continuous value that provides an estimate of how far the test sample is from the distribution's support. Section 5.2.2 provides a short background on RED.

In our instantiation, we pre-train a RED network $r^{\phi}_{\text{RED}}(s, s')$ with state transitions from the expert demonstrations. The network is then frozen and utilized in the objective to train the advisor:

$$\max_{\pi_a} \mathcal{J}(\pi_a) \coloneqq \mathbb{E}_{s\sim\rho_e(s)}\mathbb{E}_{a\sim\pi_a(a|s)}\mathbb{E}_{s'\sim p_l(s'|s,a)}\big[r^{\phi}_{\text{RED}}(s,s')\big] \tag{5.10}$$

↓ Increasing noise	Walker2d	HalfCheetah
N/S = 0	0.93	0.96
N/S = 0.2	0.80	0.71
N/S = 0.5	0.44	0.23
N/S = 1	0.12	0.04
N/S = 2	0.01	0.001

Table 5.1: Averaged $r^{\phi}_{\text{RED}}(s, s')$ values for different noise levels.

For any advisor-generated state transition, $r^{\phi}_{\text{RED}}(s, s')$ provides a smooth metric in the range $(0, 1]$, whose value increases (decreases) as the transition moves closer to (farther from) the support of the distribution $\rho_e(s, s')$. Thus, training π_a to maximize this value yields an advisor that provides guidance on how to stay close to the expert's state trajectory when operating in l-MDP (Figure 5.1b).

To provide further intuition on the use of RED, Table 5.1 shows the $r^{\phi}_{\text{RED}}(s, s')$ values obtained from a trained RED network in two environments— *Walker2d* and *HalfCheetah*. The RED network is trained with 50 state transitions $\{s_e, s'_e\}$ sampled from $\rho_e(s, s')$ and then evaluated under several noisy transitions $\{s_e, s'_e + \eta\}$, where η denotes zero-mean Gaussian noise. We show the r^{ϕ}_{RED} values averaged across the transitions for different settings of the noise-to-signal ratio (N/S), *i.e.,* the ratio of the standard deviation of the noise to the standard deviation of the states. We observe that $r^{\phi}_{\text{RED}}(s, s')$ is close to 1.0 when the transitions are on the support of the expert, and it gradually decreases as the transitions drift away from the support as a result of a larger amount of added noise.

5.3.4 Illustration of the Advisor and the Learner in Grid-world

We discuss the differences between the advisor policy π_a and the learner policy with the help of an illustration in the deterministic grid-world environment (Figure 5.2). The leftmost plot shows the demonstrated expert states $\{s_1, s_2, s_3, s_4, s_5\}$. The e-MDP and the l-MDP are the same grid-world but with the following transition dynamics mismatch – while the e-MDP allows diagonal hops such that the shown expert transitions are consecutive, in the l-MDP, the agent is restricted to only horizontal and vertical movements to the nearby cells.

As discussed in Section 5.3.2, we seek to train an advisor such that the dissimilarity between the destination state achieved with the expert policy in e-MDP and that achieved with the advisor in l-MDP is minimized. The middle plot in Figure 5.2 shows the actions

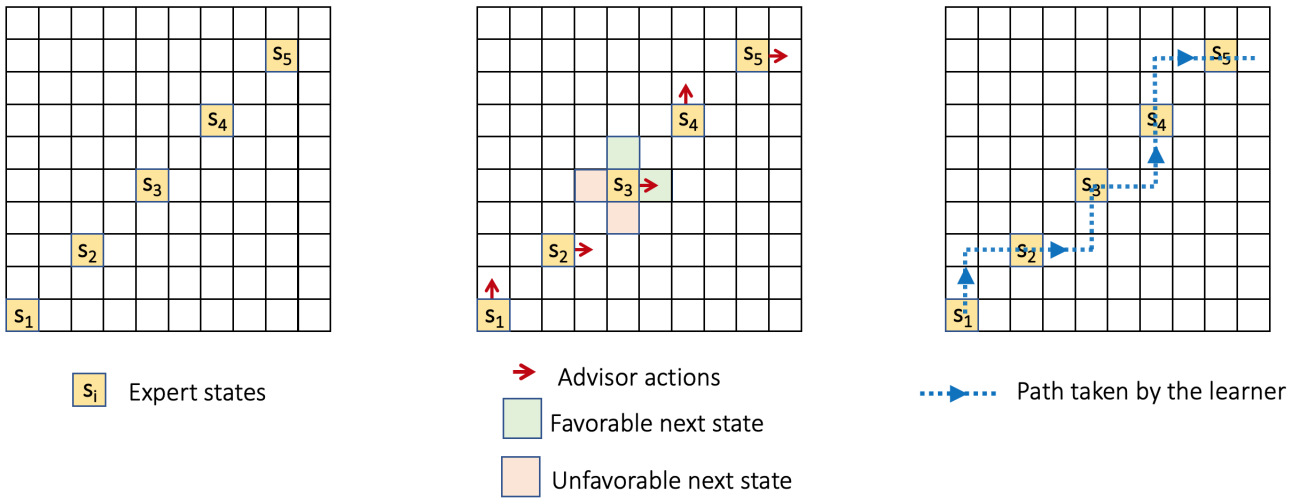


Figure 5.2: Illustration of the expert states, the actions by the advisor policy, and the path taken by the learner policy in a grid-world environment.

(marked with red arrows) that an optimal deterministic advisor may take. The advisor is trained *only* on the expert states to optimize for the next state; it is not trained for long-term behavior and thus struggles with decision-making on the non-expert states. At the state s_3, the green states are the favorable states (minimum distance to s_4), while the red states are the unfavorable states (maximum distance to s_4). In the l-MDP, the dataset $\{s_i, \pi_a(s_i)\}$ is then used to train a learner that runs in a closed feedback loop with the environment.

The learner is trained to match its stationary discounted state-action distribution with that of the generated dataset $\{s_i, \pi_a(s_i)\}$ in the l-MDP. With the visitation distribution matching, the learner reproduces the demonstrated action at the demonstrated state, while also being trained to navigate back to the manifold of the demonstrated states when it encounters non-demonstrated states, thus enabling long-horizon imitation. The advisor, that is trained to optimize just the immediate action at each expert state (similar in principle to behavior cloning), is incapable of long-horizon imitation. The rightmost plot in Figure 5.2 shows the trajectory that an optimal learner may produce.

5.3.5 Algorithm and Implementation

It is possible to use a two-stage training procedure for the overall algorithm—first, learn the advisor in l-MDP with Equation 5.10, and then use it in the IL objective laid out in Equation 5.6 to optimize for the reward network f_ω and the learner policy π. In the first stage of advisor learning, although the RED network is trained offline, computing the gradients for optimizing π_a still requires environment interaction. More importantly, since the advisor

is only trained on the expert states, the objective in Equation 5.10 requires the capability to reset the environment to the expert states in the l-MDP.

To alleviate this problem, we propose to train the advisor (π_a), the reward network (f_ω) and the learner (π) jointly, in an iterative manner, and reuse the environment interaction data generated with π for training π_a as well, using the importance sampling trick. Specifically, let β denote the parameters of π_a. Then the gradient of the objective $\mathcal{J}(\pi_a)$ is:

$$\begin{aligned}\nabla &= \nabla_\beta \Big(\mathbb{E}_{s\sim\rho_e(s)} \mathbb{E}_{a\sim\pi_a(a|s)} \mathbb{E}_{s'\sim p_l(s'|s,a)} \big[r^\phi_{\text{RED}}(s,s') \big] \Big) \\ &= \mathbb{E}_{s\sim\rho_e(s)} \mathbb{E}_{a\sim\pi_a(a|s)} \Big(\mathbb{E}_{s'\sim p_l(s'|s,a)} [r^\phi_{\text{RED}}(s,s')] . \nabla_\beta \log \pi_a(a|s) \Big) \\ &= \mathbb{E}_{\rho_\pi(s,a)} \underbrace{\frac{\rho_e(s)}{\rho_\pi(s)} \frac{\pi_a(a|s)}{\pi(a|s)}}_{\text{IS ratios}} \Big(\mathbb{E}_{s'\sim p_l(s'|s,a)} [r^\phi_{\text{RED}}(s,s')] . \nabla_\beta \log \pi_a(a|s) \Big)\end{aligned} \tag{5.11}$$

where the second equation uses the score function estimator (Kleijnen & Rubinstein, 1996), and the third employs two importance sampling ratios. Crucially, since the gradient is now computed with state-action data from $\rho_\pi(s,a)$, we no longer require environment resets. Further, we observe that approximating the inner expectation with a single sample is sufficient for our tasks.

Estimation of the importance sampling factor. In Equation 5.11, the importance sampling factor to be estimated is $\frac{\rho_e(s)}{\rho_\pi(s)} \frac{\pi_a(a|s)}{\pi(a|s)}$, which is a product of two ratios. The second ratio $\frac{\pi_a(a|s)}{\pi(a|s)}$ is easily computable since we have both the advisor policy (π_a) and the learner policy (π) as parameterized Gaussian distributions. The first ratio $\frac{\rho_e(s)}{\rho_\pi(s)}$ can be approximated by training a binary classifier to distinguish the states sampled from the distributions $\rho_e(s)$ and $\rho_\pi(s)$. Concretely, consider the objective:

$$D' = \underset{D:\mathcal{S}\to(0,1)}{\arg\max}\ \mathbb{E}_{\rho_e(s)} \big[\log D(s) \big] + \mathbb{E}_{\rho_\pi(s)} \big[\log(1 - D(s)) \big] \tag{5.12}$$

Then, we can estimate the ratio of the state distributions as $\frac{\rho_e(s)}{\rho_\pi(s)} \approx \frac{D'(s)}{1-D'(s)}$.

We abbreviate our method as *AILO*, short for Advisor-augmented Imitation Learning from Observations. Algorithm 5.1 provides an outline. We start with the offline pre-training of the RED network using a dataset of expert state transitions collected in the e-MDP. Then, we perform iterative optimization in the l-MDP where each iteration involves generating trajectories with the current learner π, followed by gradient updates to the reward network f_ω, and to the advisor and learner policies. Following Equation 5.6, the learner policy is

Algorithm 5.1: AILO (Advisor-augmented Imitation Learning from Observations)

Input: A dataset of expert state transitions $\mathcal{D}_e = \{s_e, s'_e\}$ collected in e-MDP

/* initialization and offline pre-training */

Initialize: advisor π_a, learner π, reward network $f_\omega(s, a)$, RED network $r^\phi_{\text{RED}}(s, s')$

Pre-train $r^\phi_{\text{RED}}(s, s')$ with $\mathcal{D}_e$ using the algorithm in Wang et al. (2019)

for *iter in* $\{1, \ldots, N\}$ **do**

  /* data collection */

  Roll out trajectories τ using π

  Update reward network f_ω using τ, π_a and $\mathcal{D}_e$ (Equation 5.6)

  /* update learner */

  Compute reward $f_\omega(s, a)$ for each transition in τ. Use τ to update π with MaxEnt RL

  /* update advisor */

  Compute reward $r^\phi_{\text{RED}}(s, s')$ for each transition in τ. Use τ to update π_a with Equation 5.11

end

trained with a MaxEnt RL algorithm with per-timestep entropy-regularized reward given by $f_\omega(s_t, a_t) - \alpha \log \pi(a_t|s_t)$, where α is the entropy coefficient. In our experiments, we use the clipped-ratio PPO algorithm (Schulman et al., 2017) and adaptively tune α as suggested in prior work (Haarnoja et al., 2018).

5.4 EXPERIMENTS

We evaluate the efficacy of AILO using continuous-control locomotion environments from OpenAI Gym (Brockman et al., 2016), modeled using the MuJoCo physics simulator (Todorov et al., 2012). We include a description of the tasks, the baselines, and the learning curves.

Environments. We consider five tasks - {Half-Cheetah, Walker, Hopper, Ant, Humanoid}. To create a discrepancy between the expert and the learner transition dynamics, we modify one physical property in the learner environment from the set - {density, gravity, joint-friction}. Specifically, for a task $\mathcal{T}$ from the task-set, we denote:

- $\mathcal{T}$ (heavy) $\rightarrow$ learner agent has $2\times$ the mass of the expert agent
- $\mathcal{T}$ (light) $\rightarrow$ gravity in the learner's environment is half the value in the expert's

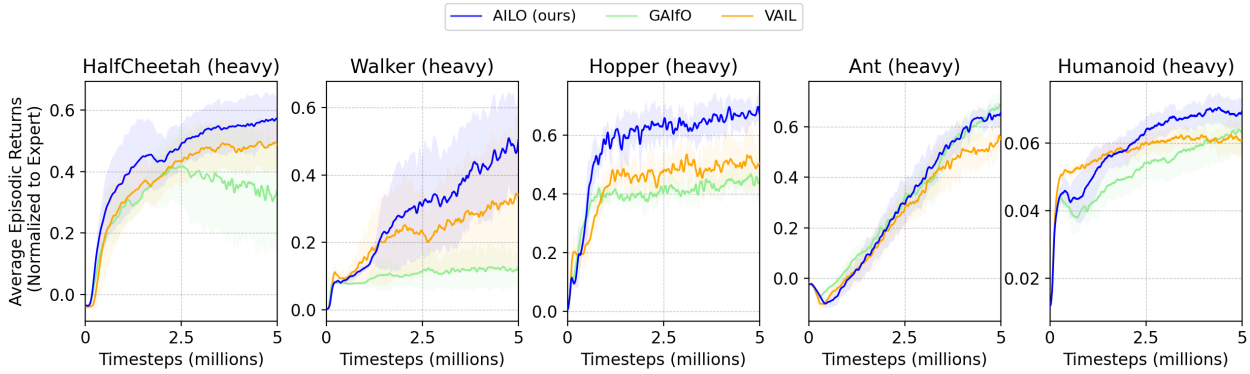
- $\mathcal{T}$ (drag) → friction coefficient for all the joints in the learner is 2× the value in the expert

Baselines. We contrast AILO with two baselines – a.) *GAIfO* strives to minimize the JS divergence between the state-transition distributions and is briefly described in §5.2.1; b.) *VAIL* (Peng et al., 2018) applies the concept of variational information bottleneck (Alemi et al., 2016) to the GAIL discriminator for improved regularization and has been successfully used for ILO with motion-capture data. To limit the effect of confounding factors during comparison, we share modules across AILO and the two baselines to the best of our ability. Concretely, all discriminator/reward networks use the same architecture and the gradient-penalty regularization (Mescheder et al., 2018) and thus exhibit the same reward biases. Furthermore, the MaxEnt-RL PPO module is the same for all algorithms.

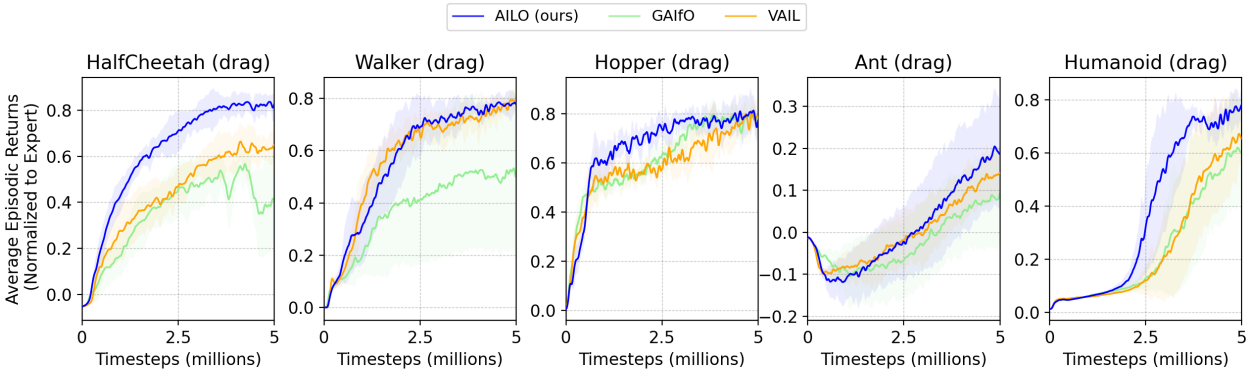
Performance. Figures 5.3a–5.3c plot the learning curves for all the algorithms across the different tasks (heavy, drag, light). We show the average episodic returns achieved by the learner in the l-MDP, normalized to the returns achieved by the expert in e-MDP. The plots include the mean and the standard deviation of returns over 6 runs with random seeds.

We observe that AILO provides a noticeable improvement in learning efficiency in several situations, such as Half-Cheetah (heavy, drag, light), Walker (heavy), Hopper (heavy), Ant (drag), Humanoid (drag, light); while being comparable to the *best* baseline in other cases. For GAIfO, we find that it does not learn any useful skill for Walker (heavy) and exhibits training instability for Half-Cheetah (heavy, drag). We attribute this to the difficulty of matching the state-transition distributions across different dynamics. VAIL, which matches state distributions, instead of state-transition distributions, is a stronger baseline and works well in several cases. Lastly, we highlight a few failure modes of AILO – in Ant (light) and Humanoid (heavy), we note that AILO (and baselines) do not make much progress towards imitating the demonstrated behavior within our time budget, motivating the need for future enhancements to enable efficient skill transfer in these challenging setups.

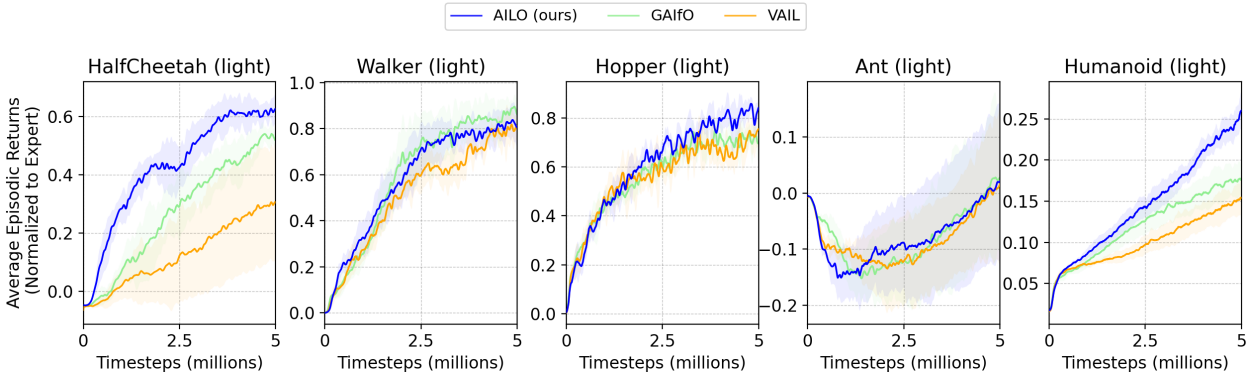
Ablation on the degree of dynamics mismatch. For the empirical results in Figure 5.3, the variation in mass, gravity, or friction, between the e-MDP and the l-MDP was kept at a constant factor. In Figure 5.4, we consider the Walker task and plot the systematic degradation in the performance of AILO and the baseline GAIfO, as the l-MDP parameters drift further from the e-MDP parameters. We show the final average episodic returns achieved by the learner in the l-MDP, normalized to the returns achieved by the expert in e-MDP, as a function of the degree of dynamics mismatch. In the plots, the expert parameters (mass, friction) are kept fixed and the learner parameters are varied such that



(a) Performance on $\mathcal{T}$ (heavy) environments



(b) Performance on $\mathcal{T}$ (drag) environments



(c) Performance on $\mathcal{T}$ (light) environments

Figure 5.3: Learning curves for AILO and the baselines for different environments with discrepancy in dynamics

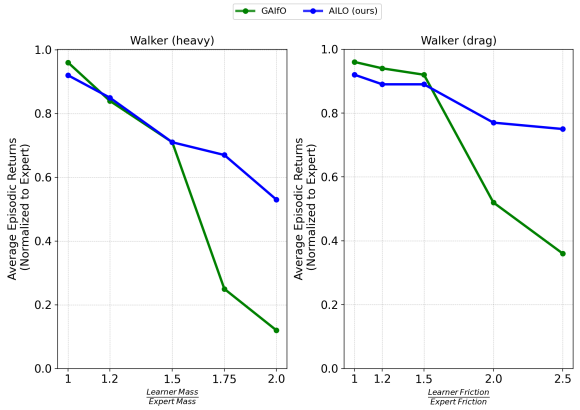


Figure 5.4: Results with different amount of dynamics mismatch

		AILO-advisor	AILO-learner
HalfCheetah	(H)	0.19	0.57
	(D)	0.27	0.82
	(L)	0.18	0.63
Walker	(H)	0.07	0.50
	(D)	0.14	0.78
	(L)	0.17	0.81
Hopper	(H)	0.53	0.67
	(D)	0.48	0.77
	(L)	0.66	0.84

Table 5.2: Normalized average episodic returns achieved by the advisor and the learner in the l-MDP.

the ratio increases from a starting value of 1 (no mismatch). We observe that although imitation naturally becomes more challenging as the dynamics become more different, the degradation with AILO is more graceful compared to GAIfO.

5.4.1 Comparing the Advisor and the Learner Performance

In Table 5.2, we report the average episodic returns achieved by the advisor and the learner in the l-MDP at the end of the training, normalized to the returns achieved by the expert in e-MDP. We show data for three environments and all the types of transition dynamics mismatch considered in the work – *Heavy* (H), *Drag* (D), *Light* (L). We note that the learner outperforms the advisor by a substantial margin for all the scenarios. This is because, unlike the learner, the advisor is not optimized for long-horizon performance.

5.5 RELATED WORK AND CONCLUSION

There is a vast amount of literature on IL since it is a powerful framework to train agents to perform complex behaviors without a reward specification. ILO, where no expert action labels are available, presents several benefits as well as some unique challenges, and thus, has garnered significant attention from the community in recent times (Torabi et al., 2018,; Liu et al., 2018; Edwards et al., 2019; Sun et al., 2019; Yang et al., 2019; Zhu et al., 2021). ILO methods adapted from GAIL have been proposed for one-shot imitation of diverse behaviors (Wang et al., 2017), training policies to generate human-like movement patterns using motion-capture data (Peng et al., 2018; Merel et al., 2017), and for locomotion control

from raw visual data (Torabi et al., 2018). Arumugam et al. (2020) introduce a framework that casts adversarial ILO as f-divergence minimization and provide insights on the design decisions that impact performance.

Several approaches have been proposed to handle the differences between the expert and the learner environments in terms of viewpoints, visual appearances, presence of distractors, and morphology changes (Stadie et al., 2017; Gupta et al., 2017; Liu et al., 2018; Sermanet et al., 2018). They typically proceed by learning a domain-invariant representation and matching features in that space. Learning such a representation is not required in our setup since the state-space is shared between the l-MDP and the e-MDP. Methods for robustness to shifts in the action-space and the dynamics model have also been researched. Zolna et al. (2019) propose to match state-pair distributions of the expert and the learner, where the states in a pair are sampled with random time gaps, rather than being consecutive. Liu et al. (2019) learn an inverse action model to predict deterministic actions in the l-MDP that could generate expert-like state transitions and use it to regularize policy updates. Gangwani & Peng (2020) filter the trajectories generated in l-MDP based on their similarity to the states in the expert dataset and perform (self-) imitation on these. The key methodological difference between these methods and our work is that we learn an intermediary stochastic policy (advisor) in the l-MDP by bringing its state-conditional next-state distribution closer to that of the expert, and propose an instantiation of this idea using an approximation based on support estimation.

