A State-Distribution Matching Approach to Non-Episodic Reinforcement Learning

Archit Sharma [* 1] **Rehaan Ahmad** [* 1] **Chelsea Finn** [1]

Abstract

While reinforcement learning (RL) provides a framework for learning through trial and error, translating RL algorithms into the real world has remained challenging. A major hurdle to real-world application arises from the development of algorithms in an episodic setting where the environment is reset after every trial, in contrast with the continual and non-episodic nature of the real-world encountered by embodied agents such as humans and robots. Enabling agents to learn behaviors autonomously in such non-episodic environments requires that the agent to be able to conduct its own trials. Prior works have considered an alternating approach where a forward policy learns to solve the task and the backward policy learns to reset the environment, but what initial state distribution should the backward policy reset the agent to? Assuming access to a few demonstrations, we propose a new method, MEDAL, that trains the backward policy to match the state distribution in the provided demonstrations. This keeps the agent close to the task-relevant states, allowing for a mix of easy and difficult starting states for the forward policy. Our experiments show that MEDAL matches or outperforms prior methods on three sparse-reward continuous control tasks from the EARL benchmark, with 40% gains on the hardest task, while making fewer assumptions than prior works. Code and videos are at: https://sites.google.com/view/medal-arl/home

1. Introduction

A cornerstone of human and animal intelligence is the ability to learn autonomously through trial and error. To that extent,

---
[*]Equal contribution [1]Stanford University, CA, USA. Correspondence to: Archit Sharma <architsh@stanford.edu>, Rehaan Ahmad <rehaan@stanford.edu>.
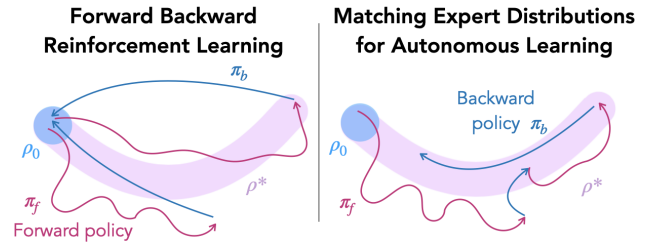
Figure 1. An overview of our proposed method MEDAL (*right*) contrasting it with forward-backward RL (Han et al., 2015; Eysenbach et al., 2017) (*left*). MEDAL trains a backward policy π_b to pull the agent back to the state distribution defined by the demonstrations, enabling the forward policy π_f to the learn the task efficiently in contrast to FBRL that retrieves the agent to the initial state distribution before every trial of π_f.

reinforcement learning (RL) presents a natural framework to develop learning algorithms for embodied agents. Unfortunately, the predominant emphasis on episodic learning represents a departure from the continual non-episodic nature of the real-world, which presents multiple technical challenges. First, episodic training undermines the autonomy of the learning agent by requiring repeated extrinsic interventions to reset the environment after every trial, which can be both time-consuming and expensive as these interventions may have to be conducted by a human. Second, episodic training from narrow initial state distributions can lead to less robust policies that are reliant on environment resets to recover; e.g. Sharma et al. (2022) show that policies learned in episodic settings with narrow initial state distributions are more sensitive to perturbations than those trained in non-episodic settings.

Prior works have found that conventional RL algorithms substantially depreciate in performance when applied in non-episodic settings (Co-Reyes et al., 2020; Zhu et al., 2020a; Sharma et al., 2022). Why do such algorithms struggle to learn in non-episodic, autonomous RL (ARL) settings? Resetting the environment after every single episode allows for natural repetition: the agent can repeatedly practice the task under a narrow set of initial conditions to incrementally improve the policy. Critically, algorithms developed for

episodic learning do not have to learn how to reach these initial conditions in the first place. Thus, the main additional challenge in non-episodic, autonomous RL settings is to enable the repetitive practice that is necessary to learn an adept policy. For example, an autonomous robot that is practicing how to close a door will also need to learn how to open a door.