5.5.1 Conclusion

In this work, we present AILO, our algorithm for imitation learning from observations under transition model disparity between the expert and the learner environments. Rather than directly matching the state-transition distributions across environments, we train an intermediary policy (advisor) in the learner environment and use it as a surrogate expert for the learner. Towards learning an advisor that acts as an effective surrogate, we propose to minimize the cross-entropy distance between the state-conditional next-state distributions of the advisor and the expert. To realize this idea into a scalable ILO algorithm, we leverage prior work on support estimation (RED). Our experiments on five MuJoCo locomotion tasks with different types of dynamics discrepancies show that AILO compares favorably to the baseline ILO methods in many cases.

CHAPTER 6: LEARNING AGENTS WITH QUALITY AND DIVERSITY

6.1 INTRODUCTION

The goal in Reinforcement Learning (RL) is to learn agents that maximize long-term environmental rewards. Deep RL, which uses deep neural networks as function approximators for the policy and value-functions, has achieved outstanding results on a wide variety of sequential decision making problems, with the barometer of success usually being the total returns accumulated by the final policy. Due to the intrinsic nature of direct reward maximization, seldom is the focus on how the *behavioral characteristics* of the trained agent compare with the other possible behaviors in the solution space. For instance, consider the robotic manipulator arm in Figure 6.1a and the peg-insertion task. Though the task description is simple, for a sufficiently flexible arm, there are numerous ways (positions of the joints and the end-effector) to insert the peg in the hole (Figure 6.1b). For reasons argued below, it is beneficial to learn these varied behaviors rather than aiming for the single most efficient solution dictated by the reward function. Quality-Diversity (QD) algorithms (Pugh et al., 2016; Cully & Demiris, 2017) are prominent in the Neuroevolution literature as a means to generate many diverse behavioral *niches*, while ensuring that each niche is populated with individuals of the highest possible quality for that niche. When applied to RL, QD offers a principled approach for learning policies that are diverse, yet achieve high returns (Mouret & Clune, 2015; Conti et al., 2017).

Prior works have examined the benefits of uncovering diversity in how the task can be solved (Hong et al., 2018; Eysenbach et al., 2018; Liu et al., 2017). In these, an explicit diversity-maximization objective is incorporated into the RL algorithm to facilitate the learning of diverse *skills*. There are several important benefits of training a population of agents with diverse skills. Firstly, this is an efficient exploration strategy in sparse-reward environments as the agents can collectively achieve much wider coverage of the state-space, while reducing the susceptibility of RL to local optimal solutions caused by deceptive rewards (Conti et al., 2017; Gangwani et al., 2018). Secondly, the acquired skills could be leveraged for accelerated learning in downstream tasks, for example, by composing the skills to solve long-horizon tasks via hierarchical RL (Florensa et al., 2017; Eysenbach et al., 2018). Diversity also helps in the transfer learning of RL policies across environments that may have discrepancies such as system dynamics mismatch. Having a repertoire of skills is useful when knowledege transfer is done to a target environment that has constraints on the set of feasible behaviors (Cully et al., 2015).

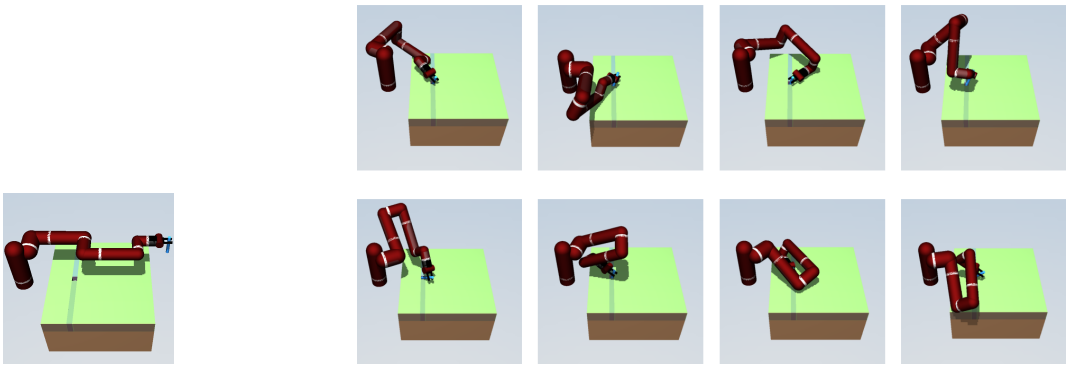
A policy π is characterized by its occupancy measure ρ_π (Puterman, 1994), which is the stationary distribution over the state-action pairs that π encounters when navigating the environment. Given two policies π, β, the ratio of their stationary distributions $\zeta = \rho_\pi / \rho_\beta$ is a well-studied quantity in RL. Estimates of the distribution ratio are useful for off-policy evaluation (Precup, 2000; Thomas & Brunskill, 2016) (where the goal is to evaluate the performance of π using fixed data generated from β), policy optimization (Sutton et al., 2016; Liu et al., 2019) and off-policy imitation learning (Kostrikov et al., 2019). In this work, we examine the use of distribution ratio estimators for learning a diverse policy ensemble with high returns (a QD ensemble). We build on the approach introduced by Liu et al. (2017). Using Stein variational gradient descent (SVGD) (Liu & Wang, 2016) as the inference method, the authors construct an update rule that includes the policy-gradient on the environmental rewards (for high quality) and a kernel-induced repulsive force gradient (for high diversity). This kernel-based algorithm is naturally impacted by the choice of the kernel. We begin with generalizing the Stein variational policy gradient (SVPG) objective (Liu et al., 2017) by using as kernels the negative exponents of an f-divergence between the stationary distributions of two policies, and discuss key properties such as positive-definiteness of kernels. For kernels based on the Jenson-Shannon and Symmetric Kullback-Leibler divergences, we show how the complete SVPG gradient can be recast in terms of the *ratio of the stationary distributions* (ζ) between policies. Then, to estimate these ratios, and hence the SVPG gradient, we study three recently proposed distribution ratio estimators for off-policy evaluation and imitation learning. These are *DualDICE* (Nachum et al., 2019), *ValueDICE* (Kostrikov et al., 2019) and *GenDICE* (Zhang et al., 2020). Additionally, we describe a fourth estimator based on Noise-Constrastive Estimation (Gutmann & Hyvärinen, 2010).

We perform experiments on various tasks to get a measure of the effectiveness of our proposed approach in generating diverse behaviors with high returns. We also evaluate on tasks with deceptive rewards and those which lack an external reward signal to further illuminate the benefits of QD.

6.2 VARIATIONAL INFERENCE FOR THE QD OBJECTIVE

6.2.1 Notations

The environment is modeled as an infinite-horizon, discrete-time Markov Decision Process (MDP), represented by the tuple ($\mathcal{S}$, $\mathcal{A}$, μ_0, r, p, γ), where $\mathcal{S}$ is the state-space, $\mathcal{A}$ is the action-space, $\gamma \in [0, 1)$ is the discount factor, and μ_0 denotes the initial state distribution. Given an action a_t sampled from a stochastic policy $\pi_\theta(a_t|s_t)$, the next state is sampled from



(a) (b)

Figure 6.1: (a) MuJoCo model of a 7 DOF arm based on the Sawyer robot, inspired by Chen et al. (2018); (b) Policies that achieve the peg-insertion task in different ways. These policies are sampled from a single ensemble trained with the algorithm QD-*DualDICE*-JS (explained in Section 6.4).

the transition dynamics distribution, $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$, and the agent receives a reward $r(s_t, a_t)$ determined by the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The RL objective is to maximize the expected discounted sum of rewards, $\eta(\pi) = (1-\gamma)\mathbb{E}_{\mu_0,p,\pi}\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\big]$.

Distribution Ratio (ζ)**.** The occupancy measure (Puterman, 1994), or the stationary discounted state-action visitation distribution of the policy π is defined as $\rho_\pi(s, a) = (1-\gamma)\pi(a|s)\sum_{t=0}^{\infty}\gamma^t\mathbb{P}(s_t{=}s|\pi)$, where $\mathbb{P}(s_t{=}s|\pi)$ is the probability of being in state s at time t, when starting in state $s_0 \sim \mu_0$ and using π thereafter. The stationary distribution [1] affords a convenient rewriting of the expected policy return as $\eta(\pi) = \mathbb{E}_{\rho_\pi}[r(s, a)]$, and the gradient is provided by the policy gradient theorem (Sutton et al., 2000) as $\nabla_\theta\eta(\pi) = \mathbb{E}_{\rho_\pi}\big[\nabla_\theta \log \pi_\theta(a|s)Q^\pi(s, a)\big]$, where Q^π is the state-action value function. For two policies π_i and π_j, we denote the ratio of their stationary distributions by $\zeta_{ij}(s, a) = \rho_{\pi_i}(s, a)/\rho_{\pi_j}(s, a)$. This ratio is widely applicable for off-policy evaluation as it enables estimating the expected returns of π_i using a fixed dataset $\mathcal{D}$ of transitions generated from a different behavioral policy π_j, since $\eta(\pi_i) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[\zeta_{ij}(s, a)r(s, a)]$, where $\mathcal{D}$ is an empirical estimate of ρ_{π_j}.

6.2.2 Policy Search with SVGD

QD when applied to policy search entails learning multiple policies that all accumulate high environmental rewards during an episode, but the agents accomplish this using diversified strategies, such as navigating dissimilar regions of the state-action space. Formally, policy

[1] Throughout, stationary *discounted* distribution is shorthanded with stationary distribution for brevity

Name of f-divergence	Formula $D_f(\mathrm{P}, \mathrm{Q})$	Generator $f(u)$ with $f(1) = 0$	Is the kernel $e^{-D_f(P,Q)/T}$ PD?
Jenson-Shannon	$\int_x \frac{p(x)}{2} \log \frac{2p(x)}{p(x)+q(x)} + \frac{q(x)}{2} \log \frac{2q(x)}{p(x)+q(x)} dx$	$\frac{u}{2} \log u - \frac{(1+u)}{2} \log \frac{1+u}{2}$	Yes
Triangular Discrimination	$\int_x \frac{(p(x)-q(x))^2}{p(x)+q(x)} dx$	$\frac{(u-1)^2}{u+1}$	Yes
Squared Hellinger	$\int_x (\sqrt{p(x)} - \sqrt{q(x)})^2 dx$	$(\sqrt{u} - 1)^2$	Yes
Total Variation	$\frac{1}{2} \int_x \lvert p(x) - q(x) \rvert dx$	$\frac{1}{2} \lvert u - 1 \rvert$	Yes
Kullback-Leibler	$\int_x p(x) \log \frac{p(x)}{q(x)} dx$	$-\log u$	No
Reverse Kullback-Leibler	$\int_x q(x) \log \frac{q(x)}{p(x)} dx$	$u \log u$	No
Symmetric Kullback-Leibler	$\int_x p(x) \log \frac{p(x)}{q(x)} + q(x) \log \frac{q(x)}{p(x)} dx$	$(u-1) \log u$	No

Table 6.1: f-divergences and positive-definiteness of the negative exponential kernels.

search with QD could be defined as learning a *distribution* over the policy parameters (θ) that maximizes the RL-objective in expectation, while maintaining a high-entropy ($\mathcal{H}$) in the parameter-space:

$$\max_q \mathbb{E}_{\theta \sim q}[\eta(\theta)] + \mathcal{H}(q) \; ; \qquad \mathcal{H}(q) = \mathbb{E}_{\theta \sim q}[-\log q(\theta)] \tag{6.1}$$

Solving the objective in Equation 6.1 analytically yields the following energy-based optimal parameter distribution: $q^*(\theta) = \exp(\eta(\theta))/Z_{q^*}$, where Z_{q^*} is the normalization constant. Let $p(\theta)$ define a trainable distribution over the policy parameters that we seek to optimize to be close (*w.r.t.* the KL-divergence) to the target distribution q^*. Representing $p(\theta)$ with a mixture of delta distributions, the variational objective is:

$$\min_p D_{\mathrm{KL}}\big[p \;||\; \exp(\eta(\theta))/Z_{q^*}\big] \; ; \qquad p(\theta) = \frac{1}{n} \sum_{i=1}^{n} \delta(\theta = \theta_i) \tag{6.2}$$

Here $\{\theta_i\}_1^n$ denotes a policy ensemble with n discrete policies that constitute the p distribution. Stein variational gradient descent (SVGD; Liu & Wang (2016)) provides an efficient solution to obtain an approximate gradient on the p distribution. Suppose we perturb each policy θ_i with $\Delta\theta_i$ such that the induced KL between p and q is reduced. The optimal perturbation direction, in the unit ball of a reproducing kernel Hilbert space associated with a kernel function k, that maximally decreases the KL is given by (Liu & Wang, 2016):

$$\Delta\theta = \mathbb{E}_{\theta' \sim p}\big[\nabla_{\theta'} \log q^*(\theta') k(\theta', \theta) + \nabla_{\theta'} k(\theta', \theta)\big] \tag{6.3}$$

Using this result and the energy-based form of the target distribution q^*, SVPG (Liu et al., 2017) iteratively updates the policies with the following rule to learn a policy ensemble with

QD behavior:

$$\theta_i \leftarrow \theta_i + \epsilon\Delta\theta_i, \qquad \Delta\theta_i = \frac{1}{n}\sum_{j=1}^{n}\Big[\underbrace{\nabla_{\theta_j}\eta(\pi_{\theta_j})k(\theta_j,\theta_i)}_{\text{Quality-enforcing}} + \underbrace{\nabla_{\theta_j}k(\theta_j,\theta_i)}_{\text{Diversity-enforcing}}\Big] \tag{6.4}$$

6.2.3 Negative Exponents of f-divergences as Kernels

The positive definite (PD) kernel function k in Equation 6.4 is an algorithmic design choice. There are two considerations. It should be possible to efficiently compute $k(\theta_j, \theta_i)$ for any two policies $(\pi_{\theta_j}, \pi_{\theta_i})$ as well as its gradient *w.r.t.* the policy parameters; and the function should be sufficiently expressive to capture the complex interactions between policy behaviors. Liu et al. (2017) employ a Gaussian RBF kernel $k(\theta_j, \theta_i) = \exp(-\|\theta_j - \theta_i\|_2^2/h)$, with a dynamically adapted bandwidth h. Gangwani et al. (2018) suggest replacing the Euclidean distance in the parameter space with a statistical distance in the stationary distribution space, and use $k(\theta_j, \theta_i) = \exp(-D_{\text{JS}}(\rho_{\pi_{\theta_j}}, \rho_{\pi_{\theta_i}})/T)$, where D_{JS} is the Jenson-Shannon divergence and T is the temperature. D_{JS} is a member of a broader class of divergences known as Ali-Silvey distances or f-divergences (Ali & Silvey, 1966). Given two distributions with continuous densities $p(x)$ and $q(x)$ over the support $\mathcal{X}$, the f-divergence between them is defined as:

$$D_f(p \,||\, q) = \int_{\mathcal{X}} q(x) f\left(\frac{p(x)}{q(x)}\right) dx \tag{6.5}$$

where $f : \mathbb{R}_+ \to \mathbb{R}$ is a convex, lower-semicontinuous function such that $f(1) = 0$. Different choices for the function f recover the well-known divergences. Although generalizing the kernel function as $k_f(\theta_j, \theta_i) = \exp(-D_f(\rho_{\pi_{\theta_j}}, \rho_{\pi_{\theta_i}})/T)$ may seem like a natural extension, for some f-divergences, $k_f(\theta_j, \theta_i)$ is not PD, and hence not a proper kernel from a theoretical standpoint. For instance, while k_{JS} is PD, kernels with other divergences commonly used for policy learning (KL, Reverse-KL) are not. Table 6.1 provides details on the various divergences that define PD and non-PD kernels after negative exponentiation. The first four divergences in Table 6.1 are squared metrics (*i.e.* $\sqrt{D_f}$ is a true metric) and the proof of positive-definiteness of the corresponding kernels k_f is provided in Hein & Bousquet (2004). Inserting $k_f(\theta_j, \theta_i)$ in Equation 6.4, the SVPG gradient becomes:

$$\Delta\theta_i = \frac{1}{n}\sum_{j=1}^{n} \exp\Big(-\underbrace{D_f(\rho_{\pi_{\theta_j}}, \rho_{\pi_{\theta_i}})}_{\text{Divergence value}}/T\Big)\Big[\underbrace{\nabla_{\theta_j}\eta(\pi_{\theta_j})}_{\text{Policy gradient}} - \frac{1}{T}\underbrace{\nabla_{\theta_j} D_f(\rho_{\pi_{\theta_j}}, \rho_{\pi_{\theta_i}})}_{\text{Divergence gradient}}\Big] \tag{6.6}$$

This provides a general framework to evaluate the SVPG gradient for learning a QD

policy ensemble. Depending on the f-divergence and the method for estimating its value and gradient, several approaches are possible, a few of which we will discuss. We use the shorthand notation ρ_i for the stationary distribution of the policy π_{θ_i}. Then $\zeta_{ij}(s,a) = \rho_i(s,a)/\rho_j(s,a)$ is the distribution ratio for two given policies, and would be the pivotal quantity in the exposition that follows. Next, we rewrite two kernels (and their gradient *w.r.t.* the policy parameters) in terms of ζ, before elucidating several methods to estimate ζ for use in a practical RL algorithm to generate a QD policy ensemble.

The k_{JS} and k_{KLS} kernels. While k_{JS} is a PD kernel, k_{KLS} is not since $\sqrt{D_{\text{KLS}}}$ is not a metric as it does not satisfy the triangle inequality. Although positive-definiteness is desirable, non-PD kernels may yet achieve good performance in practice, as shown in Moreno et al. (2004), where SVM classification with a non-PD kernel leads to better accuracy than provably PD kernels. Both k_{JS} and k_{KLS} afford the benefit that the divergence value and gradient (in Equation 6.6) can be evaluated in terms of the distribution ratio ζ_{ij}. Using the definitions from Table 6.1, we express D_{JS} and D_{KLS} as:

$$\begin{aligned} D_{\text{JS}}(\rho_i, \rho_j) &= \frac{1}{2}\mathbb{E}_{\rho_i(s,a)} \log \frac{\zeta_{ij}(s,a)}{1+\zeta_{ij}(s,a)} + \frac{1}{2}\mathbb{E}_{\rho_j(s,a)} \log \frac{1}{1+\zeta_{ij}(s,a)} + \log 2 \\ D_{\text{KLS}}(\rho_i, \rho_j) &= \mathbb{E}_{\rho_i(s,a)} \log \zeta_{ij}(s,a) - \mathbb{E}_{\rho_j(s,a)} \log \zeta_{ij}(s,a) \end{aligned} \tag{6.7}$$

The SVPG gradient involves the gradient of the f-divergence *w.r.t.* the policy parameters (θ). For D_{JS} and D_{KLS}, the gradient can be written using ζ as follows:

$$\nabla_{\theta_j} D_{\text{JS}} = \nabla_{\theta_j} \mathbb{E}_{\rho_j} \underbrace{-(1/2)\log[1+\zeta_{ij}(s,a)]}_{r(s,a)}; \quad \nabla_{\theta_j} D_{\text{KLS}} = \nabla_{\theta_j} \mathbb{E}_{\rho_j} \underbrace{[-\zeta_{ij}(s,a) - \log \zeta_{ij}(s,a)]}_{r(s,a)} \tag{6.8}$$

In practice, we can estimate these gradients with the policy-gradient theorem (Sutton et al., 2000), using the appropriate term as the reward function (noted as $r(s,a)$ above). It is thus evident that a reasonable estimation of the distribution ratio yields a good approximation of the SVPG gradient (Equation 6.6), which could then be applied to the policy parameters to learn a QD ensemble. We now discuss methods to estimate ζ efficiently from samples.

6.3 ESTIMATING DISTRIBUTION RATIOS

We start with Noise-Constrastive Estimation (NCE) (Gutmann & Hyvärinen, 2010) which has found wide applicability in representation learning, natural language processing and image synthesis, among others. We then examine three distribution ratio estimators –

DualDICE (Nachum et al., 2019) and *GenDICE* (Zhang et al., 2020) were recently proposed for behavior-agnostic off-policy evaluation, and *ValueDICE* (Kostrikov et al., 2019) enables imitating expert trajectories without requiring additional policy rollouts in the environment.

NCE. It provides a method to learn an estimator $\widetilde{\rho}_i(s, a; \omega)$ for the stationary distribution of any policy π_i in the ensemble. NCE uses a noise distribution $p_N(s, a)$ and frames the following binary classification objective:

$$\max_{\omega} \mathbb{E}_{\rho_i} \log \frac{\widetilde{\rho}_i(s, a; \omega)}{\widetilde{\rho}_i(s, a; \omega) + p_N(s, a)} + \mathbb{E}_{p_N} \log \frac{p_N(s, a)}{\widetilde{\rho}_i(s, a; \omega) + p_N(s, a)} \tag{6.9}$$

Gutmann & Hyvärinen (2010) show that under mild assumption on the noise distribution, $\widetilde{\rho}_i(\cdot; \omega)$ converges to the true density ρ_i in the limit of infinite amount of samples. They further note that for practical efficiency, it is desirable to select a noise distribution that is easy to sample from, and that is not too far from the true unknown data distribution ρ_i. Consequently, for learning the estimator for policy i, we use a uniform mixture of stationary distributions of the remaining $(n-1)$ policies in the ensemble as the constrastive noise, *i.e.*, $p_N(s, a) = (1/(n-1)) \sum_{j \neq i} \rho_j(s, a)$. The distribution ratio for a pair of policies can then be computed as $\zeta_{ij}(s, a) = \widetilde{\rho}_i(s, a)/\widetilde{\rho}_j(s, a)$.

DualDICE. Nachum et al. (2019) propose a convex optimization problem that gives the distribution ratio as its optimal solution:

$$\zeta_{ij} = \arg\min_{x: \mathcal{S} \times \mathcal{A} \to \mathbb{R}} \frac{1}{2} \mathbb{E}_{(s,a) \sim \rho_j}[x(s, a)^2] - \mathbb{E}_{(s,a) \sim \rho_i}[x(s, a)] \tag{6.10}$$

The expression is then simplified with the following *change-of-variables* trick. Define a variable $\nu(s, a)$ and the operator $\mathcal{B}^{\pi_i} \nu(s, a) = \gamma \mathbb{E}_{s' \sim p(\cdot|s,a), a' \sim \pi_i(s')}[\nu(s', a')]$. Using $x(s, a) = \nu(s, a) - \mathcal{B}^{\pi_i} \nu(s, a)$, the second expectation in Equation 6.10 telescopes and conveniently reduces into an expectation over the initial states. The transformed objective is:

$$\min_{\nu: \mathcal{S} \times \mathcal{A} \to \mathbb{R}} \frac{1}{2} \mathbb{E}_{(s,a) \sim \rho_j}[(\nu - \mathcal{B}^{\pi_i} \nu)(s, a)^2] - (1 - \gamma) \mathbb{E}_{\substack{s_0 \sim \mu_0 \\ a_0 \sim \pi_i(s_0)}}[\nu(s_0, a_0)] \tag{6.11}$$

Given an optimal solution ν^* for this equation, the distribution ratio is recovered with $\zeta_{ij}(s, a) = (\nu^* - \mathcal{B}^{\pi_i} \nu^*)(s, a)$. Further, to alleviate the bias in the sample-based Monte-Carlo estimate of the gradient, Nachum et al. (2019) suggest the use of Fenchel conjugates.

Fenchel duality provides that $\frac{1}{2} x^2 = \max_g gx - \frac{1}{2} g^2$ for a scalar $g \in \mathbb{R}$. Nachum et al. (2019) rewrite the quadratic (first) term in the objective using this maximization and use

the interchangeability principle (Shapiro et al., 2014) to replace the inner max over scalar g to a max over functions $g : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Given the definition of the $\mathcal{B}^{\pi_i}$ operator, this yields the min-max objective:

$$\min_{\nu} \max_{g} J(\nu, g) = \mathbb{E}_{\substack{(s,a)\sim\rho_j, s'\sim p(\cdot|s,a) \\ a'\sim\pi_i(s')}} \left[\big(\nu(s,a) - \gamma\nu(s',a')\big) g(s,a) - \frac{g(s,a)^2}{2} \right] - (1-\gamma)\mathbb{E}_{\substack{s_0\sim\mu_0 \\ a_0\sim\pi_i(s_0)}} [\nu(s_0,a_0)] \tag{6.12}$$

The distribution ratio is obtained from the saddle-point solution (ν^*, g^*) using the following equivalence, $\zeta_{ij}(s,a) = g^*(s,a) = (\nu^* - \mathcal{B}^{\pi_i}\nu^*)(s,a)$.

ValueDICE. The Donsker-Varadhan representation (Donsker & Varadhan, 1983) of the KL-divergence is given by:

$$D_{\text{KL}}(\rho_i || \rho_j) = \sup_{x:\mathcal{S}\times\mathcal{A}\to\mathbb{R}} \mathbb{E}_{(s,a)\sim\rho_i}[x(s,a)] - \log \mathbb{E}_{(s,a)\sim\rho_j}[e^{x(s,a)}] \tag{6.13}$$

In *ValueDICE* (Kostrikov et al., 2019), the authors use the fact that the optimality in the above equation is achieved at $x^*(s,a) = \log \zeta_{ij}(s,a) + C$, for some constant $C \in \mathbb{R}$. Therefore, a method to obtain ζ_{ij} is to first solve for x^* (written as a minimization):

$$x^* = \arg\min_{x:\mathcal{S}\times\mathcal{A}\to\mathbb{R}} \log \mathbb{E}_{(s,a)\sim\rho_j}[e^{x(s,a)}] - \mathbb{E}_{(s,a)\sim\rho_i}[x(s,a)] \tag{6.14}$$

With the same *change-of-variables* trick from *DualDICE*, $x(s,a) = \nu(s,a) - \mathcal{B}^{\pi_i}\nu(s,a)$, the second expectation over $\rho_i(s,a)$ is transformed into an expectation over the initial states:

$$\min_{\nu:\mathcal{S}\times\mathcal{A}\to\mathbb{R}} \log \mathbb{E}_{(s,a)\sim\rho_j}[e^{\nu(s,a) - \mathcal{B}^{\pi_i}\nu(s,a)}] - (1-\gamma)\mathbb{E}_{\substack{s_0\sim\mu_0 \\ a_0\sim\pi_i(s_0)}} [\nu(s_0,a_0)] \tag{6.15}$$

Different from *DualDICE*, *ValueDICE* avoids the min-max saddle-point optimization by eschewing the use of Fenchel conjugates to remove the bias in the sample-based gradient. The log distribution ratio is calculated (up to a constant shift) from ν^* as, $\log \zeta_{ij}(s,a) = \nu^*(s,a) - \mathcal{B}^{\pi_i}\nu^*(s,a)$.

GenDICE. It is known that the stationary distribution for a policy π_i satisfies the following Bellman flow constraint:

$$\rho_i(s',a') = (1-\gamma)\mu_0(s')\pi_i(a'|s') + \gamma \int \pi_i(a'|s')p(s'|s,a)\rho_i(s,a)dsda; \quad \forall (s',a') \in \mathcal{S}\times\mathcal{A} \tag{6.16}$$

This could be re-written using the distribution ratio ζ_{ij} as:

$$\rho_j(s',a')\zeta_{ij}(s',a') = \underbrace{(1-\gamma)\mu_0(s')\pi_i(a'|s') + \gamma \int \pi_i(a'|s')p(s'|s,a)\zeta_{ij}(s,a)\rho_j(s,a)dsda}_{\mathcal{T}_{(\pi_i,\rho_j)}\circ\zeta_{ij}} \tag{6.17}$$

Zhang et al. (2020) parameterize ζ_θ and suggest to estimate it by minimizing the f-divergence between the distributions (with support on $\mathcal{S}\times\mathcal{A}$) on the two sides of Equation 6.17, namely $\rho_j.\zeta_\theta$ and $\mathcal{T}_{(\pi_i,\rho_j)}\circ\zeta_\theta$, where the notation $\mathcal{T}_{(\pi_i,\rho_j)}$ denotes the distribution operator on the RHS in Equation 6.17. The objective, which further includes a penalty regularizer on ζ_θ to prevent degenerate solutions, is:

$$\zeta_{ij} = \arg\min_\theta D_f\big(\mathcal{T}_{(\pi_i,\rho_j)}\circ\zeta_\theta \,||\, \rho_j.\zeta_\theta\big) + \frac{\lambda}{2}\big(\mathbb{E}_{\rho_j}[\zeta_\theta]-1\big)^2 \tag{6.18}$$

Similar to *DualDICE*, Fenchel conjugates are used to obtain unbiased gradient estimates, resulting in a min-max saddle-point optimization. Let $g:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ be a function. The f-divergence could be substituted with its variational representation (Nguyen et al., 2010) which involves the Fenchel conjugate (f^*) of the f function in D_f:

$$D_f\big(\mathcal{T}_{(\pi_i,\rho_j)}\circ\zeta_\theta \,||\, \rho_j.\zeta_\theta\big) = \max_g \; \mathbb{E}_{\mathcal{T}_{(\pi_i,\rho_j)}\circ\zeta_\theta}[g(s,a)] - \mathbb{E}_{\rho_j.\zeta_\theta}[f^*(g(s,a))] \tag{6.19}$$

This expression can be simplified by using the definition of the $\mathcal{T}_{(\pi_i,\rho_j)}$ operator. Furthermore, since Fenchel duality provides that $\frac{1}{2}x^2 = \max_u ux - \frac{1}{2}u^2$, the quadratic penalty regularization can also be written in form of a max over a scalar variable $u\in\mathbb{R}$. This yields the min-max objective:

$$\begin{aligned}\min_\theta\max_{g,u} J(\theta,g,u) = (1-\gamma)\mathbb{E}_{\mu_0(s)\pi_i(a|s)}[g(s,a)] + \gamma\mathbb{E}_{\substack{(s,a)\sim\rho_j, s'\sim p(\cdot|s,a)\\ a'\sim\pi_i(s')}}[\zeta_\theta(s,a)g(s',a')] \\ - \mathbb{E}_{(s,a)\sim\rho_j}[\zeta_\theta(s,a)f^*(g(s,a))] + \lambda\big(\mathbb{E}_{\rho_j}[u\zeta_\theta(s,a)-u] - \frac{u^2}{2}\big)\end{aligned} \tag{6.20}$$

For a practical instantiation, Zhang et al. (2020) suggest the χ^2 divergence, which is an f-divergence with $f(x)=(x-1)^2$ and $f^*(x)=x+\frac{x^2}{4}$.

Overall Algorithm. We summarize our complete approach for training a QD policy ensemble in Algorithm 6.1. In each iteration, we sample transitions in the environment using the policies in the ensemble and update the networks that facilitate estimation of the distri-

Algorithm 6.1: Pseudo code for learning a QD ensemble

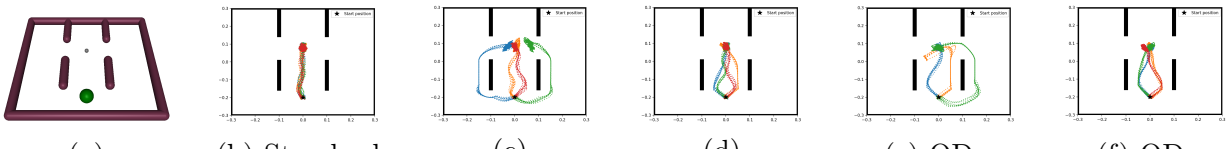
Initialize policy ensemble $\{\pi_i\}_1^n$
Initialize networks to estimate ζ_{ij} ▷ Parameterization depends on the method (Section 6.3)
for *each iteration* **do**
    Sample rollouts with $\pi_i \ \forall i$
    Update all ζ_{ij} estimation networks ▷ Objective depends on the method (Section 6.3)
    Use ζ_{ij} to compute the divergence value and divergence gradient ▷ (D_{JS} or D_{KLS})
    Update each π_i with the corresponding SVPG gradient ▷ (Equation 6.6)
end

bution ratios ζ_{ij}. The type of network(s) and the update rule is determined by the estimator choice. To form the SVPG gradient (Equation 6.6), the current value of ζ is used to compute the divergence value and the divergence gradient. The latter, as shown by Equation 6.8, is equivalent to the policy gradient with a distinctive reward function. We use the clipped PPO algorithm (Schulman et al., 2017) for the policy gradient, although other on-policy and off-policy RL methods are also applicable.

6.4 EXPERIMENTS

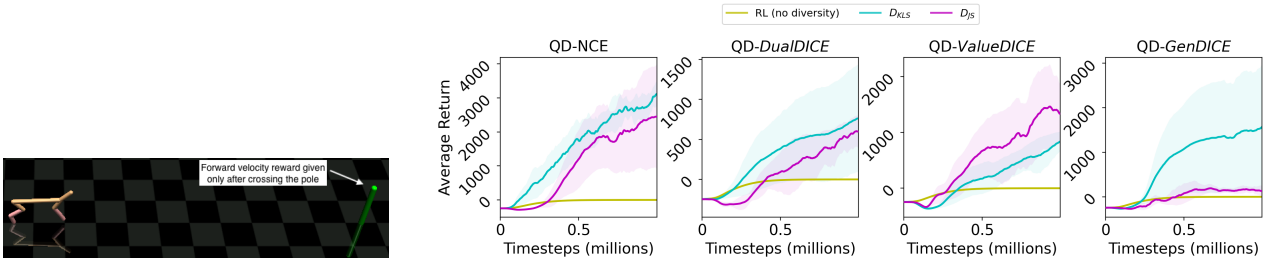
In this section, we train policy ensembles in various environments with continuous state- and action-space. We evaluate the different distribution ratio estimators and divergence metrics from the previous section. For ease of exposition, the algorithms are abbreviated as **QD-{*ratio-estimator*}-{*divergence*}**, hence QD-NCE-JS, for instance, is Algorithm 6.1 instantiated with $\exp(-D_{\text{JS}}(\rho_j, \rho_i)/T)$ as the kernel for SVPG, and NCE as the estimator for ζ. Our goal is to gauge the effectiveness of our approach in producing diverse, high-quality behaviors and suitably handling tasks with deceptive rewards. We also compare the NCE and DICE-based estimators on a quantitative metric correlated with behavioral diversity.

Qualitative Assessment of QD Behavior. We visualize the diversity of the learned skills in two environments – a robotic manipulator arm (Chen et al., 2018) and a 2D maze goal navigation task. The robotic arm models a 7 DOF Sawyer robot and is implemented in MuJoCo. For the peg-insertion task, we train a QD ensemble of 10 policies using the exponential of the negative Euclidean distance between the peg and the hole as the per-step reward for RL (the quality measure). Figure 6.1b shows some of the policies from a single ensemble trained with QD-*DualDICE*-JS. We find that while all the policies insert the peg in the hole, the final positions of the joints (marked with white rings) and the end-effector are



(a) (b) Standard RL (c) QD-NCE-JS (d) QD-NCE-KLS (e) QD-*GenDICE*-JS (f) QD-*GenDICE*-KLS

Figure 6.2: 2D Maze navigation task along with trajectories (state-visitations) for several methods.



(a) (b)

Figure 6.3: (a) Modified Half-Cheetah task that introduces multi-modality due to deceptive rewards; (b) Contrasting performance of standard RL (no diversity) with QD method in Algorithm 6.1.

markedly different. The resultant behaviors have dissimilar torque demands on the various joints, which is advantageous in scenarios such as transfer learning to a robot with system dynamics discrepancies. Figure 6.2a depicts a 2D navigation task with the start position (green ball at center bottom) and the goal location (small grey circle in the center of the maze). The per-step RL reward is the exponential of the negative Euclidean distance to the goal. We train an ensemble of 10 policies and plot final trajectories from some of them (each policy colored differently). Figure 6.2b shows results with the standard RL method, *i.e*, no diversity enforcement; the trajectories achieve the best possible cumulative returns but exhibit identical behavior. Figure 6.2c- 6.2f plots the paths for policies learned with the QD algorithm (specific instantiation mentioned in the caption). Though the cumulative returns now are lower than those with standard RL, the policies are noticeably more exploratory and cover large portions of the state-space.

Multi-modal Locomotion with Deceptive Rewards. One of the crucial benefits of learning a QD ensemble is that it potentially avoids the local optimum trap in the policy-search landscape due to deceptive rewards – if one policy gets stuck, the explicit diversity enforcement prevents other policies in the ensemble from the same fate. We evaluate this

hypothesis with the Half-Cheetah locomotion task from OpenAI Gym (Brockman et al., 2016). We modify the task such that the forward velocity reward is only given to the agent once the center-of-mass of the bot is beyond a certain threshold distance (d). Concretely, $r_t = vel_{x(t)} * \mathbb{1}(pos_{x(t)} >= d) - 0.1 * \|a_t\|_2^2$, where the second term penalizes large actions and is the default from Gym. Figure 6.3a is a rendering of the task. This change introduces multi-modality for policy optimization with a locally optimal solution to stand still at the starting location to avoid any action penalty. We compare the performance of the QD ensembles with a baseline standard-RL ensemble. The standard-RL ensemble has the same size as others but the constituent policies do not have any interactions; they apply independently computed gradients. For all baseline and QD ensembles, we select the policy with the highest cumulative returns after training and plot its learning curve in Figure 6.3b. We observe that the baseline RL (no diversity) latches onto the deceptive reward of minimizing the action penalty and gets stuck, achieving a cumulative return close to zero. In contrast, the diversity enforcing mechanism in the QD* ensemble enables *at-least* one member to reach the alternative mode where high forward velocity rewards are attained. This is evident in the final score accumulated by the member selected from each ensemble.

	Hist. Variance ↑	
Method	Walker-2d	Hopper
QD-DD-JS	**1.36**	0.45
QD-VD-JS	1.33	**0.50**
QD-GD-JS	0.63	0.14
QD-NCE-JS	0.13	0.11
QD-DD-KLS	0.10	0.10
QD-VD-KLS	0.24	0.45
QD-GD-KLS	0.07	0.40
QD-NCE-KLS	0.14	0.28
Gangwani et al. (2018)	0.10	0.08
DIAYN (Eysenbach et al., 2018)	0.22	0.11

Table 6.2: Diversity metric (histogram variance) with different estimators. Higher is better. Mnemonic: DD=*DualDICE*, VD=*ValueDICE*, GD=*GenDICE*

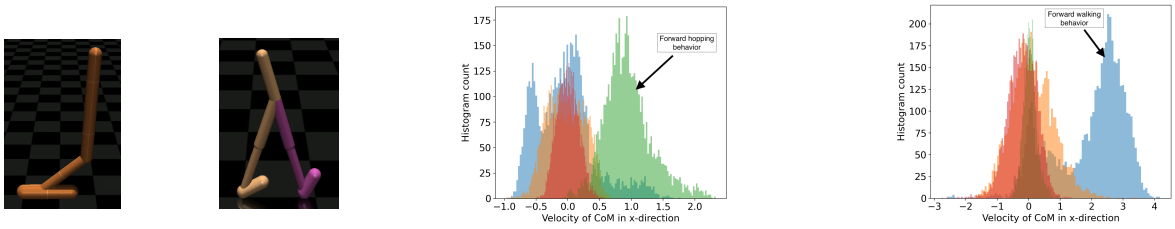
Quantitative Comparison of the Estimators. While the previous experiment exhibits that the NCE and DICE-based estimators can provide adequate diversity impetus, it does not provide insights about the comparative efficiency of the estimators in generating behavioral diversity in the trained ensemble. This is because the forward velocity reward is a *quality metric*, which is usually not aligned with the measure of diversity. For instance, an estimator may produce a policy that makes the Half-Cheetah run backwards—this is much desired from the diversity perspective but would perform badly on the quality metric that rewards forward motion. To evaluate the efficacy of our estimators for producing diverse behaviors,

and also for meaningful comparison with prior work (Eysenbach et al., 2018; Gangwani et al., 2018), we define a *diversity metric* as follows. For two locomotion tasks from Gym (Hopper and Walker-2d), we train policy ensembles without any environmental rewards. Thus, the gradient from the quality-enforcing component in Equation 6.4 is absent and the QD ensemble is trained only to maximize diversity. Post-training, we generate a few trajectories with all the constituent policies and plot a histogram with the velocity of the center-of-mass of the bot on the x-axis and the respective counts on the y-axis. We define the diversity metric to be the *variance* of this histogram. Intuitively, higher variance in the velocity of the bot is indicative of stronger behavioral diversity in the trained ensemble. Table 6.2 evaluates the various estimator on this diversity metric. We note that DICE-based estimators generally outperform NCE. Our intuition for this observation is that since NCE is an on-policy estimator (in contrast with the DICE-based estimators, which are off-policy), the availability of limited on-policy data in each iteration of Algorithm 6.1 has an impact on the efficiency of NCE. Lastly, many of the QD* methods compare favorably to the prior methods for learning diverse skills without environmental rewards (Eysenbach et al., 2018; Gangwani et al., 2018).

Diversity helps in the absence of environmental rewards. Designing a task-relevant reward function is typically laborious and error-prone. In the absence of an external reward signal, the diversity objective alone has been previously demonstrated to lead to useful skills (Eysenbach et al., 2018). We test the efficacy of our method in this setting using the Hopper and Walker tasks from Gym (Figures 6.4a- 6.4b) but modify the code to return a zero reward for each timestep. Thus, the gradient from the quality-enforcing component in Equation 6.4 is absent and the QD ensemble is trained only to maximize diversity. After training, we generate a few trajectories with the constituent policies and plot histograms with the velocity of the center-of-mass of the bot on the x-axis and the respective counts on the y-axis (Figures 6.4c- 6.4d). Both tasks are learned with QD-*GenDICE*-JS and each policy is colored differently. We note that the hopping (respectively walking) behavior *emerges* even in the absence of Gym rewards, suggesting that diversity is a strong signal for learning interesting skills.

6.5 RELATED WORK AND CONCLUSION

Neuroevolution methods inspired by Quality-Diversity (Pugh et al., 2016) have been proposed to efficiently manage the exploration-exploitation trade-off in RL. Conti et al. (2017) augment evolution strategies (Salimans et al., 2017) such that the fitness of a particle is



(a) (b) (c) (d)

Figure 6.4: (a)-(b) Rendering of the Hopper and Walker tasks, respectively; (c)-(d) Center-of-mass velocity histograms for Hopper and Walker, respectively, when trained with QD-*GenDICE*-JS. Arrows point to the emergence of locomotion in one of the policies from the ensemble.

computed by a weighted combination of the performance and novelty components. The novelty is determined based on a chosen behavior characterization (BC) metric. In MAP-Elites (Mouret & Clune, 2015), the entire behavior space is divided into a discrete grid, where each grid-dimension represents a BC. The algorithm then fills each grid cell with the highest quality solution possible for that cell. Recent methods integrate RL gradients with concepts from evolutionary computation (*e.g. random mutations*) to learn diverse exploratory agents (Khadka et al., 2019; Liu et al., 2019) or to discover coordination strategies for multi-agent RL (Khadka et al., 2019).

Diversity Maximization in RL. To aid exploration in sparse-reward tasks, Hong et al. (2018) encourage the current policy to be diverse compared to an archive of past policies, by maximizing a distance metric in the action space. Expanding on this idea, Doan et al. (2019) ensure sufficient diversity in a population by using *operators* for attraction and repulsion between agents. Towards learning diverse skills even in the absence of an external reward signal, maximization of the mutual information between the latent skill and the state-visitation of the skill-conditioned policy has been proposed (Florensa et al., 2017; Eysenbach et al., 2018). This is achieved by training a neural network to estimate the latent skill posterior, which provides proxy rewards for the policy. Zahavy et al. (2021) learn a diverse ensemble by using an explicit reward signal estimated by minimizing the correlation between the Successor-Features (Barreto et al., 2016) of the policies in the ensemble. Our work aims to broaden the SVPG algorithm (Liu et al., 2017) for learning a QD policy ensemble. We substitute the parameter-space RBF kernel used in their method with negative exponents of f-divergences, and employ distribution ratio estimation techniques to approximate the ensuing gradient on the policy parameters. Gangwani et al. (2018) avail SVPG to

improve self-imitation learning. Their procedure bears some resemblance to our NCE-based ratio estimation, though, in contrast to them, we use a mixture of stationary distributions as the contrastive noise.

6.5.1 Conclusion

In this work, we study methods to learn diverse and high-return policies. We extend the kernel-based SVPG algorithm with kernels based on f-divergence between the stationary distributions of policies. For kernels based on D_{JS} and D_{KLS}, we show that the problem reduces to that of efficient estimation of the ratio of the stationary distributions between policies. To compute these ratios, and consequently the SVPG gradient, we harness noise-contrastive estimation and several distribution ratio estimators widely used for off-policy evaluation and imitation learning. Experimental evaluation with continuous state- and action-space environments demonstrates that the approach is capable of generating diverse high-quality skills, assists in multi-modal environments with deceptive rewards, and provides a constructive learning signal when the external rewards are absent. Our algorithmic framework is general enough to accommodate any distribution ratio estimator. Utilizing future research on these estimators for improving the efficiency of QD training is an interesting direction, along with investigating which other f-divergences or integral probability metrics (IPMs such as the Wasserstein distance and the Maximum Mean Discrepancy) between stationary distributions could be incorporated into the framework.

CHAPTER 7: KNOWLEDGE TRANSFER UNDER STATE-ACTION DIMENSION MISMATCH

7.1 INTRODUCTION

Deep reinforcement learning (RL), which combines the rigor of RL algorithms with the flexibility of universal function approximators such as deep neural networks, has demonstrated a plethora of success stories in recent times. These include computer and board games (Mnih et al., 2015; Silver et al., 2016), continuous control (Lillicrap et al., 2015), and robotics (Rajeswaran et al., 2017), to name a few. Crucially though, these methods have been shown to be performant in the regime where an agent can accumulate vast amounts of experience in the environment, usually modeled with a simulator. For real-world environments such as autonomous navigation and industrial processes, data generation is an expensive (and sometimes risky) procedure. To make deep RL algorithms more sample-efficient, there is great interest in designing techniques for *knowledge transfer*, which enables accelerating agent learning by leveraging either existing trained policies (referred to as teachers), or using task demonstrations for imitation learning (Abbeel & Ng, 2004). One promising idea for knowledge transfer in RL is policy distillation (Rusu et al., 2015; Parisotto et al., 2015; Hinton et al., 2015), where information from the teacher policy network is transferred to a student policy network to improve the learning process.