Several recent works learn a backward policy to enable the main forward policy to practice the task: for example, Han et al. (2015); Eysenbach et al. (2017) propose a backward policy that learns to match the initial state distribution. However, unlike the episodic setting, the agent can practice the task from any initial state, and not just the narrow initial state distribution that is usually provided by resets. Can the backward policy create starting conditions that enable the forward policy to improve efficiently? It could be useful for the agent to try the task both from "easy" states that are close to the goal and harder states that are representative of the starting conditions at evaluation. Easier and harder initial conditions can be seen as a curriculum that simplifies exploration. Kakade & Langford (2002) provide a theoretical discussion on how the initial state distribution affects the performance of the learned policy. One of the results show that the closer the starting state distribution is to the *state distribution of the optimal policy* ρ^*, the faster the policy moves toward the optimal policy π^*. While an oracle access to ρ^* is rarely available, we often have access to a modest set of demonstrations. In this work, we aim to improve autonomous RL by learning a backward policy that matches the starting state distribution to the state distribution observed in the demonstrations. This enables the agent to practice the task from a variety of initial states, including some that are possibly easier to explore from. An intuitive representation of the algorithm is shown in Figure 1.

The primary contribution of our work is an autonomous RL algorithm *Matching Expert Distributions for Autonomous Learning* (MEDAL), which learns a backward policy that matches the state distribution of a small set of demonstrations, in conjunction with a forward policy that optimizes the task reward. We use a classification based approach that implicitly minimizes the distance between the state distribution of the backward policy and the state distribution in the demonstrations without requiring the density under either distribution. In Section 5, we empirically analyze the performance of MEDAL on the Environments for Autonomous RL (EARL) benchmark (Sharma et al., 2022). We find that MEDAL matches or outperforms competitive baselines in all of the sparse-reward environments, with a more than a 40% gain in success rate on the hardest task where all other comparisons fail completely. Our ablations additionally indicate the importance of matching the state distribution in the demonstrations, providing additional empirical support for the hypothesis that the expert state distribution constitutes a good starting state distribution for learning a task.

2. Related Work

Autonomous RL. Using additional policies to enable autonomous learning goes back to the works of (Rivest & Schapire, 1993) in context of finite state automaton, also referred to as "homing strategies" in (Even-Dar et al., 2005) in context of POMDPs. More recently, in context of continuous control, several works propose autonomous RL methods targeting different starting distributions to learn from: Han et al. (2015); Eysenbach et al. (2017) match the initial state distribution, Zhu et al. (2020a) leverage state-novelty (Burda et al., 2018) to create new starting conditions for every trial, and Sharma et al. (2021) create a curriculum of starting states based on the performance of the forward policy to accelerate the learning. In addition, (Xu et al., 2020; Lu et al., 2020) leverage ideas from unsupervised skill discovery (Gregor et al., 2016; Eysenbach et al., 2018; Sharma et al., 2019; Hazan et al., 2019; Campos et al., 2020), with the former using it to create an adversarial initial state distribution and the latter to tackle non-episodic lifelong learning with a non-stationary task-distribution. Our work proposes a novel algorithm MEDAL that, unlike these prior works, opts to match the starting distribution to the state distribution of demonstrations. Value-accelerated Persistent RL (VaPRL) (Sharma et al., 2021) also considers the problem of autonomous RL with a few initial demonstrations. Unlike VaPRL, our algorithm does not rely on relabeling transitions with new goals (Andrychowicz et al., 2017), and thus does not require access to the functional form of the reward function, eliminating the need for additional hyperparameters that require task-specific tuning. A simple and task-agnostic ARL method would accelerate the development of autonomous robotic systems, the benefits of such autonomy being demonstrated by several recent works (Chatzilygeroudis et al., 2018; Gupta et al., 2021; Smith et al., 2021; Ha et al., 2020; Bloesch et al., 2022).