Prior work has incorporated policy distillation in a variety of settings (Czarnecki et al., 2019). Some examples include the transfer of knowledge from simple to complex agents while following a curriculum over agents (Czarnecki et al., 2018), learning a centralized policy that captures shared behavior across tasks for multi-task RL (Teh et al., 2017), distilling information from parent policies into a child policy for a genetically-inspired RL algorithm (Gangwani & Peng, 2017), and speeding-up large-scale population-based training using multiple teachers (Schmitt et al., 2018). A common motif in these approaches is the use of Kullback-Leibler (KL) divergence between the state-conditional action distributions of the teacher and student networks, as the minimization objective for knowledge transfer. While simple and intuitive, this restricts learning from teachers that have the same output (action) space as the student, since KL divergence is only defined for distribution over a common space. An alternative to knowledge sharing in the action-space is information transfer through the embedding-space formed via the different layers of a deep neural network. (Liu et al., 2019) provides an example of this; it utilizes learned lateral connections between intermediate layers of the teacher and student networks. Although the action-spaces can now be different, the state-space is still required to be identical between the teacher and the student, since

the same input observation is fed to both the networks (Liu et al., 2019).

In our work, we present a transfer learning approach to accelerate the training of the student policy, by leveraging teacher policies trained in an environment with *different state- and action-space*. Arguably, there is a huge potential for data-efficient student learning by tapping into teachers trained on dissimilar, but related tasks. For instance, consider an available teacher policy for locomotion of a quadruped robot, where the (97-dimensional) state-space is the set of joint-angles and joint-velocities and the (10-dimensional) action-space is the torques to the joints. If we wish to learn locomotion for a hexapod robot (state-dimension 139, action-dimension 16), we conjecture that the learning could be kick-started by harnessing the information stored in the trained neural network for the quadruped, since both the tasks are locomotion for legged robots and therefore share an inherent structure. However, the dissimilar state- and action-space preclude the use of the knowledge transfer mechanisms proposed in prior work.

Our approach deals with the mismatch in the state- and action-space of the teacher and student in the following manner. To handle disparate actions, rather than using divergence minimization in the action-space, we transfer knowledge by augmenting representations in the layers of the student network with representations from the layers of the teacher network. This is similar to the knowledge flow in (Liu et al., 2019) using lateral connections, but with the important difference that we do not employ learnable matrices to transform the teacher representation. The mismatch in the observation- or state-space has not been considered in prior literature, to the best of our knowledge. We manage this by learning an embedding space which can be used to extract the necessary information from the available teacher policy network. These embeddings are trained to adhere to two properties. Firstly, they must be task-aligned. Our RL objective is the maximization of cumulative discounted rewards in the student environment, and therefore, the embeddings must be aligned to serve that goal. Secondly, we would like the embeddings to be correlated with the states encountered by the student policy. The embeddings are used to deterministically draw out knowledge from the teacher network. Therefore, a high correlation ensures that the most suitable teacher guidance is derived for each student state. We achieve this by maximizing the mutual information between the embeddings and student states. We evaluate our method on a set of challenging robotic locomotion tasks modeled using the MuJoCo simulator. We demonstrate the successful transfer of knowledge from trained teachers to students, in the scenario of mismatched state- and action-space. This leads to appreciable gains in sample-efficiency, compared to RL from scratch using only the environmental rewards.

7.1.1 Background and Notations

We consider the RL setting where the environment is modeled as an infinite-horizon discrete-time Markov Decision Process (MDP). The MDP is characterized by the tuple $(\mathcal{S},\ \mathcal{A},\ \mathcal{R},\ \mathcal{T},\ \gamma,\ p_0)$, where $\mathcal{S}$ and $\mathcal{A}$ are the continuous state- and action-space, respectively, $\gamma \in [0, 1)$ is the discount factor, and p_0 is the initial state distribution. Given an action $a_t \in \mathcal{A}$, the next state is sampled from the transition dynamics distribution, $s_{t+1} \sim \mathcal{T}(s_{t+1}|s_t, a_t)$, and the agent receives a scalar reward $r(s_t, a_t)$ determined by the reward function $\mathcal{R}$. A policy $\pi_\theta(a_t|s_t)$ defines the state-conditioned distribution over actions. The RL objective is to learn the policy parameters (θ) to maximize the expected discounted sum of rewards, $\eta(\pi_\theta) = \mathbb{E}_{p_0,\mathcal{T},\pi}\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\big]$.

Policy-gradient algorithms are widely used to estimate the gradient of the RL objective. Proximal policy optimization (PPO, Schulman et al. (2017)) is a model-free policy-gradient algorithm that serves as an efficient approximation to trust-region methods (Schulman et al., 2015). In each iteration of PPO, the rollout policy $(\pi_{\theta_{\text{old}}})$ is used to collect sample trajectories τ and the following surrogate loss is minimized over multiple epochs:

$$L^{\theta}_{\text{PPO}} = -\mathbb{E}_{\tau}\Big[\min\Big(r_t(\theta)\,\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\Big)\Big] \tag{7.1}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the ratio of the action probabilities under the current policy and rollout policy, and $\hat{A}_t$ is the estimated advantage. Variance in the policy-gradient estimates is reduced by employing the state-value function as a control variate (Mnih et al., 2016). This is usually modeled as a neural network V_ψ and updated using temporal difference learning:

$$L^{\psi}_{\text{PPO}} = \mathbb{E}_{\tau}\Big[\left(V_\psi(s_t) - V^{\text{targ}}(s_t)\right)^2\Big] \tag{7.2}$$

where $V^{\text{targ}}(s_t)$ is the bootstrapped target value obtained with TD(λ). To further reduce variance, Generalized Advantage Estimation (GAE, Schulman et al. (2015)) is used when estimating advantage. The overall PPO minimization objective then is:

$$L_{\text{PPO}}(\theta, \psi) = L^{\theta}_{\text{PPO}} + L^{\psi}_{\text{PPO}} \tag{7.3}$$

7.2 MUTUAL INFORMATION-BASED KNOWLEDGE TRANSFER

In this section, we outline our method for distilling knowledge from a pre-trained teacher policy to a student policy, in the hope that such knowledge sharing improves the sample-efficiency of the student learning process. Our problem setting is as follows. We assume

that the teacher and the student policies operate in two different MDPs. All the MDP properties ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{R}$, $\mathcal{T}$, γ, p_0) could be different, provided that some high-level structural commonality exists between the MDPs, such as the example of transfer from a quadruped robot to hexapod robot introduced in Section 7.1. Henceforth, for notational convenience, we refer to the MDP of the teacher as the *source* MDP, and that of the student as the *target* MDP. We assume the availability of a teacher policy network pre-trained in the source MDP. Crucially though, we do not assume access to the source MDP for any further exploration, or for obtaining demonstration trajectories that could be used for training in the target MDP using cross-domain imitation-learning techniques. We instead focus on extracting representations from the teacher policy network which are useful for learning in the target MDP.

In this work, we address knowledge transfer when $\mathcal{S}_{\text{src}} \neq \mathcal{S}_{\text{targ}}$, where $\mathcal{S}_{\text{src}}$ and $\mathcal{S}_{\text{targ}}$ denote the state-space of the source and target MDPs, respectively. To handle the mismatch, we introduce a learned embedding-space parameterized by an encoder function $\phi(\cdot)$, and defined as $\mathcal{S}_{\text{emb}} \coloneqq \{\phi(s) \mid s \in \mathcal{S}_{\text{targ}}\}$. Data points from this embedding space are used to extract useful information from the teacher policy network. Therefore, we further enforce that the dimension of the embedding space matches the dimension of the state-space in the source MDP, i.e., $|\mathcal{S}_{\text{emb}}| = |\mathcal{S}_{\text{src}}|$. Note that this does not necessitate that any embedding vector $s \in \mathcal{S}_{\text{emb}}$ be a feasible input state in the source MDP. To learn the encoder function $\phi(\cdot)$, we consider the following two desiderata. Firstly, the embeddings must be learned to facilitate our objective of maximizing the cumulative discount rewards in the target MDP. In subsection 7.2.1, we show how to achieve this by utilizing the policy gradient to update embedding parameters. Secondly, we wish for a high correlation between the input states of the target MDP and the embedding vectors produced from them. The embeddings are used to deterministically derive representations from the teacher network, and hence a high correlation helps to obtain the most appropriate teacher guidance for each of the states encountered by the target policy. To this end, we propose a mutual information maximization objective; this is detailed in subsection 7.2.2.

7.2.1 Task-aligned Embedding Space

This section describes our approach for training the encoder parameters (ϕ) such that the generated embeddings are aligned with the RL objective. We begin by detailing the architecture that we use for transfer of knowledge from a teacher, pre-trained in source MDP, to a student policy in the target MDP with different state- and action-space. Inspired by the concept of knowledge-flow used in (Liu et al., 2019), we employ lateral connections

between the student and teacher networks, which augment the representations in the layers of the student with useful representations from the layers of the teacher. A crucial benefit of this approach is that since information sharing happens through the hidden layers, the output (action) space of the source and target MDPs can be disparate, as is the scenario in our experiments. It is also quite straightforward to include multiple teachers in this architecture to distill diverse knowledge into a student; we leave this to future work.

We draw out knowledge from both the teacher policy and state-value networks. We denote the teacher policy and value network with $\pi_{\theta'}$ and $V_{\psi'}$, respectively, where the parameters (θ', ψ') are held fixed throughout the training. Analogously, (θ, ψ) are the trainable parameters for the student policy and value networks. Let N_π denote the number of hidden layers in the teacher (and student) policy network, and N_V be the number of hidden layers in the teacher (and student) value network. In general, the teacher and student networks could have a different number of layers, but we assume them to be the same for ease of exposition.

In the target MDP, the student policy observes a state $s_{\text{targ}} \in \mathcal{S}_{\text{targ}}$, which is fed to the encoder to produce the embedding $\phi(s_{\text{targ}}) \in \mathcal{S}_{\text{emb}}$. Since $|\mathcal{S}_{\text{emb}}| = |\mathcal{S}_{\text{src}}|$, this embedding can be readily passed through the teacher networks to extract $\{z^j_{\theta'}, 1 \leq j \leq N_\pi\}$, representing the pre-activation outputs of the N_π hidden layers of the teacher policy network, and $\{z^j_{\psi'}, 1 \leq j \leq N_V\}$, representing the pre-activation outputs of the N_V hidden layers of the teacher value function network. To obtain the pre-activation representations in the student networks, we feed in the state s_{targ} and perform a weighted linear combination of the appropriate outputs with the corresponding pre-activations from the teacher networks. Concretely, to obtain the hidden layer outputs $h^j_{\pi_\theta}$ and $h^j_{V_\psi}$ at layer j in the student networks, we have the following:

$$h^j_{\pi_\theta} = \sigma\left(p^j_\theta z^j_\theta + (1 - p^j_\theta) z^j_{\theta'}\right) \qquad h^j_{V_\psi} = \sigma\left(p^j_\psi z^j_\psi + (1 - p^j_\psi) z^j_{\psi'}\right) \tag{7.4}$$

where σ is the activation function, and $p^j_\theta, p^j_\psi \in [0, 1]$ are layer-specific learnable parameters denoting the mixing weights. In the target MDP, the student network is optimized for the RL objective $L_{\text{PPO}}(\theta, \psi)$, mentioned in Equation 7.3. The outputs of the student policy and value networks, and hence L_{PPO}, depend on the encoder parameters (ϕ) through the representation sharing (Equation 7.4) enabled by the lateral connections stemming from the pre-trained teacher network. Therefore, an intuitive objective for shaping the embeddings such that they become task-aligned is to optimize them using the original RL loss gradient: $\phi \leftarrow \phi - \alpha \nabla_\phi L_{\text{PPO}}(\theta, \psi, \phi, \theta', \psi')$. Note that $L_{\text{PPO}}(\cdot)$ now also depends on the fixed teacher parameters (θ', ψ').

The learnable mixing weights $p^j_\theta, p^j_\psi \in [0, 1]$ control the influence of the teacher's representation on the student outputs – higher the value, lesser the impact. We argue that a

low value for these coefficients helps in the early phases of the training process by providing necessary information to kick-start learning. At the end of the training, however, we desire that the student becomes completely independent of the teacher, since this helps in faster test-time deployment of the agent. To encourage this, we introduce additional *coupling-loss* terms that drive p_θ^j, p_ψ^j towards 1 as the training progresses:

$$L_{\text{coupling}} = -\frac{1}{N_\pi}\sum_{j=1}^{N_\pi}\log\left(p_\theta^j\right) - \frac{1}{N_V}\sum_{j=1}^{N_V}\log\left(p_\psi^j\right) \tag{7.5}$$

Experimentally, we observe that although the student becomes independent in the final stages of training, it is able to achieve the same level of performance that it would if it could still rely on the teacher.

7.2.2 Enriched Embeddings with Mutual Information Maximization

As outlined in the previous section, at each timestep of the discrete-time target MDP, the representation distilled from the teacher networks is a fixed function f of the embedding vector generated from the current input state: $f(\theta', \psi', \phi(s_{\text{targ}}))$, where (θ', ψ') are fixed. It is desirable to have a high degree of correlation between s_{targ} and $f(\theta', \psi', \phi(s_{\text{targ}}))$ because, intuitively, the teacher representation that is the most useful for the student should be different at different input states. To aid with this, we utilize a surrogate objective that instead maximizes the correlation between s_{targ} and the embeddings $\phi(s_{\text{targ}})$, defined using the principle of mutual information (MI). If we view s_{targ} as a stochastic input $\boldsymbol{s}$, the encoder output is then also a random variable $\boldsymbol{e}$, and the mutual information between the two is defined as:

$$\mathcal{I}(\boldsymbol{s}; \boldsymbol{e}) = \mathcal{H}(\boldsymbol{s}) - \mathcal{H}(\boldsymbol{s}|\boldsymbol{e}) \tag{7.6}$$

where $\mathcal{H}$ denotes the differential entropy. Direct maximizing of the MI is intractable due to the unknown conditional densities. However, it is possible to obtain a lower bound to the MI using a variational distribution $q_\omega(\boldsymbol{s}|\boldsymbol{e})$ that approximates the true conditional distribution $p(\boldsymbol{s}|\boldsymbol{e})$ as follows:

$$\begin{aligned}
\mathcal{I}(\boldsymbol{s}; \boldsymbol{e}) &= \mathcal{H}(\boldsymbol{s}) - \mathcal{H}(\boldsymbol{s}|\boldsymbol{e}) \\
&= \mathcal{H}(\boldsymbol{s}) + \mathbb{E}_{\boldsymbol{s},\boldsymbol{e}}[\log p(\boldsymbol{s}|\boldsymbol{e})] \\
&= \mathcal{H}(\boldsymbol{s}) + \mathbb{E}_{\boldsymbol{s},\boldsymbol{e}}[\log q_\omega(\boldsymbol{s}|\boldsymbol{e})] + \mathbb{E}_{\boldsymbol{e}}\big[D_{\text{KL}}(p(\boldsymbol{s}|\boldsymbol{e})||q_\omega(\boldsymbol{s}|\boldsymbol{e}))\big] \\
&\geq \mathcal{H}(\boldsymbol{s}) + \mathbb{E}_{\boldsymbol{s},\boldsymbol{e}}[\log q_\omega(\boldsymbol{s}|\boldsymbol{e})]
\end{aligned} \tag{7.7}$$

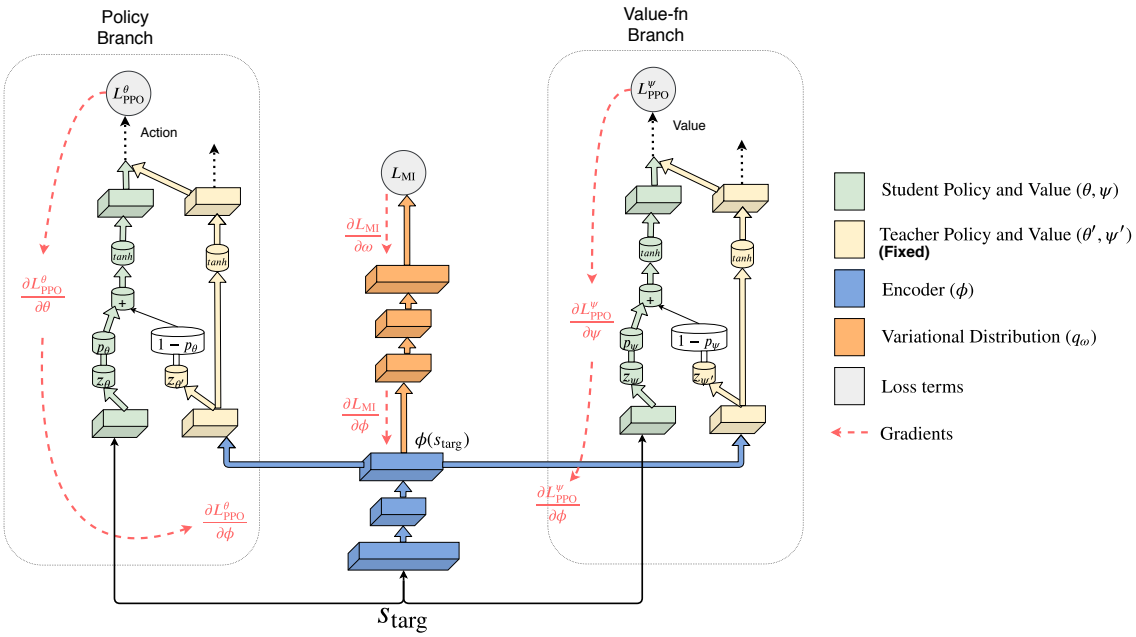


Figure 7.1: Schematic diagram of our complete architecture (best viewed in color). The encoder parameters ϕ (blue) receive gradients from three sources: the policy-gradient loss L^{θ}_{PPO}, the value function loss L^{ψ}_{PPO}, and the mutual information loss L_{MI}. The teacher networks (yellow) remain fixed throughout training and do not receive any gradients. In the student networks (green), the pre-activation representations are linearly combined (using learnt mixing weights) with the corresponding representations from the teacher (Equation 7.4). This knowledge-flow occurs at all layers, although we show it only once for clarity of exposition.

where the last inequality is due to the non-negativity of the KL divergence. This is known as the variational information maximization algorithm (Agakov & Barber, 2004). Re-writing in terms of target-MDP states and the encoder parameters, the surrogate objective jointly optimizes over the variational and encoder parameters:

$$\max_{\omega,\phi} \mathbb{E}_{s_{\text{targ}}}[\log q_\omega(s_{\text{targ}}|\phi(s_{\text{targ}}))] \tag{7.8}$$

where $\mathcal{H}(\boldsymbol{s})$ is omitted since it is a constant *w.r.t.* the concerned parameters. In terms of the loss function to minimize, we can succinctly write:

$$L_{\text{MI}}(\phi, \omega) = -\mathbb{E}_{s\sim\rho_{\pi_\theta}}[\log q_\omega(s|\phi(s))] \tag{7.9}$$

where ρ_{π_θ} is the state-visitation distribution of the student policy in the target MDP. In our experiments, we use a multivariate Gaussian distribution (with a learned diagonal covariance matrix) to model the variational distribution q_ω. Although this simple model yields good

Algorithm 7.1: Mutual Information based Knowledge Transfer (MIKT)

Input : θ', ψ' fixed teacher policy and value networks
θ, ψ: student policy and value networks
$\{p\}$: set of coupling parameters for policy and value networks
ϕ: encoder parameters
ω: variational distribution parameters

for *each iteration* **do**
1 Run π_θ in target MDP and collect few trajectories τ
2 **for** *each minibatch* $m \in \tau$ **do**
3 Update θ, ψ with $\nabla_{\theta,\psi} L_{\text{PPO}}(\theta, \psi, \phi, \theta', \psi')$
4 Update ϕ with $\nabla_\phi \big[L_{\text{MI}}(\phi, \omega) + L_{\text{PPO}}(\theta, \psi, \phi, \theta', \psi')\big]$
5 Update ω with $\nabla_\omega L_{\text{MI}}(\phi, \omega)$
6 Update $\{p\}$ using $[L_{\text{coupling}} + L_{\text{PPO}}]$
7 **end**
8 **end**

performance, more expressive model classes, such as mixture density networks and flow-based models (Rezende & Mohamed, 2015) could be readily incorporated as well, to learn complex and multi-modal distributions.

7.2.3 Algorithm and Architecture

Figure 7.1 shows the schematic diagram of our complete architecture and gradient flows, along with a description of the implemented neural networks. We refer to our algorithm as MIKT, for *Mutual Information based Knowledge Transfer*. Algorithm 7.1 outlines the main steps of the training procedure. In each iteration, we run the policy in the target MDP and collect a batch of trajectories. This experience is then used to compute the RL loss (Equation 7.3) and the mutual information loss (Equation 7.9), enabling the calculation of gradients for the different parameters (Lines 3–6). Using both the losses to update the encoder (ϕ) helps us to satisfy the desiderata on the embeddings – that they should be task-aligned and correlated with the states in the target MDP. The coupling parameters $\{p^j\}$, used for the weighted combination of the representations in the teacher and student networks, are updated with the coupling-loss (Equation 7.5) along with the RL loss. In each iteration of the algorithm, the PPO update ensures that the state-action visitation distribution of the policy π_θ is modified by only a small amount. This is because of the clipping on the importance-sampling ratio when computing the PPO gradient. In addition to this, we experimentally found that enforcing an explicit KL-regularization on the policy

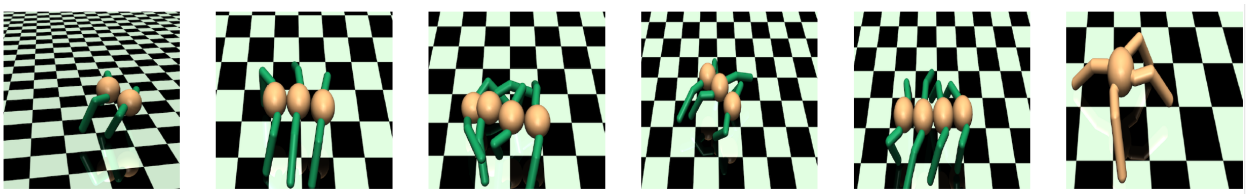


Figure 7.2: Our MuJoCo locomotion environments. From left to right: *CentipedeFour, CentipedeSix, CentipedeEight, CpCentipedeSix, CpCentipedeEight, Ant.* The centipede agents are configured using the details in Wang et al. (2018), while the Ant task is from OpenAI Gym. *Cp* denotes that the centipede is crippled (some legs disabled).

further stabilizes learning. Let $\pi_\theta, \pi_{\theta_{\text{old}}}$ denote the current and the rollout policy, respectively. The loss is then formalized as:

$$L_{\text{KL}}(\theta, \theta_{\text{old}}) = \mathbb{E}_{s\sim\rho_{\pi_{\theta_{\text{old}}}}}\left[\mathcal{D}_{KL}(\pi_\theta(\cdot|s)||\pi_{\theta_{\text{old}}}(\cdot|s))\right] \tag{7.10}$$

7.3 EXPERIMENTS

In this section, we perform experiments to quantify the efficacy of our algorithm, MIKT, for transfer learning in RL. We address the following questions: a) Can we do successful knowledge transfer between a teacher and a student with *different state- and action-space*? b) Are both the losses $\{L_{\text{PPO}}, L_{\text{MI}}\}$ important for learning useful embeddings ϕ? c) How does task-similarity affect the benefits that can be reaped from MIKT?