Distribution Matching in RL. Critical to our method is matching the state distribution of the demonstrations. Such a distribution matching perspective is often employed in inverse RL (Ng et al., 2000; Ziebart et al., 2008; 2010; Finn et al., 2016) and imitation learning (Ghasemipour et al., 2020; Argall et al., 2009) or to encourage efficient exploration (Lee et al., 2019). More recently, several works have leveraged implicit distribution matching by posing a classification problem, pioneered in Goodfellow et al. (2014), to imitate demonstrations (Ho & Ermon, 2016; Baram et al., 2017; Kostrikov et al., 2018; Rafailov et al., 2021), to imitate sequences of observations (Torabi et al., 2019; Zhu et al., 2020b), or to learn reward functions for goal-reaching (Fu et al., 2018; Singh et al., 2019). Our work employs a similar discriminator-based approach to encourage the

state distribution induced by the policy to match that of the demonstrations. Importantly, our work focuses on creating an initial state distribution that the forward policy can learn efficiently from, as opposed to these prior works that are designed for the episodic RL setting. As the experiments in Section 5.2 and Section 5.3 show, naïve extensions of these methods to non-episodic settings don't fare well.

Accelerating RL using Demonstrations. There is rich literature on using demonstrations to speed up reinforcement learning, especially for sparse reward problems. Prior works have considering shaping rewards using demonstrations (Brys et al., 2015), pre-training the policy (Rajeswaran et al., 2017), using behavior cloning loss as a regularizer for policy gradients (Rajeswaran et al., 2017) and Q-learning (Nair et al., 2018), and initializing the replay buffer (Nair et al., 2018; Vecerik et al., 2017; Hester et al., 2018). MEDAL leverages demonstrations to accelerate non-episodic reinforcement learning by utilizing demo distribution to create initial conditions for the forward policy. The techniques proposed in these prior works are complimentary to our proposal, and can be leveraged for non-episodic RL in general as well. Indeed, for all methods in our experiments, the replay buffer is initialized with demonstrations.

3. Preliminaries

Autonomous Reinforcement Learning. We use the ARL framework for non-episodic learning defined in Sharma et al. (2022), which we briefly summarize here. Consider a Markov decision process $\mathcal{M} \equiv (\mathcal{S}, \mathcal{A}, p, r, \rho_0)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}_{\geq 0}$ denotes the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ denotes the reward function and ρ_0 denotes the initial state distribution. The learning algorithm $\mathbb{A}$ is defined as $\mathbb{A} : \{s_i, a_i, s_{i+1}, r_i\}_{i=0}^t \mapsto \{a_t, \pi_t\}$, which maps the transitions collected in the environment until time t to an action a_t and its best guess at the optimal policy $\pi_t : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}_{\geq 0}$. First, the initial state is sampled exactly once ($s_0 \sim \rho_0$) at the beginning of the interaction and the learning algorithm interacts with the environment through the actions a_t till $t \to \infty$. This is the key distinction from an episodic RL setting where the environment resets to a state from the initial state distribution after a few steps. Second, the action taken in the environment does not necessarily come from π_t, for example, a backward policy π_b may generate the action taken in the environment.

ARL defines two metrics: *Continuing Policy Evaluation* measures the reward accumulated by $\mathbb{A}$ over the course of training, defined as $\mathbb{C}(\mathbb{A}) = \lim_{h \to \infty} \frac{1}{h} \mathbb{E} \left[\sum_{t=0}^h r(s_t, a_t) \right]$ and *Deployed Policy Evaluation* metric measures how quickly an algorithm improves the output policy π_t at the

task defined by the reward function r, defined as:

$$\mathbb{D}(\mathbb{A}) = \sum_{t=0}^{\infty} J(\pi^*) - J(\pi_t), \tag{1}$$

where $J(\pi) = \mathbb{E} \left[\sum_{j=0}^{\infty} \gamma^j r(s_j, a_j) \right], s_0 \sim \rho_0, a_t \sim \pi(\cdot \mid s_t), s_{t+1} \sim p(\cdot \mid s_t, a_t)$ and $\pi^* \in \arg \max_{\pi} J(\pi)$. The goal for an algorithm $\mathbb{A}$ is to minimize $\mathbb{D}(\mathbb{A})$, that is to bring $J(\pi_t)$ close to $J(\pi^*)$ in the least number of samples possible. Intuitively, minimizing $\mathbb{D}(\mathbb{A})$ corresponds to maximizing the area under the curve for $J(\pi_t)$ versus t.

$\mathbb{C}(\mathbb{A})$ corresponds to the more conventional average-reward reinforcement learning. While algorithms are able to accumulate large rewards during training, they do not necessarily recover the optimal policy in non-episodic settings (Zhu et al., 2020a; Co-Reyes et al., 2020; Sharma et al., 2022). In response, Sharma et al. (2022) introduce $\mathbb{D}(\mathbb{A})$ to explicitly encourage algorithms to learn task-solving behaviors and not just accumulate reward through training.

Imitation Learning via Distribution Matching. Generative Adversarial Networks (Goodfellow, 2016) pioneered implicit distribution matching for distributions where likelihood cannot be computed explicitly. Given a dataset of samples $\{x_i\}_{i=1}^N$, where $x_i \sim p^*(\cdot)$ for some target distribution p^* over the data space $\mathcal{X}$, generative distribution $p_\theta(\cdot)$ can be learned through the following minimax optimization:

$$\min_{p_\theta} \max_D \mathbb{E}_{x \sim p^*} [\log D(x)] + \mathbb{E}_{x \sim p_\theta} [\log(1 - D(x))] \tag{2}$$

where $D : \mathcal{X} \mapsto [0, 1]$ is discriminator solving a binary classification problem. This can be shown to minimize the Jensen-Shannon divergence, that is $\mathcal{D}_{\text{JS}}(p_\theta \mid\mid p^*)$ (Goodfellow et al., 2014; Nowozin et al., 2016) by observing that the Bayes-optimal classifier satisfies $D^*(x) = \frac{p^*(x)}{p^*(x) + p^\theta(x)}$ (assuming that prior probability of true data and fake data is balanced). Because we do not require an explicit density under the generative distribution and only require the ability to sample the distribution, this allows construction of imitation learning methods such as GAIL (Ho & Ermon, 2016) which minimizes $\mathcal{D}_{\text{JS}}(\rho^\pi(s, a) \mid\mid \rho^*(s, a))$, where the policy π is rolled out in the environment starting from initial state distribution ρ_0 to generate samples from the state-action distribution $\rho^\pi(s, a)$ and $\rho^*(s, a)$ is the target state-action distribution of the demonstrations.

4. Matching Expert Distributions for Autonomous Learning (MEDAL)

Several prior works demonstrate the ineffectiveness of standard RL methods in non-episodic settings (Co-Reyes et al., 2020; Zhu et al., 2020a; Sharma et al., 2022). Adding noise

to actions, for example ϵ-greedy in DQN (Mnih et al., 2015) or Gaussian noise in SAC (Haarnoja et al., 2018)), can be sufficient for exploration in episodic setting where every trial starts from a narrow initial state distribution. However, such an approach becomes ineffective in non-episodic settings because the same policy is expected to both solve the task and be sufficiently exploratory. As a result, a common solution in non-episodic autonomous RL settings is to learn another policy in addition to the forward policy π_f that solves the task (Han et al., 2015; Eysenbach et al., 2017; Zhu et al., 2020a): a backward policy π_b that targets a set of states to explore solving the task from. More precisely, the forward policy π_f learns to solve the task from a state sampled from ρ^b, the marginal state distribution of π^b. An appropriate ρ^b can improve the efficiency of learning π_f by creating an effective initial state distribution for it. What should the π_b optimize? We discuss this question in Section 4.1 and a practical way to optimize the suggested objective in Section 4.2. An overview of our proposed algorithm is given in Section 4.3.
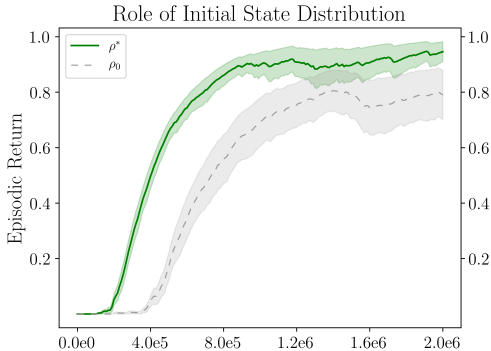

Figure 2. Comparison of sampling initial states s_0 from the state distribution of the optimal policy ρ^*, with sampling the initial state from the default distribution ρ_0 in the episodic setting. The episodic return is computed by initializing the agent at $s_0 \sim \rho_0$ in both the cases. The former improves both the sample efficiency and the performance of the final policy.