Environment	State Dimension	Action Dimension
CentipedeFour	97	10
CentipedeSix	139	16
CentipedeEight	181	22
CpCentipedeSix	139	12
CpCentipedeEight	181	18
Ant	111	8

Table 7.1: MuJoCo locomotion environments.

Environments. We evaluate using locomotion tasks for legged robots, modeled in OpenAI Gym (Brockman et al., 2016) using the MuJoCo physics simulator. Specifically, we use the environments provided by Wang et al. (2018), where the agent structure resembles that of a centipede – it consists of repetitive torso bodies, each having two legs. Figure 7.2 shows an illustration of the different centipede agents. The agent is rewarded for running fast in a particular direction. Table 7.1 includes the state and action dimensions of all the agents. Centipede-x refers to a centipede with x legs; we use $x \in \{4, 6, 8\}$. We use additional

environments where the centipede is crippled (some legs disabled) and denote these by *Cp*Centipede-x. Finally, we include the standard Ant task from the MuJoCo suite. Note that all robots have different state and action dimensions. Nevertheless, these locomotion tasks share an inherent structure that could be exploited for transfer learning between the centipedes of various types. We now demonstrate that MIKT achieves this successfully.

Baselines. We compare MIKT with two baselines: a) *Vanilla Policy Gradient (VPG)*, which learns the task in the target MDP from scratch using only the environmental rewards. Any transfer learning algorithm which effectively leverages the available teacher networks should be able to outperform this baseline that does not receive any prior knowledge it can use. We use the standard PPO (Schulman et al., 2017) algorithm for this baseline. b) *MLP Pre-trained (MLPP)* In our setting, the teacher and the student networks have dissimilar input and output dimensions (because the MDPs have different state- and action-spaces). A natural transfer learning strategy is to remove the input and output layers from the pre-trained teacher and replace them with new learnable layers that match the dimensions required of the student policy (analogously value) network. The middle stack of the deep neural network is then fine-tuned with the RL loss. Prior work has shown that such a transfer is effective in certain computer vision tasks.

7.3.1 Results

Figure 7.3 plots the learning curves for MIKT and our two baselines in different transfer learning experiments. Each plot is titled "x to y", where x is the source (teacher) MDP and y is the target (student) MDP. We run each experiment with 5 different random seeds and plot the average episodic returns (mean and standard deviation) on the y-axis, against the number of timesteps of environment interaction (2 million total) on the x-axis. VPG does not use utilize the pre-trained teachers. We observe that its performance improves with the training iterations, albeit at a sluggish pace. MLPP uses the middle stack of the pre-trained teacher network as an initialization and trains the input and output layers from scratch. It only performs on par with VPG, potentially due to the non-constructive interaction between the pre-trained and randomly initialized parameters of the student networks. This indicates that the MLPP strategy is not productive for transfer learning across the RL locomotion tasks considered. Finally, we note that our algorithm (MIKT) vastly outperforms the two baselines, both achieving higher returns in earlier stages of training and reaching much higher final performance. This proves that firstly, these tasks do have a structural commonality such that a teacher policy trained in one task could be used advantageously to accelerate learning in a different task; and secondly, that MIKT is a successful approach for achieving such a

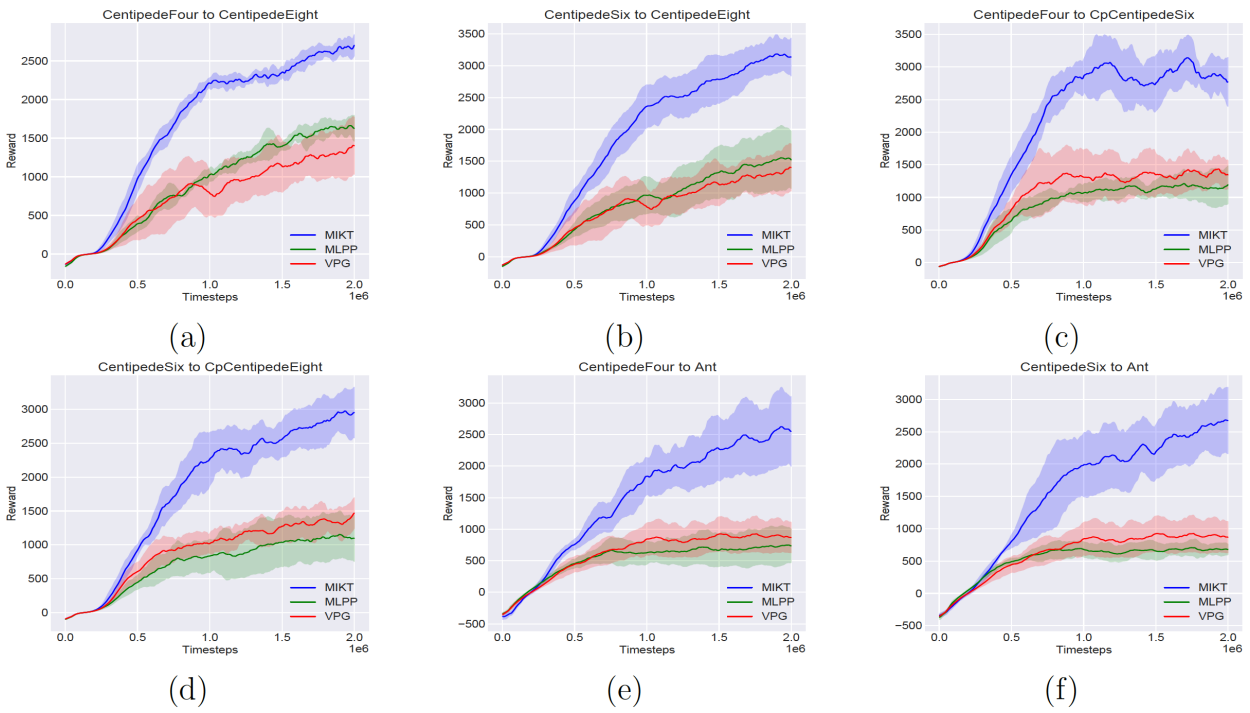


Figure 7.3: Performance of our transfer learning algorithm (MIKT) and the baselines (VPG and MLPP) on the MuJoCo locomotion tasks. Each plot is titled "x to y", where x is the source (teacher) MDP and y is the target (student) MDP. (a) CentipedeFour to CentipedeEight, (b) CentipedeSix to CentipedeEight, (c) CentipedeFour to CpCentipedeSix, (d) CentipedeSix to CpCentipedeEight, (e) CentipedeFour to Ant, (f) CentipedeSix to Ant.

knowledge transfer. This works even when the teacher and student MDPs have different state- and action-spaces, and is realized by learning embeddings that are task-aligned and are optimized with a mutual information loss (Algorithm 7.1).

7.3.2 Ablation Studies

Are gradients from both $\{L_{\mathbf{PPO}}, L_{\mathbf{MI}}\}$ to the encoder beneficial? To quantify this, we experiment with two variants of our algorithm, each of which removes one of the components: *MIKT w/o MI*, which does not update ϕ with the mutual information loss proposed in Section 7.2.2, and *MIKT w/o RL gradients*, which omits using the policy-gradient and the value function TD-error gradient for the encoder. Figure 7.4 plots the performance of these variants and compares it to MIKT (which includes both the losses). We note that *MIKT w/o MI* generally struggles to learn in the early stages of training; see for instance Figure 7.4 (c), (d). *MIKT w/o RL gradients* does comparatively better early on in training, but it is evident that MIKT is the most performant, both in terms of early training efficiency and the average episodic returns of the final policy. This supports our design choice of using

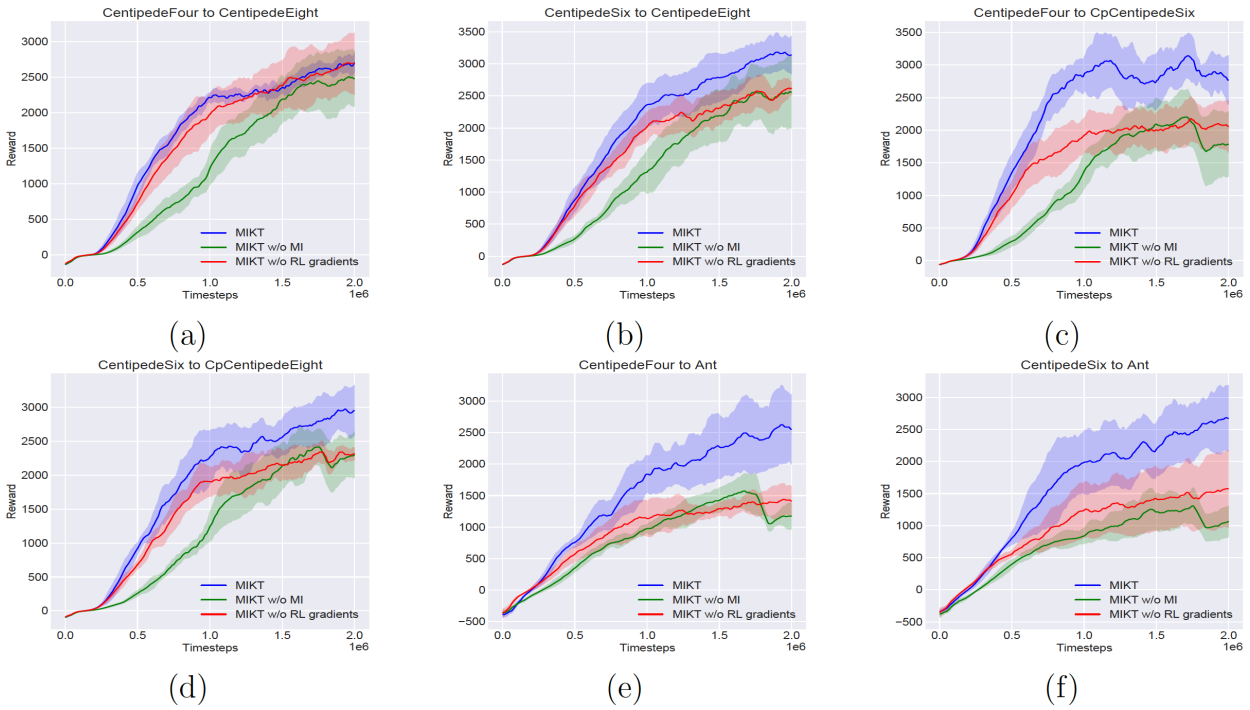


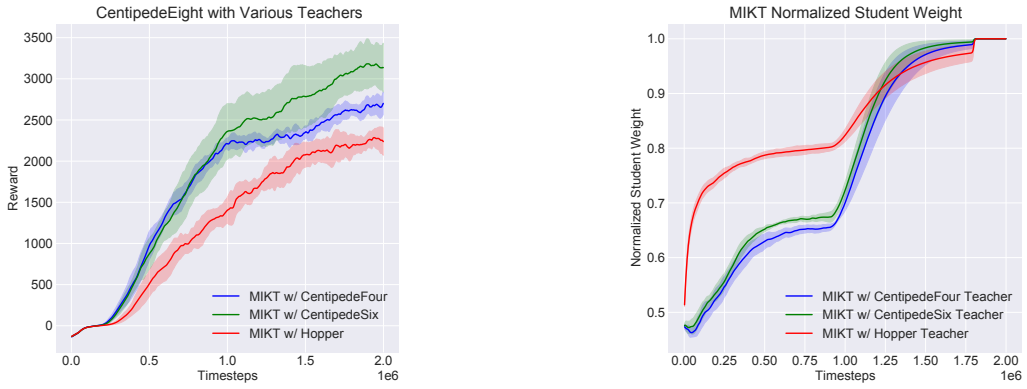
Figure 7.4: Ablation on the importance of each of $\{L_{\text{PPO}}, L_{\text{MI}}\}$ for training the encoder ϕ. MIKT (blue) is compared with two variants: *MIKT w/o MI* (L_{MI} not used) and *MIKT w/o RL gradients* (L_{PPO} not used).

both $\{L_{\text{PPO}}, L_{\text{MI}}\}$ to update the encoder ϕ.

How sensitive is MIKT to the task-similarity? It stands to reason that the benefits of transfer learning depend on the task-similarity between the teacher and the student. To better understand this in the context of MIKT, we consider learning in the CentipedeEight environment using different types of teachers – CentipedeFour, CentipedeSix, and Hopper. In Figure 7.5a, we note that the influence of the Centipede-{Four,Six} teachers is much more significant than the Hopper teacher. This is likely because the motions of the centipedes share strong similarity, while the Hopper (that is trained to hop) is a dissimilar task and thus less useful for transfer learning. Figure 7.5b plots the value of the weight on the student representation that is used for the weighted linear combination with the teacher (*cf.* §7.2.1). With the Hopper teacher, quite early in the training, the student learns to trust its own learned representations rather than incorporate knowledge from the dissimilar teacher.

7.4 RELATED WORK AND CONCLUSION

The concepts of knowledge transfer and information sharing between deep neural networks



(a) (b)

Figure 7.5: Training on CentipedeEight with different teachers. (a) Transfer from a dissimilar teacher (Hopper) is less effective compared to using Centipede teachers. (b) Value of the weight on the student representation in the weighed linear combination. With the Hopper teacher, the value rises sharply in the early stages, indicating a low teacher contribution.

have been extensively researched for a wide variety of tasks in machine learning. In the context of reinforcement learning, the popular paradigms for knowledge transfer include imitation-learning, meta-RL, and policy distillation; each of these being applicable under different settings and assumptions. Imitation learning algorithms (Ng et al., 2000; Ziebart et al., 2008) utilize teacher demonstrations to extract useful information (such as the teacher reward function in inverse-RL methods) and use that to accelerate student learning. Meta-RL considers a distribution over tasks that share some structural similarity, and the objective is to discover this generalizable knowledge for accelerating the process of learning on a new task. Our work is most closely related to policy distillation methods (Rusu et al., 2015; Parisotto et al., 2015; Czarnecki et al., 2019), where pre-trained teacher networks are available and can expedite learning in dissimilar (but related) student tasks. Prior work has considered teachers in various capacities. Rusu et al. (2016) and Liu et al. (2019) utilize learned cross-connections between intermediate layers of teacher networks—that have been pre-trained on various source tasks—and a student network to effectively transfer knowledge and enable more efficient learning on a target task. Ahn et al. (2019) use an objective based on the mutual information between the corresponding layers of teacher and student networks, and show gains in image classification tasks. In Hinton et al. (2015), information from a large model (teacher) is compressed into a smaller model (student) using a distillation process that uses the temperature-regulated softmax outputs from the teacher as targets to train the student. Schmitt et al. (2018) propose a large-scale population-based training pipeline that allows a student policy to leverage multiple teachers specialized in different tasks. The aforementioned methods assume that the teacher and student share the input observation

space. Different from these, our approach handles the mismatch in the state-space by training an embedding space which is utilized for efficient knowledge transfer. Rozantsev et al. (2018) employ layer-wise weight regularization and evaluate on (un-)supervised tasks where the input distributions for source and target domains have semantic similarity and are static. For RL tasks, the input distributions change dynamically as the student policy is updated; it is unclear if enforcing similarity between the networks for all inputs by coupling the weights is ideal. Gamrian & Goldberg (2018) use GANs to learn a mapping from target states to source states. In addition to requiring that the source and the target domains have the same action-space, their method also relies on the exploratory samples collected in the source MDP for training the GAN. In contrast, we handle the action-space mismatch and do not assume access to the source MDP for exploration.

Our work also has connections to policy distillation methods that use *implicit* teachers, rather than external pre-trained models. In Czarnecki et al. (2018), the authors recommend a curriculum over agents, rather than the usual curriculum over tasks. Such a curriculum trains simple agents first, the knowledge of which is then distilled into more complex agents over time. Akkaya et al. (2019) iterate on policy architectures by utilizing behavior-cloning with DAgger; the new architecture (student) is trained using the old architecture (teacher). Distillation has been used in multi-task RL (Teh et al., 2017) to learn a centralized policy that captures generalizable information from policies trained on individual tasks. (Gangwani & Peng, 2017) combine ideas from the genetic-algorithms literature and distillation to train offspring policies that inherit the best traits of both the parent policies. Since all these approaches transfer information in the action-space by minimizing the KL-divergence between state-conditional action distributions, they share the limitation that the student can only leverage a teacher with the same output (action) space. Our approach avoids this by using the representations in the different layers of the neural network for knowledge sharing, enabling transfer-learning in many diverse scenarios as shown in our experiments.

7.4.1 Conclusion

In this work, we proposed an algorithm for transfer learning in RL where the teacher (source) and the student (task) agents can have arbitrarily different state- and action-spaces. We achieve this by learning an encoder to produce embeddings that draw out useful representations from the teacher networks. We argue that training the encoder with both the RL-loss and the mutual information-loss yields rich representations; we provide empirical validation for this as well. Our experiments on a set of challenging locomotion tasks involving many-legged centipedes show that MIKT is a successful approach for achieving knowledge transfer when the teacher and student MDPs have mismatched state- and action-space.

CHAPTER 8: DATA-SHARING IN META REINFORCEMENT LEARNING

8.1 INTRODUCTION

Deep Reinforcement Learning (RL) has achieved success on a wide variety of tasks, ranging from computer games to robotics. However, RL agents are typically trained on a single task and are extremely sample-inefficient, often requiring millions of samples to learn a good policy for just that one task. Ideally, RL agents should be able to utilize their prior knowledge and adapt to tasks quickly, just as humans do. Meta-learning, or learning to learn, has achieved promising results in this regard, allowing agents to exploit the shared structure between tasks in order to adapt to new tasks quickly during meta-test time.

Although meta-learned policies can adapt quickly during meta-test time, training these meta-learned policies could still require a large amount of data. Several popular meta-RL methods (Duan et al., 2016; Wang et al., 2016; Finn et al., 2017; Mishra et al., 2017; Rothfuss et al., 2018) utilize on-policy data during meta-training to better align with the setup at meta-test time, where the agent must generate on-policy data for an unseen task and use it for adapting to the task. Recent works (Rakelly et al., 2019; Fakoor et al., 2019) have sought to incorporate off-policy RL (Haarnoja et al., 2018; Fujimoto et al., 2018) into meta-RL to improve sample efficiency. The combination of off-policy RL and data relabeling, in which experience is shared across tasks, has been utilized in the multi-task RL setting where the agent learns to solve multiple different yet related tasks. Experience collected for one task may be insignificant for training a policy to learn that task, but could be extremely informative in training a policy to learn a different task (Andrychowicz et al., 2017; Eysenbach et al., 2020). For example, an agent trying to shoot a hockey puck into a net might miss to the right. This experience could easily be used to train an agent to shoot a puck into a net positioned further to the right.

Both meta-RL and multi-task RL involve training on a distribution of tasks, so it follows that we can also combine relabeling techniques with meta-RL algorithms in order to boost both sample efficiency and asymptotic performance. In meta-RL, an agent learns to explore sufficiently to identify the task it is supposed to be solving, and then uses that knowledge to achieve high task returns. The agent collects exploratory pre-adaptation data, then undergoes some adaptation process using that pre-adaptation data. Finally, after adaptation, the agent attempts to solve the task. Meta-RL algorithms typically have a meta-training phase followed by a meta-test phase. The goal during meta-training is to train the meta-parameters such that they could be quickly adapted to solve any task from the meta-train

task distribution, given a small amount of data from that task. At meta-test time, given a new unseen task, the goal is to rapidly adapt the *learned* meta-parameters for this task, using a small amount of task-specific data. The focus in this work is to improve the sample efficiency of the meta-training phase via data sharing.

Using concepts from maximum entropy RL (MaxEnt RL), we introduce a relabeling scheme for the meta-RL setting. Prior relabeling methods for multi-task RL have used the total reward of the trajectory under different tasks to guide the relabeling (Eysenbach et al., 2020; Li et al., 2020). Direct application of this type of relabeling to the meta-RL setting is potentially sub-optimal since the multi-task RL and meta-RL objectives are distinct (learning to perform many tasks vs. learning to learn a new task). Towards developing an approach more suited to meta-RL, we define the notion of the *utility* of a trajectory under the different tasks, where the utility captures the usefulness of the trajectory for efficient adaptation under those tasks.

We call our method Hindsight Foresight Relabeling (HFR) – we use *hindsight* in replaying the experience using reward functions from different tasks, and we use *foresight* in computing the utility of trajectories under different tasks and constructing a relabeling distribution over tasks using these utilities. We demonstrate the efficacy of our method on a variety of robotic manipulation and locomotion tasks. Notably, we show that our method, as the first meta-RL relabeling technique (applied during meta-training) that we are aware of, leads to improved performance compared to prior relabeling schemes designed for multi-task RL.

8.2 PRELIMINARIES

In reinforcement learning (RL), the environment is modeled as a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \mathsf{p}, \gamma, p_1)$, where $\mathcal{S}$ is the state-space, $\mathcal{A}$ is the action-space, r is the reward function, p is the transition dynamics, $\gamma \in [0, 1)$ is the discount factor, and p_1 is the initial state distribution. At timestep t, the agent π_θ, parameterized by parameters θ, observes the state $s_t \in \mathcal{S}$, takes an action $a_t \sim \pi_\theta(a_t|s_t)$, and observes the next state $s_{t+1} \sim \mathsf{p}(s_{t+1}|s_t, a_t)$ and the reward $r(s_t, a_t)$. The goal is to maximize the expected cumulative discounted rewards: $\max_\theta \mathbb{E}_{s_t, a_t \sim \pi_\theta}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t)\right]$.

8.2.1 Meta-Reinforcement Learning

In the general meta-reinforcement learning (meta-RL) setting, there is a family of tasks that is characterized by a distribution $p(\psi)$, where each task ψ is represented by an MDP $\mathcal{M}_\psi = (\mathcal{S}, \mathcal{A}, r_\psi, \mathsf{p}_\psi, \gamma, p_1)$. The tasks share the components $(\mathcal{S}, \mathcal{A}, \gamma, p_1)$, but can differ in

the reward function r_ψ (e.g. navigating to different goal locations) and/or the transition dynamics p_ψ (e.g. locomotion on different terrains). In this work, we consider the setting where the *tasks share the same transition dynamics (*i.e., $\mathsf{p}_\psi = \mathsf{p}$*), but differ in the reward function*. The goal in meta-learning is to learn a set of meta-parameters such that given a new task from $p(\psi)$ and small amount of data for the new task, the meta-parameters can be efficiently adapted to solve the new task. In the context of meta-RL, given new task ψ, the agent collects some initial trajectories $\{\tau_{\text{pre}}\}$, each being a sequence $\{s_1, a_1, s_2, a_2, \dots\}$, and then undergoes some adaptation procedure $f_\phi(\pi_\theta, \tau_{\text{pre}}, r_\psi)$ (*e.g.,* a gradient update (Finn et al., 2017) or a forward pass through an RNN (Duan et al., 2016)). The adaptation procedure returns a new policy π'. With this post-adaptation policy, the agent ought to maximize the cumulative discounted rewards it achieves. Overall, the meta-RL objective is:

$$\max_{\theta,\phi} \mathbb{E}_{\psi\sim p(\psi),(s_t,a_t)\sim\pi'(\theta,\phi)}\left[\sum_{t=1}^{\infty}\gamma^{t-1}r_\psi(s_t,a_t)\right]; \quad \pi'(\theta,\phi) = f_\phi(\pi_\theta, \tau_{\text{pre}}, r_\psi) \tag{8.1}$$

where θ, ϕ are the meta-parameters that are learned in the meta-training phase. A meta-RL agent must learn a good adaptation procedure f_ϕ that is proficient in extracting salient information about the task at hand, using few pre-adaptation trajectories τ_{pre}. At the same time, it should learn the policy meta-parameters θ such that it can achieve high returns after the adaptation process, *i.e.*, while following the policy $\pi' = f_\phi(\pi_\theta, \tau_{\text{pre}}, r_\psi)$.