4.1. Finding Better Starting States

In episodic settings, π_f always starts exploring from ρ_0, which is the same distribution from which it will be evaluated. A natural objective for π_b then is to minimize the distance between ρ^b and ρ_0. And indeed, prior works have proposed this approach (Han et al., 2015; Eysenbach et al., 2017) by learning a backward controller to retrieve the initial state distribution ρ_0. While the initial state distribution cannot be changed in the episodic setting, π_b does not have any restriction to match ρ_0 in the autonomous RL setting. Is there a better initial state distribution to efficiently learn π_f from?

Interestingly, Kakade & Langford (2002) provide a theoret-

ical discussion on how the initial state distribution affects the performance. The main idea is that learning an optimal policy often requires policy improvement at states that are unlikely to be visited. Creating a more uniform starting state distribution can accelerate policy improvement by encouraging policy improvement at those unlikely states. The formal statement can be found in (Kakade & Langford, 2002, Corollary 4.5). Informally, the result states that the upper bound on the difference between the optimal performance and that of policy π is proportional to $\|\frac{\rho^*(s)}{\mu}\|_\infty$, where ρ^* is the state distribution of the optimal policy and μ is the initial state distribution. This suggests that an initial state distribution μ that is close to the optimal state distribution ρ^* would enable efficient learning. Intuitively, some initial states in the optimal state distribution would simplify the exploration by being closer to high reward states, which can be bootstrapped upon to learn faster from the harder initial states. To empirically verify the theoretical results, we compare the learning speed of RL algorithm in the episodic setting on *tabletop organization* (environment details in Section 5) when starting from (a) the standard initial state distribution, that is $s_0 \sim \rho_0$, versus (b) states sampled from the stationary distribution of the optimal policy, that is $s_0 \sim \rho^*(s)$. We find in Figure 2 that the latter not only improves the learning speed, but also improves the performance by nearly 18%.

4.2. Resetting to Match the Expert State Distribution

The theoretical and empirical results in the previous section suggest that π_f should attempt to solve the task from an initial state distribution that is close to $\rho^*(s)$, thus implying that π_b should try to match $\rho^*(s)$. How do we match ρ^b to ρ^*? We will assume access to a small set of samples from $\rho^*(s)$ in the form of demonstrations $\mathcal{D}_f$. Because we are limited to sampling ρ^b and only have a fixed set of samples from ρ^*, we consider the following optimization problem:

$$\min_{\pi_b} \max_C \mathbb{E}_{s \sim \rho^*} \big[\log C(s) \big] + \mathbb{E}_{s \sim \rho^b} \big[\log(1 - C(s)) \big] \quad (3)$$

where $C : \mathcal{S} \mapsto [0, 1]$ is a state-space classifier. This optimization is very much reminiscent of implicit distribution matching techniques used in (Goodfellow et al., 2014; Nowozin et al., 2016; Ho & Ermon, 2016; Ghasemipour et al., 2020) when only the samples are available and densities cannot be explicitly measured. This can be interpreted as minimizing the Jensen-Shannon divergence $\mathcal{D}_{JS}(\rho^b \| \rho^*)$. Following these prior works, $C(s)$ solves a binary classification where $s \sim \rho^*$ has a label 1 and $s \sim \rho^b$ has a label 0. Further, π_b solves a RL problem to maximize $\mathbb{E}_{s \sim \rho^b}[r(s, a)]$, where the reward function $r(s, a) = -\log(1 - C(s))$. Assuming sufficiently expressive non-parametric function classes, $(\rho^*, 0.5)$ is a saddle point for Equation 3.

Relationship to Prior Imitation Learning Methods. GAIL (Ho & Ermon, 2016) proposes to match the state-

action distribution $\rho^\pi(s, a)$ to that of the expert $\rho^*(s, a)$, that is minimize $\mathcal{D}_{\text{JS}}(\rho^\pi(s, a) \parallel \rho^*(s, a))$. Prior works have considered the problem of imitation learning when state-only observations are available (Torabi et al., 2019; Zhu et al., 2020b) by minimizing $\mathcal{D}_f(\rho^\pi(s, s') \parallel \rho^*(s, s'))$, where f-divergence enables generalized treatment of different discrepancy measures such KL-divergence of JS-divergence used in prior work (Nowozin et al., 2016). In contrast to these works, our work proposes to minimize $\mathcal{D}_{\text{JS}}(\rho^\pi(s) \parallel \rho^*(s))$. Furthermore, state distribution matching is only used for the backward policy in our algorithm, whereas the forward policy is maximizing return, as we summarize in the next section. Finally, unlike prior works, the motivation for matching the state distributions is to create an effective initial state distribution for the forward policy π_f. Our experimental results in Section 5.2 suggest that naively extending GAIL to non-episodic settings is not effective, validating the importance of these key differences.

Algorithm 1 Matching Expert Distributions for Autonomous Learning (MEDAL)

> **require:** forward demos $\mathcal{D}_f$;
> **optional:** backward demos $\mathcal{D}_b$;
> **initialize:** $\mathcal{R}_f, \pi_f(a \mid s), Q^{\pi_f}(s, a)$; // forward policy
> **initialize:** $\mathcal{R}_b, \pi_b(a \mid s), Q^{\pi_b}(s, a)$; // backward policy
> **initialize:** $C(s)$; // state-space discriminator
> $\mathcal{R}_f \leftarrow \mathcal{R}_f \cup \mathcal{D}_f, \mathcal{R}_b \leftarrow \mathcal{R}_b \cup \mathcal{D}_b$;
> $s \sim \rho_0$; // sample initial state
> **while** not done **do**
>   // run forward policy for a fixed number of steps or until goal is reached, otherwise run backward policy
>   **if** forward **then**
> $a \sim \pi_f(\cdot \mid s)$;
> $s' \sim p(\cdot \mid s, a), r \leftarrow r(s, a)$;
> $\mathcal{R}_f \leftarrow \mathcal{R}_f \cup \{(s, a, s', r)\}$;
>   update π_f, Q^{π_f};
>   **else**
> $a \sim \pi_b(\cdot \mid s)$;
> $s' \sim p(\cdot \mid s, a), r \leftarrow -\log(1 - C(s'))$;
> $\mathcal{R}_b \leftarrow \mathcal{R}_b \cup \{(s, a, s', r)\}$;
>   update π_b, Q^{π_b};
>   **end if**
>   // train disriminator every K steps
>   **if** train-discriminator **then**
>   // sample a batch of positives S_p from the forward demos $\mathcal{D}_f$, and a batch of negatives S_n from backward replay buffer $\mathcal{R}_b$
> $S_p \sim \mathcal{D}_f, S_n \sim \mathcal{R}_b$;
>   update C on $S_p \cup S_n$;
>   **end if**
> $s \leftarrow s'$;
> **end while**

4.3. MEDAL Overview

With these components in place, we now summarize our proposed algorithm, *Matching Expert Distributions for Autonomous Learning* (MEDAL). We simultaneously learn the following components: a *forward policy* that learns to solve the task and will also be used for evaluation, a *backward policy* that learns creates the initial state distribution for the forward policy by matching the state distribution in the demonstrations, and finally a *state-space discriminator* that learns to distinguish between the states visited by the backward policy and the states visited in the demonstrations. MEDAL assumes access to a set of forward demonstrations $\mathcal{D}_f$, completing the task from the initial state distribution, and optionally, a set of backward demonstrations $\mathcal{D}_b$ undoing the task back to the initial state distribution. The forward policy π_f is trained to maximize $\mathbb{E}[\sum_{t=0}^\infty \gamma^t r(s_t, a_t)]$ and the replay buffer for the forward policy is initialized using $\mathcal{D}_f$. The backward policy π_b trains to minimize $\mathcal{D}_{\text{JS}}(\rho^b(s) \parallel \rho^*(s))$ which translates into maximizing $-\mathbb{E}[\sum_{t=0}^\infty \gamma^t \log(1 - C(s_{t+1}))]$ and the replay buffer for the backward policy is initialized using the backward demonstrations $\mathcal{D}_b$, if available. Finally, the state-space discriminator $C(s)$ trains to classify states sampled from the *forward* demonstrations $\mathcal{D}_f$ with label 1 and states visited by π_b as label 0. Note, we are trying to match the state marginal of policy π_b (i.e. $\rho^b(s)$) to the optimal state distribution $\rho^*(s)$ (approximated via *forward demonstrations* $\mathcal{D}_f$, not *backward demonstrations*), thereby motivating the classification problem for $C(s)$.