8.2.2 PEARL

In this work, we use PEARL (Rakelly et al., 2019) as our base meta-RL algorithm since it uses off-policy RL and provides structured exploration via posterior sampling. PEARL is built on top of Soft Actor-Critic (Haarnoja et al., 2018) and trains an encoder network $q_\phi(z|c)$ that takes in the "context" c, which consists of a batch of (s_t, a_t, r_t, s_{t+1}) transitions, and produces the latent embedding z. The intent is to learn the encoder such that embedding z encodes some salient information about the task. The adaptation step f_ϕ in PEARL corresponds to generating this latent z and then conditioning the policy and the value function networks on it. The policy $\pi_\theta(a|s,z)$ is trained using loss L_{actor} :

$$L_{\text{actor}} = \mathbb{E}_{s\sim B, a\sim\pi_\theta, z\sim q_\phi(z|c)}\left[D_{\text{KL}}\left(\pi_\theta(a|s,z)\,||\,\frac{\exp(Q_\theta(s,a,z))}{Z_\theta(s)}\right)\right] \tag{8.2}$$

where B is the replay buffer. The critic $Q_\theta(s,a,z)$ and the encoder $q_\phi(z|c)$ are trained with temporal difference learning:

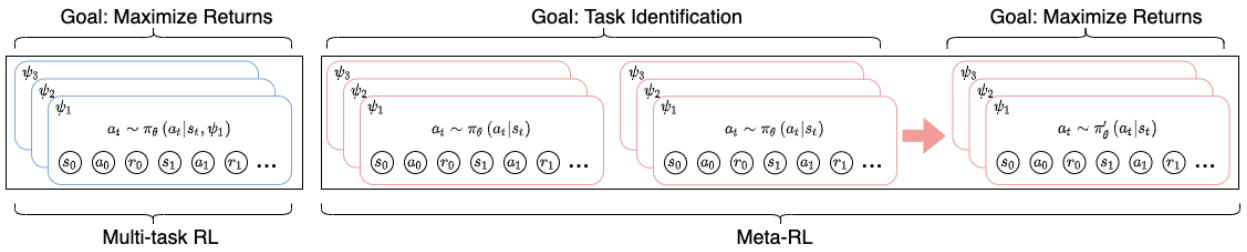


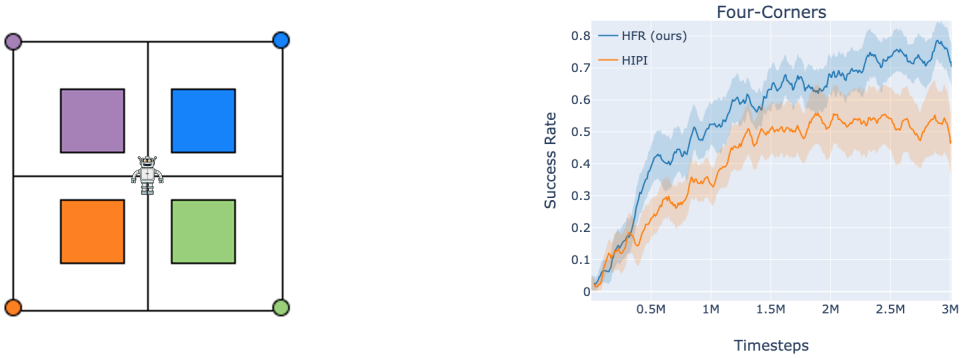
Figure 8.1: An illustration of the differences between multi-task RL and meta-RL. In multi-task RL (blue) the agent simply maximizes its returns given a task ψ, while in meta-RL (orange) the agent must first quickly identify the task with a limited number of exploratory trajectories (first two orange stacks in the figure), before adapting to the task and maximizing returns. Because of these differences, existing multi-task relabeling methods may be sub-optimal for meta-RL.

$$L_{\text{critic}} = \mathbb{E}_{(s,a,r,s')\sim B, z\sim q_\phi(z|c)}\left[\left(Q_\theta\left(s,a,z\right) - \left(r + \bar{V}\left(s',\bar{z}\right)\right)\right)^2\right] \tag{8.3}$$

where $\bar{V}$ is the target state value and $\bar{z}$ denotes that the gradient does not flow back through the latent.

8.3 HINDSIGHT FORESIGHT RELABELING

The objective in this section is to derive a formalism for data-sharing amongst the tasks during the meta-training phase. This is achieved via trajectory-relabeling, wherein a trajectory collected for a training task ψ^i is reused or re-purposed for training a different task ψ^j. Reward-based trajectory-relabeling has received a lot of attention in recent works on multi-task RL and goal-conditioned RL (Andrychowicz et al., 2017; Eysenbach et al., 2020; Li et al., 2020). The intuition is that if a trajectory τ collected while solving for the task ψ^i achieves high returns under the reward definition for another task ψ^j (*i.e.*, $\sum_t r_{\psi^j}(s_t, a_t)$ is large), then τ can be readily used for policy-optimization for the task ψ^j as well. The meta-RL setting presents the following subtlety – for any given task, the meta-RL agent generates trajectories with the aim of utilizing them in the adaptation procedure and subsequently seeks to maximize the post-adaptation returns (*cf.* §8.2.1). To improve the efficiency of the meta-training stage, we would like to share these pre-adaptation trajectories amongst the different tasks, accounting for the fact that the metric of interest with these trajectories is their usefulness for task-identification, rather than the returns (as in multi-task RL). This difference is illustrated in Figure 8.1. Hence, when deciding if a trajectory τ collected for task ψ^i is appropriate to be reused for task ψ^j, it is sub-optimal to consider the return value



(a) Schematic diagram

(b) Success-rate on Four-Corners

Figure 8.2: The Four-Corners environment

of this trajectory under ψ^j. Instead, we argue that this reuse compatibility should be determined based on the performance on the task ψ^j, *after* the agent has undergone adaptation using τ. Concretely, we define a function to measure the utility of the trajectory τ for a task ψ^j:

$$U_{\psi_j}(\tau) = \mathbb{E}_{s_t, a_t \sim \pi'} \left[\sum_{t=1}^{\infty} \gamma^{t-1} r_{\psi^j}(s_t, a_t) \right] \tag{8.4}$$

where $\pi' = f_\phi\left(\pi_\theta, \tau, r_{\psi^j}\right)$ denotes the policy after using τ for adaptation. The trajectory-relabeling mechanism during meta-training now incorporates this function U_{ψ^j}, which we refer to as the *utility function*, rather than the return R_{ψ^j}. Broadly, a trajectory τ collected for task ψ^i can be relabeled for use in another task ψ^j if $U_{\psi^j}(\tau)$ is high. Subsection §8.3.1 makes this more precise by deriving a relabeling distribution $q(\psi|\tau)$ that informs us of the tasks for which τ should be reused. Figure 8.3, and the caption therein, describe a high-level overview of our approach, HFR.

Comparison to HIPI (Eysenbach et al., 2020) with a didactic example. We consider a toy environment to further motivate that return-value based data sharing and trajectory relabeling (as proposed by HIPI) is potentially sub-optimal for meta-RL. The Four-Corners environment consists of a point robot placed at the center of a square where each corner of the square represents a goal location, as shown in Figure 8.2a. For each goal (task), there is a section of the space in the corresponding quadrant in which the robot receives a large negative reward. Consider a trajectory τ that hovers over the blue square in top-right quadrant. Note that τ could have been generated by the agent while collecting data for any of the four tasks. We examine if τ can be reused for the blue task. Since $R_{\text{blue}}(\tau)$ is highly negative, the relabeling strategy in HIPI does not reuse τ for meta-training on the blue

task. It is clearly evident, however, that τ carries a significant amount of signal pertaining to task-identification on the blue task, making it a useful pre-adaptation trajectory. HFR reuses τ for the blue task since the utility $U_{\text{blue}}(\tau)$ is high.

To quantify this effect, we include the numerical data on the returns and the utility values for a sampled trajectory that hovers over the blue square in top-right quadrant. The values for the returns $\{R_{\text{purple}}(\tau), R_{\text{blue}}(\tau), R_{\text{orange}}(\tau), R_{\text{green}}(\tau)\}$ are $\{-20, \mathbf{-58}, -20, -20\}$, while the utility values $\{U_{\text{purple}}(\tau), U_{\text{blue}}(\tau), U_{\text{orange}}(\tau), U_{\text{green}}(\tau)\}$ are $\{-1015, \mathbf{-756}, -935, -931\}$. These (unnormalized) numbers show that the probability of relabeling this trajectory with the blue task is low under HIPI, but high under HFR. Figure 8.2b compares HFR and HIPI in terms of the success-rate in the Four-Corners environment, and shows the performance benefit of using the utility function for trajectory relabeling.

8.3.1 Deriving a Meta-RL Relabeling Distribution

Our derivation in this subsection largely follows HIPI (Eysenbach et al., 2020), but differs in that we adapt it to the meta-RL setting to promote sharing of pre-adaptation trajectories amongst tasks, using the concept of trajectory utility. Assume a dataset $\mathcal{D}$ of trajectories gathered by the meta-RL agent when solving the different tasks in the meta-train task distribution. We wish to learn a trajectory relabeling distribution $q(\psi|\tau)$ such that, given any trajectory $\tau \sim \mathcal{D}$, we could reuse τ for tasks with high density under this posterior distribution. To that end, we start by defining a variational distribution $q(\tau|\psi)$ to designate the trajectories used for the adaptation process f_ϕ, for a given task ψ. Using the definition of the utility function (Equation 8.4), the meta-RL objective from Equation 8.1 could be written as: $\max_{\theta,\phi} \mathbb{E}_{\psi\sim p(\psi)}\mathbb{E}_{\tau\sim q(\tau|\psi)}[U_\psi(\tau)]$. For fixed meta-parameters (θ, ϕ), a natural approach to optimize the variational distribution $q(\tau|\psi)$ is to use this same objective since it facilitates alignment with the goals of the meta-learner. Thus, the combined objective for the variational distributions for *all* the tasks, augmented with entropy regularization, is:

$$\max_q \mathbb{E}_{\psi\sim p(\psi)}\Big[\mathbb{E}_{\tau\sim q(\tau|\psi)}[U_\psi(\tau)] + \mathcal{H}_{q(\tau|\psi)}\Big] \tag{8.5}$$

where $\mathcal{H}_{q(\tau|\psi)}$ denotes the causal entropy of the policy associated with $q(\tau|\psi)$. Now, it is easy to show that the above optimization is equivalent to a reverse-KL divergence minimization objective: $\min_{q(\tau,\psi)} D_{\text{KL}}\left[q\left(\tau,\psi\right)||p\left(\tau,\psi\right)\right]$. In this objective, the joint distributions over the tasks and the trajectories are defined as $q\left(\tau,\psi\right) = q(\tau|\psi)p(\psi)$ and $p\left(\tau,\psi\right) = p(\tau|\psi)p(\psi)$,

where

$$p(\tau|\psi) \triangleq \frac{1}{Z(\psi)} p_1(s_1) e^{U_\psi(\tau)} \prod_{t=1}^{T} \mathsf{p}(s_{t+1}|s_t, a_t) \tag{8.6}$$

Our goal is to formulate the trajectory relabeling distribution $q(\psi|\tau)$. To make this explicit in our objective, we use the trick proposed in HIPI (Eysenbach et al., 2020) and factor $q(\tau, \psi)$ as $q(\psi|\tau)\, q(\tau)$. This permits rewriting the reverse-KL minimization objective as:

$$\min_{q(\tau,\psi)} \mathbb{E}_{\substack{\tau\sim q(\tau) \\ \psi\sim q(\psi|\tau)}} \Big[\log q(\psi|\tau) + \log q(\tau) - \log p(\psi) + \log Z(\psi) - U_\psi(\tau) - \log p_1(s_1) - \sum_t \log \mathsf{p}(s_{t+1}|s_t, a_t) \Big] \tag{8.7}$$

Ignoring the terms independent of ψ, we can analytically solve (by differentiating and setting to zero) for the *optimal* trajectory relabeling distribution for the meta-RL setting:

$$q(\psi|\tau) \propto p(\psi)\, e^{U_\psi(\tau) - \log Z(\psi)} \tag{8.8}$$

Given a trajectory $\tau \sim \mathcal{D}$, the HFR algorithm uses this relabeling distribution to sample tasks for which τ should be reused. Concretely, we compute the utility function under τ for all the tasks, construct the distribution $q(\psi|\tau)$ using these utilities (Equation 8.8), and sample tasks from it. Please see Figure 8.3 for details. We assume a uniform prior $p(\psi)$ over the tasks in our experiments.

8.3.2 Algorithm and Implementation Details

Our relabeling algorithm is summarized in Algorithm 8.1 and fits seamlessly into the meta-training process of any of the base meta-RL algorithms. Once the meta-RL agent generates a trajectory τ for a training task, τ is fed as input to HFR, and it returns another task that can reuse this experience τ. We compute the utility of the input trajectory for every training task, along with an empirical estimate of the log-partition function of the tasks. The task to relabel the trajectory with is then sampled from a categorical distribution. For our experiments, we build on top of the PEARL algorithm (Rakelly et al., 2019), which is a data-efficient off-policy meta-RL method. PEARL maintains task-specific replay buffers B_ψ. If HFR returns the task ψ', then τ is relabeled using the reward function $r_{\psi'}$ and added to $B_{\psi'}$ for meta-training on the task ψ'.

The adaptation procedure $\pi' = f_\phi(\pi_\theta, \tau, r_\psi)$ for a task ψ corresponds to a sequence of steps: 1.) augment τ by marking each transition with a reward value computed using $r_\psi(s_t, a_t)$, 2.) condition the encoder on τ to sample an embedding, $z \sim q_\phi(z|\tau)$; and 3.) condition the policy on z to obtain the post-adaptation policy, $\pi' = \pi_\theta(\cdot|s, z)$. The calculation of

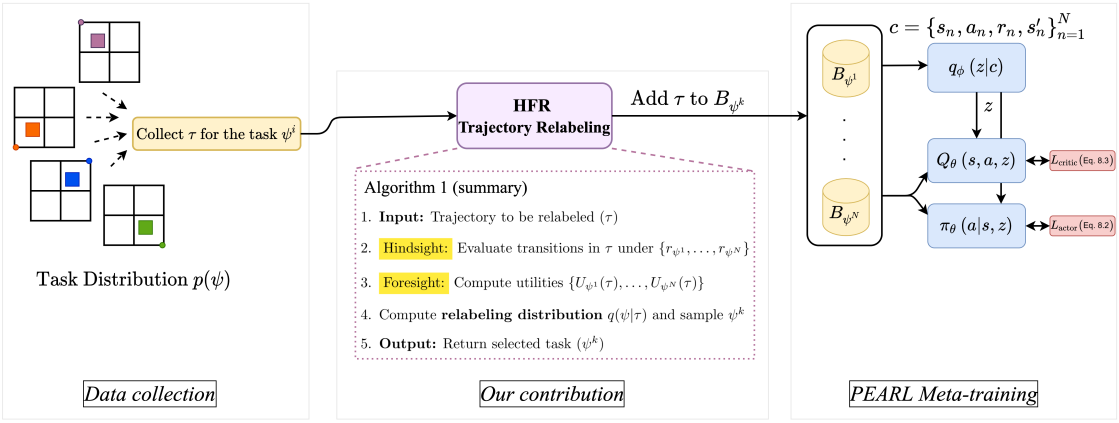


Figure 8.3: During meta-training, after a trajectory τ is collected for task ψ^i, HFR uses *hindsight* to relabel this trajectory using reward functions for different tasks, and then uses *foresight* to compute the utility of the relabeled trajectory for the different tasks. A distribution over tasks is constructed using the utilities, and a task ψ^k is sampled from the distribution, with tasks for which the trajectory has higher (normalized) utility having higher probability mass. The trajectory is then relabeled using the reward function r_{ψ^k} and added to the task-specific replay buffer B_{ψ^k}. Finally, the meta-training update rules are applied. This process repeats throughout the entirety of meta-training. HFR uses PEARL as the base meta-RL algorithm and does not alter its data collection or meta-gradient computation rules. Please see the Algorithm 8.1 box for details.

the utility function (Equation 8.4) requires generation of post-adaptation trajectories, which could be computationally inefficient, especially if the number of tasks is large. To avoid this cost, for each task, we sample a batch of initial states $s_1 \sim p_1(s_1)$ and the corresponding actions from the post-adaptation policy, and compute the utility based on an estimate of the state-action value function $Q_\psi^{\pi'}(s_1, a_1)$ as:

$$U_\psi(\tau) = \mathbb{E}_{s_1 \sim p_1, a_1 \sim \pi'(\cdot|s_1)} \left[Q_\psi^{\pi'}(s_1, a_1) \right] \tag{8.9}$$

Since we use PEARL, we can avoid training separate task-specific value functions Q_ψ, and instead get the required estimates from the task-conditioned critic $Q(s, a, z)$ already used by PEARL (Equation 8.3). We highlight that HFR facilitates efficient data-sharing among the training tasks via trajectory-relabeling *without* altering the meta-train and test-time adaptation rules of the base meta-RL algorithm.

8.4 EXPERIMENTS

The goal in this section is to quantitatively evaluate the benefit of sharing experience among tasks using HFR, during the meta-train stage. We evaluate on a set of both sparse and

Algorithm 8.1: Hindsight Foresight Relabeling (HFR)

Input : Trajectory to be relabeled (τ)
Output: Task to relabel the trajectory with (ψ)

for *each training task* ψ^i **do**
  $U_{\psi^i}(\tau) \leftarrow$ `ComputeUtility`(τ, ψ^i)
  $\log Z\left(\psi^i\right) \leftarrow$ `GetLogPartition`(ψ^i)
end
Return $\psi \sim \text{softmax}\{U_{\psi^i}(\tau) - \log Z(\psi^i)\}$ (Equation 8.8)

Function `GetLogPartition`(ψ)**:**
  Sample batch of trajectories $\left\{\tau^i\right\}_{i=1}^N \sim B_\psi$
  for *each trajectory* τ^i **do**
    $U_\psi\left(\tau^i\right) \leftarrow$ `ComputeUtility`(τ^i, ψ)
  end
  Return $\log Z(\psi) \approx \log\left(\frac{1}{N}\sum_{i=1}^N e^{U_\psi(\tau^i)}\right)$

Function `ComputeUtility`(τ, ψ)**:**
  for *each* $(s_t, a_t, r_t) \in \tau$ **do**
    Replace r_t with $r_\psi(s_t, a_t)$
  end
  Sample embedding using encoder $z \sim q_\phi(z|\tau)$
  Sample a batch of initial states $\left\{s_1^i\right\}_{i=1}^N \sim B_\psi$
  Sample actions for these states using the post-adaptation policy $\pi_\theta(\cdot|s, z)$: $\left\{a_1 \sim \pi_\theta\left(a_1^i|s_1^i, z\right)\right\}_{i=1}^N$
  Return $U_\psi = \frac{1}{N}\sum_{i=1}^N Q_\theta\left(s_1^i, a_1^i, z\right)$ (Equation 8.9)

dense reward MuJoCo environments (Todorov et al., 2012) modeled in OpenAI Gym (Brockman et al., 2016). We compare HFR with two relabeling methods: **Random**, in which each trajectory is relabeled with a randomly chosen task, and **HIPI** (Eysenbach et al., 2020), which utilizes MaxEnt RL to devise an algorithm that relabels each transition using a distribution over tasks involving the soft Q values for that transition. In contrast, HFR proposes the concept of the utility of a trajectory for a given task. Using the utility aligns the relabeling methodology with the objective of the meta-RL agent (*cf.* section 8.3). All methods are built on top of PEARL. Finally, we compare to PEARL with no relabeling at all, which we refer to as **None**.

8.4.1 Results

Sparse Reward Environments. We first evaluate HFR on a set of sparse reward robotic manipulation and locomotion tasks. We use five environments: a goal-reaching task involving a quadruped Ant robot, a pushing task on the Sawyer robot, a reaching task on the Sawyer robot, a velocity-matching task involving a bipedal Cheetah robot, and a reaching task involving the MuJoCo Reacher where the agent learns directly from images. Figure 8.4 plots the performance (average returns or success-rate) on the held-out meta-test tasks on the y-axis, with the total timesteps of environment interaction for meta-training on the x-axis. We note that HFR tends to be more sample-efficient than the baselines and achieves

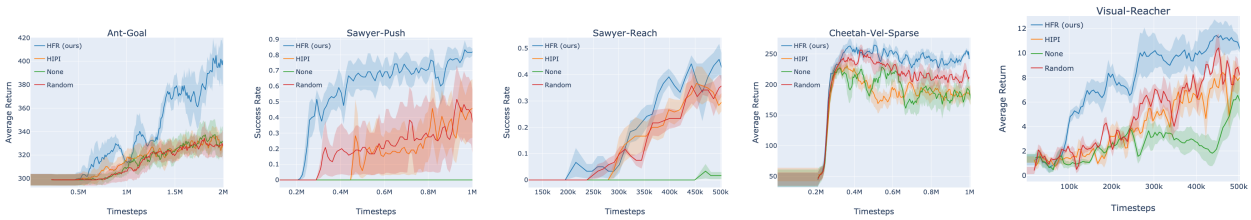


Figure 8.4: Performance of our relabeling algorithm HFR (shown in blue) on sparse reward tasks. HFR consistently outperforms baselines on both sparse reward robotic manipulation and locomotion tasks. Visual-Reacher uses image observations, while the other environments use proprioceptive states.

a higher asymptotic score. Meta-RL in sparse reward tasks is hard due to the challenges of task-identification and efficient exploration. HFR is especially useful for these tasks as the data-sharing afforded by the trajectory relabeling algorithm mitigates the need for an elaborate exploration strategy during meta-training. This leads to the sample-efficiency gains exhibited in Figure 8.4.

Dense Reward Environments. We next evaluate HFR on dense reward environments (Figure 8.5). We experiment with the Cheetah-Highdim environment, in which the bipedal robot is required to best match its state vector to a set of predefined vectors, as well as the Cheetah-Vel and quadruped Ant-Vel environments, in which the robots are required to run at various velocities. Note that the impact of HFR is much less pronounced for these environments, with the exception of Cheetah-Highdim. We believe this is because exploration is not as critical for these environments as it was for the sparse reward tasks. This hypothesis is supported by the fact that, in these environments, PEARL with no relabeling is competitive with the various relabeling methods, which all share similar performance, whereas in the sparse reward environments HFR is the relabeling method that performs best, with the two other relabeling methods also vastly outperforming vanilla PEARL. In the case of the Cheetah-Vel and Ant-Vel environments, agents are provided with an informative dense reward that immediately informs them as to which task they're supposed to be solving. Although the agent in Cheetah-Highdim is also provided with an informative dense reward, the reward function in this task is a linear combination of an 18-dimensional state vector. Thus, reasonably good exploration is needed to determine optimal values for each of these 18 dimensions. HFR relabeling provides improvement over the baselines for this task.