When interacting with the environment during training, we alternate between collecting samples using π_f for a fixed number of steps and collecting samples using π_b for a fixed number of steps. The policies can be updated using any RL algorithm. The state-space discriminator $C(s)$ is updated every K steps collected in the environment, with the states visited by π_b being labeled as 0 and states in $\mathcal{D}_f$ labeled as 1. The minibatch for updating the parameters of $C(s)$ is balanced to ensure equal samples from $\rho^*(s)$ and $\rho^b(s)$. The pseudocode for MEDAL is provided in Algorithm 1, and further implementation details can be found in Appendix A.

5. Experiments

In this section, we empirically analyze the performance of MEDAL to answer to following questions: (1) How does MEDAL compare to other non-episodic, autonomous RL methods? (2) Given the demonstrations, can existing imitation learning methods suffice? (3) How important is it for the backward controller to match the entire state distribution, instead of just the initial state distribution?

Environments. To analyze these questions, we consider three sparse-reward continuous-control environments from
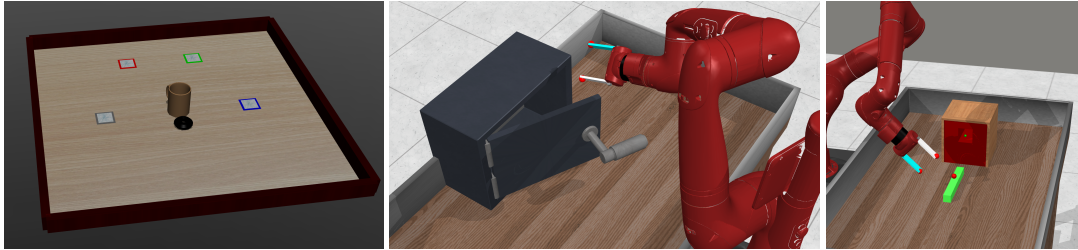
Figure 3. Continuous-control environments from the EARL benchmark: (*left*) Table-top organization (**TO**) where a gripper is tasked with moving a mug to one of the four goal locations, (*center*) sawyer door closing (**SD**) where the sawyer robot is tasked with closing the door, (*right*) sawyer peg insertion (**SP**) where the robot is tasked with picking up the peg and inserting into the goal location.

the EARL benchmark (Sharma et al., 2022). The *table-top organization* is a simplified manipulation environment where a gripper is tasked to move the mug to one of four coasters. The *sawyer door closing* environment requires a sawyer robot arm to learn how to close a door starting from various initial positions. The challenge in the ARL setting arises from the fact that the agent has to open the door to practice closing it again. Finally, the *sawyer peg insertion* environment requires the sawyer robot arm to pick up a peg and insert it into a designated goal location. This is a particularly challenging environment as the autonomously operating robot can push the peg into places where it can be hard to retrieve it back, a problem that is not encountered in the episodic setting as the environment is reset to the initial state distribution every few hundred steps.

Evaluation. We follow the evaluation protocol laid down in the EARL benchmark. All algorithms are reset to a state $s_0 \sim \rho_0$ and interact with their environments almost fully autonomously thereon, only being reset to an initial state intermittently after several hundreds of thousands of steps of interaction. Since our objective is to acquire task policies in a sample efficient way, we will focus on *deployed policy evaluation*. Specifically, we approximate $J(\pi_t) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$ by averaging the return of the policy over 10 episodes starting from $s_0 \sim \rho_0$, every $10,000$ steps collected in the training environment. Note, the trajectories collected for evaluation are *not* provided to the learning algorithm $\mathbb{A}$. For all considered environments, the reward functions are sparse in nature and correspondingly, EARL provides a small set of demonstrations to the algorithms, that correspond to doing and undoing the task (a total of 10-30 demonstrations depending on the environment). Environment specific details such as reward functions and intermittent resets can be found in Appendix B.