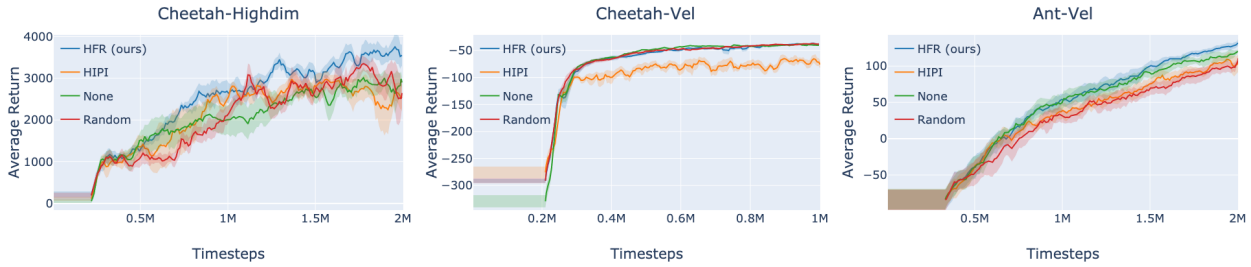


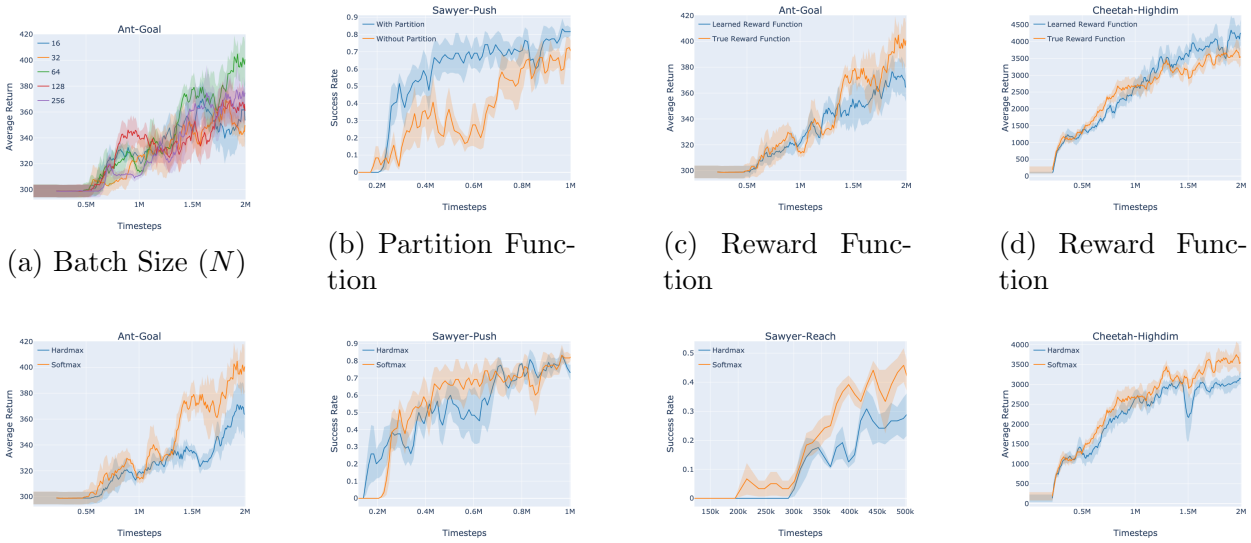
Figure 8.5: Performance of our relabeling algorithm HFR (shown in blue) on dense reward tasks. With the exception of Cheetah-Highdim, relabeling in general offers no benefit in dense reward meta-RL tasks, likely due to the highly informative nature of a dense reward function.

8.4.2 Ablation Studies

Batch Size. We investigate the impact of the batch size N used in Algorithm 8.1 for computing an empirical estimate of the state-action value: $\frac{1}{N}\sum_{i=1}^{N} Q_\theta\left(s_1^i, a_1^i, z\right)$. We compare the effect of batch size across batch sizes $\{16, 32, 64, 128, 256\}$. We expect lower values of N to lead to a higher variance estimate of the post-adaptation cumulative discounted rewards and thus potentially a worse approximation of the optimal meta-RL relabeling distribution. In Figure 8.6a, we see some evidence of this in the comparatively worse performance when using $N = 16$ and $N = 32$. However, we note that even with these small batch sizes, our method still performs well, and in general achieves high returns across all choices of N.

Partition Function. We investigate the impact of the log-partition function $\log Z(\psi)$ in the optimal meta-RL relabeling distribution (Equation 8.8). Prior work has noted the importance of the partition function in the multi-task RL setting when tasks may have different reward scales (Eysenbach et al., 2020). We believe that in the meta-RL setting, the partition function may be crucial even if the tasks share the reward scale, since some tasks in the meta-training distribution may be easier to solve than others. We speculate that with the omission of the partition function from our relabeling distribution, trajectories would find high utility and thus be disproportionately labeled with the easily-solved tasks, causing a degradation in overall performance. In our experiments, we find that the partition function serves as an essential normalization factor; Figure 8.6b shows an example.

Reward Function. One assumption our method assumes is access to the true reward function $r_\psi\left(s, a\right)$, which we can query to get the reward for an individual transition under any task ψ. This availability has also been utilized in existing works on relabeling for multi-task RL (Eysenbach et al., 2020; Li et al., 2020). In many real-world applications of meta-RL, *e.g.*,



(a) Batch Size (N)

(b) Partition Function

(c) Reward Function

(d) Reward Function

(e) Soft-max vs Hard-max

(f) Soft-max vs Hard-max

(g) Soft-max vs Hard-max

(h) Soft-max vs Hard-max

Figure 8.6: Ablation analyses. (a) HFR with different values for the batch size N used in approximating Equation 8.9. HFR is relatively robust to choice of batch size, with some smaller choices of N leading to slightly worse performance. (b) HFR with and without the log-partition function ($\log Z(\psi)$). The partition function serves as a necessary normalization factor and prevents against simply relabeling every trajectory using the easiest task. (c), (d) HFR with true reward function (orange) and HFR with learned reward function (blue). (e), (f), (g), (h) HFR with soft-max (orange) vs hard-max (blue) relabeling distribution

a distribution over robotic tasks, it is reasonable to assume that the task-designer outlines a rough template for the rewards corresponding to the different tasks from the distribution. Furthermore, several of the rewards used in our experiments are success/failure indicators, which are simple to specify. Nevertheless, we consider the scenario where we are unable to query the true reward function for individual transitions. In this case, we train a reward prediction network using the experience gathered by the agent. Figures 8.6c and 8.6d show good performance even when we use a learned reward function rather than the true reward function to relabel the trajectories.

Soft-max vs Hard-max Relabeling Distribution. Given a trajectory τ, the relabeling distribution derived in Section §8.3.1 samples tasks for which τ should be reused. Specifically, tasks are sampled as: $\psi \sim \texttt{softmax}\{U_{\psi^i}(\tau) - \log Z(\psi^i)\}$. This raises the following question: is it crucial to have stochasticity in the relabeling distribution, or could we deterministically select the task for which the utility value is the highest, *i.e.*, $\psi = \texttt{argmax}\{U_{\psi^i}(\tau) - \log Z(\psi^i)\}$?

A minor modification to the equations in Section §8.3.1 yields the hard-max relabeling

distribution. Concretely, we can add a term to the starting objective for the variational distribution q that explicitly *minimizes* the entropy of the relabeling distribution $q(\psi|\tau)$:

$$\max_{q} \mathbb{E}_{\psi\sim p(\psi)}\Big[\mathbb{E}_{\tau\sim q(\tau|\psi)}[U_{\psi}(\tau)] + \mathcal{H}_{q(\tau|\psi)}\Big] - (1-\epsilon)\mathbb{E}_{\tau\sim q(\tau)}\left[\mathcal{H}_{q(\psi|\tau)}\right] \tag{8.10}$$

where ϵ is a value less than 1. Proceeding with the derivation in the exact same manner as in Section §8.3.1, we obtain the adjusted relabeling distribution:

$$q_{\epsilon}\left(\psi|\tau\right) \propto e^{\frac{U_{\psi}(\tau) - \log Z(\psi)}{\epsilon}} \tag{8.11}$$

In the limit when $\epsilon \to 0$, $q_{\epsilon}(\psi|\tau)$ is the hard-max relabeling distribution. Figures 8.6e, 8.6f, 8.6g, and 8.6h compare the performance of HFR when sampling tasks from a soft-max relabeling distribution (orange) vs a hard-max distribution (blue). The results indicate that stochasticity is an important factor.

8.5 RELATED WORK AND CONCLUSION

Meta-learning, or learning to learn (Schmidhuber, 1987; Naik et al., 1992; Thrun & Pratt, 1998; Baxter, 1998), has been a topic of interest since the 1980s. Various approaches have been developed in recent years. Prior works have attempted to represent the RL process using a recurrent neural network (Duan et al., 2016; Wang et al., 2016; Miconi et al., 2018) - the hidden state is maintained across episode boundaries and informs the policy as to what task it is currently solving. Similarly, Mishra et al. (2017) also maintain the internal state across episode boundaries while incorporating temporal convolution and attention into a recursive architecture. Gradient-based meta-learning methods have also been explored (Finn et al., 2017; Nichol & Schulman, 2018; Xu et al., 2018; Zheng et al., 2020). MAML (Finn et al., 2017) seek to learn a good policy initialization so that only a few gradient steps are needed to achieve good performance on unseen meta-test tasks. Stadie et al. (2018) build on this gradient-based approach but explicitly consider the effect of the original sampling distribution on final performance. Similar to these works, our relabeling method also considers the impact of pre-adaptation data on the post-adaptation performance. Another body of work focuses on designing strategies for structured exploration in meta-RL such that task-relevant information could be efficiently recovered (Rakelly et al., 2019; Zintgraf et al., 2019; Liu et al., 2020). Rakelly et al. (2019) devise an off-policy meta-RL method called PEARL that trains an encoder to generate a latent code (with task-relevant information) on which the meta-RL agent is conditioned.

In the context of multi-task RL (Kaelbling, 1993; Caruana, 1997; Schaul et al., 2015),

recent methods have proposed relabeling to improve the sample-efficiency (Andrychowicz et al., 2017; Eysenbach et al., 2020; Li et al., 2020). HER (Andrychowicz et al., 2017) relabels transitions using goals that the agent actually achieves. Doing so allows for learning even with a sparse binary reward signal. However, HER is only applicable to goal-reaching tasks and cannot be incorporated into meta-RL algorithms because the meta-RL agent is trained on a batch of tasks sampled from a fixed task distribution. Our work is inspired by Eysenbach et al. (2020), who construct an optimal relabeling distribution for multi-task RL and apply this to both goal-reaching tasks and tasks with arbitrary reward functions.

Experience Relabeling in Meta-RL. Recent work has studied the scope of sharing experience among tasks in the meta-RL paradigm. Mendonca et al. (2020) propose to tackle meta-RL via a model identification process, where context-dependent neural networks parameterize the transition dynamics and the rewards function. Their method performs experience relabeling only at the meta-test time, with the purpose of consistent adaptation to the out-of-distribution tasks. Crucially, there is no relabeling or sharing of data amongst tasks during the meta-train time. In contrast, the goal of the relabeling in HFR is to improve the sample efficiency of the meta-training phase. Dorfman et al. (2020) study the offline meta-RL problem. They propose reward relabeling as a mechanism to mitigate the "MDP ambiguity" issue, which the authors note is specific to the offline meta-RL setting. Their relabeling is based on random task selection. HFR, on the other hand, operates in the online meta-RL setting and provides a principled approach to compute a relabeling distribution that suggests tasks for relabeling.

8.5.1 Conclusion

In this work, we introduced HFR, a trajectory relabeling method for meta-RL that enables data sharing between tasks during meta-train. We argue that unlike the multi-task RL setting, where the appropriateness of a trajectory for a task could be measured by the returns under the task, for meta-RL, it is preferable to consider the future (expected) task-returns of an agent adapted using that trajectory. We capture this notion by defining the *utility* function for a trajectory-task pair and incorporate these utilities in our relabeling mechanism. Inspired by prior work on multi-task RL, an optimal relabeling distribution is then derived that informs us of the tasks for which a generated trajectory should be reused. *Hindsight* is used to relabel trajectories with different reward functions, while *foresight* is used in computing the utility of each trajectory under different tasks and constructing a relabeling distribution. HFR is easy to implement, can be integrated into any existing meta-RL algorithm, and yields improvement on a variety of meta-RL tasks with sparse rewards.

APPENDIX A:

A.1 MISSING PROOFS IN CHAPTER 4

This section includes proofs for the inequalities mentioned in Section 4.2.1. We first prove the inequality connecting D_{JS} between the state-visitation distribution and belief-visitation distribution of the agent and the expert:

$$D_{JS}[\rho_\pi(s) \mid\mid \rho_E(s)] \leq D_{JS}[\rho_\pi(b) \mid\mid \rho_E(b)] \tag{A.1}$$

Proof. The proof is a simple application of the data-processing inequality for f-divergences (Ali & Silvey, 1966), of which D_{JS} is a type.

We denote the filtering posterior distribution over states, given the belief, by $p(s|b)$. Note that $p(s|b)$ is characterized by the environment, and does not depend on the policy (agent or expert). The posterior over belief, given the state, however, is policy-dependent and obtained using Bayes rule as: $p_\pi(b|s) = \frac{p(s|b)\rho_\pi(b)}{\rho_\pi(s)}$. Also, $\rho_\pi(s,b) = \rho_\pi(s)p_\pi(b|s) = \rho_\pi(b)p(s|b)$. Analogously definitions exist for expert E.

We write $D_{JS}[\rho_\pi(b) \mid\mid \rho_E(b)]$ in terms of the template used for f-divergences. Let $f : (0,\infty) \mapsto \mathbb{R}$ be the following convex function with the property $f(1) = 0$: $f(u) = -(u+1)\log\frac{1+u}{2} + u\log u$. Then,

$$\begin{aligned}
&D_{JS}[\rho_\pi(b) \mid\mid \rho_E(b)] \\
&\quad= \mathbb{E}_{b\sim\rho_E(b)}\left[f(\frac{\rho_\pi(b)}{\rho_E(b)})\right] \\
&\quad= \mathbb{E}_{s,b\sim\rho_E(s,b)}\left[f(\frac{\rho_\pi(s,b)}{\rho_E(s,b)})\right] \\
&\quad= \mathbb{E}_{s\sim\rho_E(s)}\left[\mathbb{E}_{b\sim\rho_E(b|s)} f(\frac{\rho_\pi(s,b)}{\rho_E(s,b)})\right] \\
&\quad\geq \mathbb{E}_{s\sim\rho_E(s)}\left[f\left(\mathbb{E}_{b\sim\rho_E(b|s)}\frac{\rho_\pi(s,b)}{\rho_E(s,b)}\right)\right] \quad \textit{(Jensen's inequality)} \\
&\quad= \mathbb{E}_{s\sim\rho_E(s)}\left[f\left(\mathbb{E}_{b\sim\rho_\pi(b|s)}\frac{\rho_\pi(s,b)\rho_E(b|s)}{\rho_E(s,b)\rho_\pi(b|s)}\right)\right] \\
&\quad= \mathbb{E}_{s\sim\rho_E(s)}\left[f\left(\mathbb{E}_{b\sim\rho_\pi(b|s)}\frac{\rho_\pi(s)}{\rho_E(s)}\right)\right] \\
&\quad= \mathbb{E}_{s\sim\rho_E(s)}\left[f(\frac{\rho_\pi(s)}{\rho_E(s)})\right] \\
&\quad= D_{JS}[\rho_\pi(s) \mid\mid \rho_E(s)]
\end{aligned} \tag{A.2}$$

QED.

Similarity, we can prove the inequality connecting D_{JS} between belief-visitation distribution and belief-action-visitation distribution of the agent and the expert:

$$D_{JS}[\rho_\pi(b) \;||\; \rho_E(b)] \leq D_{JS}[\rho_\pi(b,a) \;||\; \rho_E(b,a)] \tag{A.3}$$

Proof. Replace $s \mapsto b'$ and $b \mapsto (b,a)$ in the previous proof. The only *required* condition for that result to hold is the non-dependence of the distribution $p(s|b)$ on the policy. Therefore, if it holds that $p(b'|b,a)$ is independent of the policy, then we have,

$$D_{JS}[\rho_\pi(b') \;||\; \rho_E(b')] \leq D_{JS}[\rho_\pi(b,a) \;||\; \rho_E(b,a)] \tag{A.4}$$

The independence holds under the trivial case of a deterministic mapping $b'{=}b$. This gives us the desired inequality. QED.

A.2 MISSING PROOFS IN CHAPTER 6

A.2.1 Gradient of Divergences *w.r.t.* the Policy Parameters

We derive the expressions for $\nabla_{\theta_j} D_{\text{JS}}$ and $\nabla_{\theta_j} D_{\text{KLS}}$ mentioned in Equation 6.8. θ_j denotes the parameters for π_j. The distribution ratio, $\zeta_{ij} = \rho_i/\rho_j$, depends on θ_j through ρ_j. A bar above a symbol signifies that it is a constant *w.r.t.* θ_j; for instance, while ρ_j depends on θ_j, $\bar{\rho}_j$ does not. The derivation uses the property that the expectation of the score function estimator is 0.

Jenson-Shannon divergence:

$$D_{\text{JS}}(\rho_i, \rho_j) = \frac{1}{2}\mathbb{E}_{\rho_i} \log \frac{\rho_i}{\rho_i + \rho_j} + \frac{1}{2}\mathbb{E}_{\rho_j} \log \frac{\rho_j}{\rho_i + \rho_j} + \log 2 \tag{A.5}$$

Differentiating with the product rule,

$$
\begin{aligned}
\nabla_{\theta_j} 2D_{\text{JS}} =& -\mathbb{E}_{\rho_i} \nabla_{\theta_j} \log[\rho_i + \rho_j] + \underbrace{\mathbb{E}_{\rho_j} \nabla_{\theta_j} \log[\rho_j]}_{\substack{=0 \\ \text{Exp. score function}}} + \nabla_{\theta_j} \mathbb{E}_{\rho_j} \log[\bar{\rho}_j] \\
& - \mathbb{E}_{\rho_j} \nabla_{\theta_j} \log[\rho_i + \rho_j] - \nabla_{\theta_j} \mathbb{E}_{\rho_j} \log[\rho_i + \bar{\rho}_j] \\
=& - \underbrace{\mathbb{E}_{\rho_i + \rho_j} \nabla_{\theta_j} \log[\rho_i + \rho_j]}_{\substack{=0 \\ \text{Exp. score function}}} + \nabla_{\theta_j} \mathbb{E}_{\rho_j} \log[\bar{\rho}_j] - \nabla_{\theta_j} \mathbb{E}_{\rho_j} \log[\rho_i + \bar{\rho}_j]
\end{aligned} \tag{A.6}
$$

$$
\nabla_{\theta_j} D_{\text{JS}} = -(1/2) \nabla_{\theta_j} \mathbb{E}_{\rho_j} \log[1 + \bar{\zeta}_{ij}]
$$

Symmetric Kullback-Leibler divergence:

$$
D_{\text{KLS}}(\rho_i, \rho_j) = \mathbb{E}_{\rho_i} \log \frac{\rho_i}{\rho_j} - \mathbb{E}_{\rho_j} \log \frac{\rho_i}{\rho_j} \tag{A.7}
$$

Differentiating with the product rule,

$$
\nabla_{\theta_j} D_{\text{KLS}} = -\mathbb{E}_{\rho_i} \nabla_{\theta_j} \log[\rho_j] - \nabla_{\theta_j} \mathbb{E}_{\rho_j} \log \bar{\zeta}_{ij} + \underbrace{\mathbb{E}_{\rho_j} \nabla_{\theta_j} \log[\rho_j]}_{\substack{=0 \\ \text{Exp. score function}}} \tag{A.8}
$$

For the first term, interchanging the gradient and the expectation, we can write:

$$
\mathbb{E}_{\rho_i} \nabla_{\theta_j} \log[\rho_j] = \sum_{(s,a)} \rho_i \frac{\nabla_{\theta_j} \rho_j}{\bar{\rho}_j} = \sum_{(s,a)} \bar{\zeta}_{ij} \nabla_{\theta_j} \rho_j = \nabla_{\theta_j} \mathbb{E}_{\rho_j}[\bar{\zeta}_{ij}] \tag{A.9}
$$

Therefore,

$$
\nabla_{\theta_j} D_{\text{KLS}} = \nabla_{\theta_j} \mathbb{E}_{\rho_j}[-\bar{\zeta}_{ij} - \log \bar{\zeta}_{ij}] \tag{A.10}
$$

A.2.2 Optimality in the Donsker-Varadhan Representation

The Donsker-Varadhan representation (Donsker & Varadhan, 1983) of the KL-divergence is given by:

$$
D_{\text{KL}}(\rho_i || \rho_j) = \sup_{x: \mathcal{S} \times \mathcal{A} \to \mathbb{R}} \mathbb{E}_{(s,a) \sim \rho_i}[x(s,a)] - \log \mathbb{E}_{(s,a) \sim \rho_j}[e^{x(s,a)}] \tag{A.11}
$$

The optimality is achieved at $x^*(s,a) = \log \zeta_{ij}(s,a) + C$, for some constant $C \in \mathbb{R}$.

Proof. We begin with a re-write of the expression inside the supremum:

$$\begin{aligned}
&\mathbb{E}_{(s,a)\sim\rho_i}\left(\log\left[e^{x(s,a)}.\frac{\rho_i}{\rho_j}.\frac{\rho_j}{\rho_i}\right]\right) - \log\mathbb{E}_{(s,a)\sim\rho_j}[e^{x(s,a)}] \\
&= \underbrace{\mathbb{E}_{(s,a)\sim\rho_i}[\log\frac{\rho_i}{\rho_j}]}_{\mathrm{KL}} + \mathbb{E}_{(s,a)\sim\rho_i}\left(\log\left[\frac{\rho_j}{\rho_i}.e^{x(s,a)}\right]\right) - \log\mathbb{E}_{(s,a)\sim\rho_j}[e^{x(s,a)}] \\
&\leq D_{\mathrm{KL}}(\rho_i||\rho_j) + \log\mathbb{E}_{(s,a)\sim\rho_i}\left[\frac{\rho_j}{\rho_i}.e^{x(s,a)}\right] - \log\mathbb{E}_{(s,a)\sim\rho_j}[e^{x(s,a)}] \qquad \text{(Jensen's inequality)} \\
&= D_{\mathrm{KL}}(\rho_i||\rho_j) + \log\mathbb{E}_{(s,a)\sim\rho_j}[e^{x(s,a)}] - \log\mathbb{E}_{(s,a)\sim\rho_j}[e^{x(s,a)}] \\
&= D_{\mathrm{KL}}(\rho_i||\rho_j)
\end{aligned} \tag{A.12}$$

Therefore, this expression is upper bounded by $D_{\mathrm{KL}}(\rho_i||\rho_j)$. To complete the proof, we show that this upper bound is indeed achieved when $x(s,a) = \log\zeta_{ij}(s,a) + C$, for some constant $C \in \mathbb{R}$. Inserting this into the expression, we get:

$$\begin{aligned}
&\mathbb{E}_{(s,a)\sim\rho_i}[\log\zeta_{ij}(s,a) + C] - \log\mathbb{E}_{(s,a)\sim\rho_j}[e^{\log\zeta_{ij}(s,a)+C}] \\
&= D_{\mathrm{KL}}(\rho_i||\rho_j) + C - \log\left(e^C \underbrace{\mathbb{E}_{(s,a)\sim\rho_j}[\zeta_{ij}(s,a)]}_{=1}\right) \\
&= D_{\mathrm{KL}}(\rho_i||\rho_j)
\end{aligned} \tag{A.13}$$

QED.

REFERENCES

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

[2] Daniel A Abolafia, Mohammad Norouzi, and Quoc V Le. Neural program synthesis with priority queue training. *arXiv preprint arXiv:1801.03526*, 2018.

[3] M Mehdi Afsar, Trafford Crump, and Behrouz Far. Reinforcement learning based recommender systems: A survey. *arXiv preprint arXiv:2101.06286*, 2021.

[4] Felix Agakov and David Barber. Variational information maximization for neural coding. In *International Conference on Neural Information Processing*, pp. 543–548. Springer, 2004.

[5] Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9163–9171, 2019.

[6] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[7] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

[8] Syed Mumtaz Ali and Samuel D Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society: Series B (Methodological)*, 28(1):131–142, 1966.

[9] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

[10] Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems*, pp. 13544–13555, 2019.

[11] Dilip Arumugam, Debadeepta Dey, Alekh Agarwal, Asli Celikyilmaz, Elnaz Nouri, and Bill Dolan. Reparameterized variational divergence minimization for stable imitation. *arXiv preprint arXiv:2006.10810*, 2020.

[12] K. J. Aström. Optimal control of Markov decision processes with incomplete state estimation. *J. Math. Anal. Appl.*, 10:174–205, 1965.

[13] Fahiem Bacchus, Craig Boutilier, and Adam Grove. Rewarding behaviors. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 1160–1167, 1996.

[14] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1606.05312*, 2016.

[15] Jonathan Baxter. Theoretical models of learning to learn. In *Learning to learn*, pp. 71–94. Springer, 1998.

[16] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 449–458. JMLR. org, 2017.

[17] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[18] Kiante Brantley, Wen Sun, and Mikael Henaff. Disagreement-regularized imitation learning. In *International Conference on Learning Representations*, 2019.

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

[20] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[21] Lars Buesing, Theophane Weber, Sebastien Racaniere, SM Eslami, Danilo Rezende, David P Reichert, Fabio Viola, Frederic Besse, Karol Gregor, Demis Hassabis, et al. Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*, 2018.

[22] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[23] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[24] Tao Chen, Adithyavairavan Murali, and Abhinav Gupta. Hardware conditioned policies for multi-robot transfer learning. In *Advances in Neural Information Processing Systems*, pp. 9333–9344, 2018.

[25] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[26] Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. *arXiv preprint arXiv:1712.06560*, 2017.

[27] Antoine Cully and Yiannis Demiris. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259, 2017.

[28] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

[29] Wojciech Marian Czarnecki, Siddhant M Jayakumar, Max Jaderberg, Leonard Hasenclever, Yee Whye Teh, Simon Osindero, Nicolas Heess, and Razvan Pascanu. Mix&match-agent curricula for reinforcement learning. *arXiv preprint arXiv:1806.01780*, 2018.

[30] Wojciech Marian Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant M Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. *arXiv preprint arXiv:1902.02186*, 2019.

[31] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. *arXiv preprint arXiv:1806.06923*, 2018.

[32] Ernesto De Vito, Lorenzo Rosasco, and Alessandro Toigo. A universally consistent spectral estimator for the support of a distribution. *Appl Comput Harmonic Anal*, 37: 185–217, 2014.

[33] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

[34] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Openai baselines. https://github.com/openai/baselines, 2017.

[35] Thang Doan, Bogdan Mazoure, Audrey Durand, Joelle Pineau, and R Devon Hjelm. Attraction-repulsion actor-critic for continuous control reinforcement learning. *arXiv preprint arXiv:1909.07543*, 2019.

[36] Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time. iv. *Communications on Pure and Applied Mathematics*, 36(2):183–212, 1983.

[37] Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline meta learning of exploration. *arXiv preprint arXiv:2008.02598*, 2020.

[38] Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016.

[39] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pp. 1329–1338, 2016.

[40] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[41] Ashley Edwards, Himanshu Sahni, Yannick Schroecker, and Charles Isbell. Imitating latent policies from observation. In *International Conference on Machine Learning*, pp. 1755–1763. PMLR, 2019.

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

[43] Benjamin Eysenbach, Xinyang Geng, Sergey Levine, and Ruslan Salakhutdinov. Rewriting history with inverse rl: Hindsight inference for policy improvement. *arXiv preprint arXiv:2002.11089*, 2020.

[44] Rasool Fakoor, Pratik Chaudhari, Stefano Soatto, and Alexander J Smola. Meta-q-learning. *arXiv preprint arXiv:1910.00125*, 2019.

[45] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

[46] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pp. 49–58, 2016.

[47] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

[48] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

[49] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Advances in neural information processing systems*, pp. 2199–2207, 2016.

[50] Scott Fujimoto, Herke van Hoof, and Dave Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[51] Shani Gamrian and Yoav Goldberg. Transfer learning for related reinforcement learning tasks via image-to-image translation. *arXiv preprint arXiv:1806.07377*, 2018.

[52] Tanmay Gangwani and Jian Peng. Policy optimization by genetic distillation. *arXiv preprint arXiv:1711.01012*, 2017.

[53] Tanmay Gangwani and Jian Peng. State-only imitation with transition dynamics mismatch. *arXiv preprint arXiv:2002.11879*, 2020.

[54] Tanmay Gangwani, Qiang Liu, and Jian Peng. Learning self-imitating diverse policies. *arXiv preprint arXiv:1805.10309*, 2018.

[55] Tanmay Gangwani, Joel Lehman, Qiang Liu, and Jian Peng. Learning belief representations for imitation learning in pomdps. *arXiv preprint arXiv:1906.09510*, 2019.

[56] Tanmay Gangwani, Jian Peng, and Yuan Zhou. Harnessing distribution ratio estimators for learning agents with quality and diversity. *arXiv preprint arXiv:2011.02614*, 2020.

[57] Tanmay Gangwani, Yuan Zhou, and Jian Peng. Learning guidance rewards with trajectory-space smoothing. *arXiv preprint arXiv:2010.12718*, 2020.

[58] Tanmay Gangwani, Yuan Zhou, and Jian Peng. Imitation learning from observations under transition model disparity. under submission, 2021.

[59] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1243–1252. JMLR. org, 2017.

[60] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[61] Anirudh Goyal ALIAS PARTH Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. Z-forcing: Training stochastic recurrent networks. In *Advances in neural information processing systems*, pp. 6713–6723, 2017.

[62] Karol Gregor, George Papamakarios, Frederic Besse, Lars Buesing, and Theophane Weber. Temporal difference variational auto-encoder. *arXiv preprint arXiv:1806.03107*, 2018.

[63] Yijie Guo, Junhyuk Oh, Satinder Singh, and Honglak Lee. Generative adversarial self-imitation learning. *arXiv preprint arXiv:1812.00950*, 2018.

[64] Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Bernardo A Pires, Toby Pohlen, and Rémi Munos. Neural predictive belief representations. *arXiv preprint arXiv:1811.06407*, 2018.

[65] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017.

[66] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 297–304, 2010.

[67] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pp. 2455–2467, 2018.

[68] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[69] Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Gheshlaghi Azar, Bilal Piot, Nicolas Heess, Hado P van Hasselt, Gregory Wayne, Satinder Singh, Doina Precup, et al. Hindsight credit assignment. In *Advances in neural information processing systems*, pp. 12467–12476, 2019.

[70] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015.

[71] Milos Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of artificial intelligence research*, 13:33–94, 2000.

[72] Ahmed Hefny, Zita Marinho, Wen Sun, Siddhartha Srinivasa, and Geoffrey Gordon. Recurrent predictive state policy networks. *arXiv preprint arXiv:1803.01489*, 2018.

[73] Daniel Hein, Stefan Depeweg, Michel Tokic, Steffen Udluft, Alexander Hentschel, Thomas A Runkler, and Volkmar Sterzing. A benchmark environment motivated by industrial control problems. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8. IEEE, 2017.

[74] Matthias Hein and Olivier Bousquet. Hilbertian metrics and positive definite kernels on probability measures. 2004.

[75] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, et al. Deep q-learning from demonstrations. *arXiv preprint arXiv:1704.03732*, 2017.

[76] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[77] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pp. 4565–4573, 2016.

[78] Zhang-Wei Hong, Tzu-Yun Shann, Shih-Yang Su, Yi-Hsiang Chang, Tsu-Jui Fu, and Chun-Yi Lee. Diversity-driven exploration strategy for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 10489–10500, 2018.

[79] Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. Optimizing agent behavior over long time scales by transporting value. *Nature communications*, 10(1):1–12, 2019.

[80] Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. *arXiv preprint arXiv:1806.02426*, 2018.

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

[82] Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pp. 1094–1099. Citeseer, 1993.

[83] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[84] Liyiming Ke, Matt Barnes, Wen Sun, Gilwoo Lee, Sanjiban Choudhury, and Siddhartha Srinivasa. Imitation learning as f-divergence minimization. *arXiv preprint arXiv:1905.12888*, 2019.

[85] Shauharda Khadka, Somdeb Majumdar, Santiago Miret, Stephen McAleer, and Kagan Tumer. Evolutionary reinforcement learning for sample-efficient multiagent coordination. *arXiv preprint arXiv:1906.07315*, 2019.

[86] Shauharda Khadka, Somdeb Majumdar, Tarek Nassar, Zach Dwiel, Evren Tumer, Santiago Miret, Yinyin Liu, and Kagan Tumer. Collaborative evolutionary reinforcement learning. *arXiv preprint arXiv:1905.00976*, 2019.

[87] Shauharda Khadka, Estelle Aflalo, Mattias Marder, Avrech Ben-David, Santiago Miret, Shie Mannor, Tamir Hazan, Hanlin Tang, and Somdeb Majumdar. Optimizing memory placement using evolutionary graph reinforcement learning. *arXiv preprint arXiv:2007.07298*, 2020.

[88] Jack PC Kleijnen and Reuven Y Rubinstein. Optimization and sensitivity analysis of computer simulation models by the score function method. *European Journal of Operational Research*, 88(3):413–427, 1996.

[89] Jens Kober and Jan R Peters. Policy search for motor primitives in robotics. In *Advances in neural information processing systems*, pp. 849–856, 2009.

[90] Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. *arXiv preprint arXiv:1912.05032*, 2019.

[91] Tadashi Kozuno, Yunhao Tang, Mark Rowland, Rémi Munos, Steven Kapturowski, Will Dabney, Michal Valko, and David Abel. Revisiting peng's q (λ) for modern reinforcement learning. *arXiv preprint arXiv:2103.00107*, 2021.

[92] Su Young Lee, Choi Sungik, and Sae-Young Chung. Sample-efficient deep reinforcement learning via episodic backward update. In *Advances in Neural Information Processing Systems*, pp. 2110–2119, 2019.

[93] Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, pp. 1–9, 2013.

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

[95] Alexander C Li, Lerrel Pinto, and Pieter Abbeel. Generalized hindsight for reinforcement learning. *arXiv preprint arXiv:2002.11708*, 2020.

[96] Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*, 2016.

[97] Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc Le, and Ni Lao. Memory augmented policy optimization for program synthesis with generalization. *arXiv preprint arXiv:1807.02322*, 2018.

[98] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[99] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

[100] Michael L Littman and Richard S Sutton. Predictive representations of state. In *Advances in neural information processing systems*, pp. 1555–1561, 2002.

[101] Evan Zheran Liu, Aditi Raghunathan, Percy Liang, and Chelsea Finn. Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices. *arXiv preprint arXiv:2008.02790*, 2020.

[102] Fangchen Liu, Zhan Ling, Tongzhou Mu, and Hao Su. State alignment-based imitation learning. *arXiv preprint arXiv:1911.10947*, 2019.

[103] Iou-Jen Liu, Jian Peng, and Alexander G Schwing. Knowledge flow: Improve upon your teachers. *arXiv preprint arXiv:1904.05878*, 2019.

[104] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pp. 2378–2386, 2016.

[105] Siqi Liu, Guy Lever, Josh Merel, Saran Tunyasuvunakool, Nicolas Heess, and Thore Graepel. Emergent coordination through competition. *arXiv preprint arXiv:1902.07151*, 2019.

[106] Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein variational policy gradient. *arXiv preprint arXiv:1704.02399*, 2017.

[107] Yang Liu, Yunan Luo, Yuanyi Zhong, Xi Chen, Qiang Liu, and Jian Peng. Sequence modeling of temporal credit assignment for episodic reinforcement learning. *arXiv preprint arXiv:1905.13420*, 2019.

[108] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *arXiv preprint arXiv:1904.08473*, 2019.

[109] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1118–1125. IEEE, 2018.

[110] Shie Mannor, Reuven Y Rubinstein, and Yohai Gat. The cross entropy method for fast policy search. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 512–519, 2003.

[111] Russell Mendonca, Xinyang Geng, Chelsea Finn, and Sergey Levine. Meta-reinforcement learning robust to distributional shift via model identification and experience relabeling. *arXiv preprint arXiv:2006.07178*, 2020.

[112] Josh Merel, Yuval Tassa, Dhruva TB, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017.

[113] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pp. 3481–3490. PMLR, 2018.

[114] Thomas Miconi, Kenneth Stanley, and Jeff Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *International Conference on Machine Learning*, pp. 3559–3568. PMLR, 2018.

[115] Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1): 8–30, 1961.

[116] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[117] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[118] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529, 2015.

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

[120] Pedro J Moreno, Purdy P Ho, and Nuno Vasconcelos. A kullback-leibler divergence based kernel for svm classification in multimedia applications. In *Advances in neural information processing systems*, pp. 1385–1392, 2004.

[121] Pol Moreno, Jan Humplik, George Papamakarios, Bernardo Avila Pires, Lars Buesing, Nicolas Heess, and Théophane Weber. Neural belief states for partially observed domains. In *NeurIPS 2018 workshop on Reinforcement Learning under Partial Observability*, 2018.

[122] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

[123] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Advances in Neural Information Processing Systems*, pp. 2318–2328, 2019.

[124] DK Naik, Richard Mammone, and A Agarwal. Meta-neural network approach to learning by learning. In *Proceedings of the 1992 Artificial Neural Networks in Engineering, ANNIE'92*, 1992.

[125] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. *arXiv preprint arXiv:1709.10089*, 2017.

[126] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pp. 278–287, 1999.

[127] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, pp. 663–670, 2000.

[128] XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

[129] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(2):1, 2018.

[130] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in neural information processing systems*, pp. 2863–2871, 2015.

[131] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. *arXiv preprint arXiv:1806.05635*, 2018.

[132] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9 (1):48, 2017.

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

[134] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

[135] Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow. *arXiv preprint arXiv:1810.00821*, 2018.

[136] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[137] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750. ACM, 2007.

[138] Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *AAAI*, pp. 1607–1612. Atlanta, 2010.

[139] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, volume 3, pp. 1025–1032, 2003.

[140] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

[141] Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, pp. 80, 2000.

[142] Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adria Puigdomenech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2827–2836. JMLR. org, 2017.

[143] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.

[144] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994. ISBN 0471619779.

[145] Yu Qiao and Nobuaki Minematsu. A study on invariance of f-divergence and its application to speech recognition. *IEEE Transactions on Signal Processing*, 58(7): 3884–3890, 2010.

[146] Hazhir Rahmandad, Nelson Repenning, and John Sterman. Effects of feedback delay on learning. *System Dynamics Review*, 25(4):309–338, 2009.

[147] Aida Rahmattalabi, Jen Jen Chung, Mitchell Colby, and Kagan Tumer. D++: Structural credit assignment in tightly coupled multiagent domains. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4424–4429. IEEE, 2016.

[148] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[149] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pp. 5331–5340, 2019.

[150] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[151] Stephane Ross and J Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

[152] Stéphane Ross, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-Draa. Online planning algorithms for pomdps. *Journal of Artificial Intelligence Research*, 32:663–704, 2008.

[153] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, 2011.

[154] Stéphane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J Andrew Bagnell, and Martial Hebert. Learning monocular reactive uav control in cluttered natural environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 1765–1772. IEEE, 2013.

[155] Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search. *arXiv preprint arXiv:1810.06784*, 2018.

[156] Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. Beyond sharing weights for deep domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 41(4):801–814, 2018.

[157] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[158] Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*, volume 10. John Wiley & Sons, 2016.

[159] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[160] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[161] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[162] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320. PMLR, 2015.

[163] Jurgen Schmidhuber. Evolutionary principles in self-referential learning. *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, 1(2), 1987.

[164] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pp. 222–227, 1991.

[165] Jürgen Schmidhuber. Artificial curiosity based on discovering novel algorithmic predictability through coevolution. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3, pp. 1612–1618. IEEE, 1999.

[166] Simon Schmitt, Jonathan J Hudson, Augustin Zidek, Simon Osindero, Carl Doersch, Wojciech M Czarnecki, Joel Z Leibo, Heinrich Kuttler, Andrew Zisserman, Karen Simonyan, et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

[167] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, pp. 3528–3536, 2015.

[168] John Schulman, Sergey Levine, Pieter Abbeel, Michael I Jordan, and Philipp Moritz. Trust region policy optimization. In *Icml*, volume 37, pp. 1889–1897, 2015.

[169] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

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

[172] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1134–1141. IEEE, 2018.

[173] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2014.

[174] Hassam Sheikh, Shauharda Khadka, Santiago Miret, and Somdeb Majumdar. Learning intrinsic symbolic rewards in reinforcement learning. *arXiv preprint arXiv:2010.03694*, 2020.

[175] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pp. 2164–2172, 2010.

[176] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. 2014.

[177] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[178] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[179] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[180] Satinder Singh, Richard L Lewis, and Andrew G Barto. Where do rewards come from. In *Proceedings of the annual conference of the cognitive science society*, pp. 2601–2606. Cognitive Science Society, 2009.

[181] Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

[182] Jonathan Sorg, Satinder P Singh, and Richard L Lewis. Internal rewards mitigate agent boundedness. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 1007–1014, 2010.

[183] Jonathan Sorg, Satinder Singh, and Richard L Lewis. Optimal rewards versus leaf-evaluation heuristics in planning agents. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[184] Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. Third-person imitation learning. *arXiv preprint arXiv:1703.01703*, 2017.

[185] Bradly C Stadie, Ge Yang, Rein Houthooft, Xi Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*, 2018.

[186] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.

[187] Wen Sun, Anirudh Vemula, Byron Boots, and J Andrew Bagnell. Provably efficient imitation learning from observation alone. *arXiv preprint arXiv:1905.10948*, 2019.

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

[189] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

[190] Richard S Sutton, A Rupam Mahmood, and Martha White. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 17(1):2603–2631, 2016.

[191] Richard Stuart Sutton. Temporal credit assignment in reinforcement learning. 1984.

[192] Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. In *Advances in neural information processing systems*, pp. 1449–1456, 2008.

[193] Umar Syed, Michael Bowling, and Robert E Schapire. Apprenticeship learning using linear programming. In *Proceedings of the 25th international conference on Machine learning*, pp. 1032–1039, 2008.

[194] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4496–4506, 2017.

[195] Philip Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pp. 2139–2148, 2016.

[196] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pp. 3–17. Springer, 1998.

[197] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

[198] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

[199] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

[200] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[201] Matej Večerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[202] Arun Venkatraman, Nicholas Rhinehart, Wen Sun, Lerrel Pinto, Martial Hebert, Byron Boots, Kris Kitani, and J Bagnell. Predictive-state decoders: Encoding the future into recurrent networks. In *Advances in Neural Information Processing Systems*, pp. 1172–1183, 2017.

[203] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[204] Michael Wan, Tanmay Gangwani, and Jian Peng. Mutual information based knowledge transfer under state-action dimension mismatch. *arXiv preprint arXiv:2006.07041*, 2020.

[205] Michael Wan, Jian Peng, and Tanmay Gangwani. Hindsight foresight relabeling for meta-reinforcement learning. *arXiv preprint arXiv:2109.09031*, 2021.

[206] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[207] Ruohan Wang, Carlo Ciliberto, Pierluigi Vito Amadori, and Yiannis Demiris. Random expert distillation: Imitation learning via expert policy support estimation. In *International Conference on Machine Learning*, pp. 6536–6544. PMLR, 2019.

[208] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. 2018.

[209] Ziyu Wang, Josh Merel, Scott Reed, Greg Wayne, Nando de Freitas, and Nicolas Heess. Robust imitation of diverse behaviors. *arXiv preprint arXiv:1707.02747*, 2017.

[210] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4): 279–292, 1992.

[211] Markus Wulfmeier, Ingmar Posner, and Pieter Abbeel. Mutual alignment transfer learning. *arXiv preprint arXiv:1707.07907*, 2017.

[212] Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-gradient reinforcement learning. *arXiv preprint arXiv:1805.09801*, 2018.

[213] Chao Yang, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Huaping Liu, Junzhou Huang, and Chuang Gan. Imitation learning from observations by minimizing inverse dynamics disagreement. *arXiv preprint arXiv:1910.04417*, 2019.

[214] Chao Yu, Jiming Liu, and Shamim Nemati. Reinforcement learning in healthcare: A survey. *arXiv preprint arXiv:1908.08796*, 2019.

[215] Tom Zahavy, Brendan O'Donoghue, Andre Barreto, Volodymyr Mnih, Sebastian Flennerhag, and Satinder Singh. Discovering diverse nearly optimal policies withsuccessor features. *arXiv preprint arXiv:2106.00669*, 2021.

[216] Marvin Zhang, Zoe McCarthy, Chelsea Finn, Sergey Levine, and Pieter Abbeel. Learning deep neural network policies with continuous memory states. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 520–527. IEEE, 2016.

[217] Ruiyi Zhang, Bo Dai, Lihong Li, and Dale Schuurmans. Gendice: Generalized offline estimation of stationary values. *arXiv preprint arXiv:2002.09072*, 2020.

[218] Zeyu Zheng, Junhyuk Oh, Matteo Hessel, Zhongwen Xu, Manuel Kroiss, Hado Van Hasselt, David Silver, and Satinder Singh. What can learned intrinsic rewards capture? In *International Conference on Machine Learning*, pp. 11436–11446. PMLR, 2020.

[219] Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. Off-policy imitation learning from observations. *arXiv preprint arXiv:2102.13185*, 2021.

[220] Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010.

[221] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

[222] Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *arXiv preprint arXiv:1910.08348*, 2019.

[223] Konrad Zolna, Negar Rostamzadeh, Yoshua Bengio, Sungjin Ahn, and Pedro O Pinheiro. Reinforced imitation in heterogeneous action space. *arXiv preprint arXiv:1904.03438*, 2019.