5.1. Benchmarking MEDAL on EARL

First, we benchmark our proposed method MEDAL on the aforementioned EARL environments against state-of-the-art non-episodic ARL methods.

Method	Tabletop Organization	Sawyer Door	Sawyer Peg
naïve RL	0.32 (0.17)	0.00 (0.00)	0.00 (0.00)
FBRL	0.94 (0.04)	**1.00 (0.00)**	0.00 (0.00)
R3L	0.96 (0.04)	0.54 (0.18)	0.00 (0.00)
VaPRL	**0.98 (0.02)**	0.94 (0.05)	0.00 (0.00)
MEDAL	**0.98 (0.02)**	**1.00 (0.00)**	**0.40 (0.16)**
oracle RL	0.80 (0.11)	1.00 (0.00)	1.00 (0.00)

Table 1. Average return of the final learned policy. Performance is averaged over 5 random seeds. The mean and and the standard error are reported, with the best performing entry in bold. For all domains, 1.0 indicates the maximum performance and 0.0 indicates minimum performance.

Comparisons. We briefly review the methods benchmarked on EARL, which MEDAL will be compared against: (1) forward-backward RL (**FBRL**) (Han et al., 2015; Eysenbach et al., 2017), where the backward policy recovers the initial state distribution; (2) value-accelerated persistent RL (**VaPRL**) (Sharma et al., 2021), where the backward policy creates a curriculum based on the forward policy's performance; (3) **R3L** (Zhu et al., 2020a) has a backward policy that optimizes a state-novelty reward (Burda et al., 2018) to encourage the forward policy to solve the tasks from new states in every trial; (4) **naïve RL** represents the episodic RL approach where only a forward policy optimizes the task-reward throughout training; and finally (5) **oracle RL** is the same episodic RL baseline but operating in the episodic setting. For a fair comparison, the forward policy for all baselines use SAC (Haarnoja et al., 2018), and the replay buffer is always initialized with the forward demonstrations. Further, the replay buffers for backward policies in **FBRL, VaPRL** is also initialized with the backward demos. The replay buffer of the backward policy in R3L is not initialized with backward demos as it will reduce the novelty of the states in the backward demos for the RND reward without the backward policy ever visiting those states.
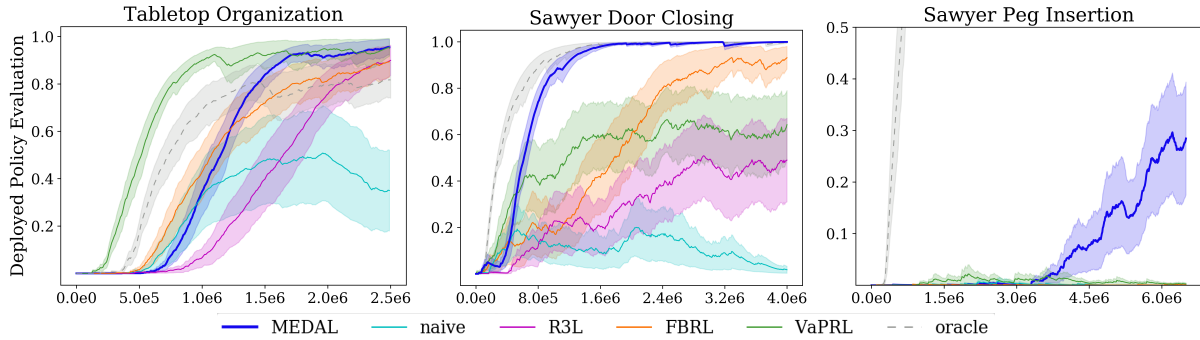
It's important to note that some of these comparisons make

Figure 4. Performance of each method on (*left*) the table-top organization environment, (*center*) the sawyer door closing environment, and (*right*) the sawyer peg environment. Plots show learning curves with mean and standard error over 5 random seeds.

additional assumptions compared to MEDAL:

- **oracle RL** operates in the episodic setting, that is the environment is reset to a state from the initial state distribution every few hundred steps. The baseline is included as a reference to compare performance of baselines in ARL versus the conventional episodic setting. It also enables us to compare the performance of conventional RL algorithms when moving from the episodic setting to the ARL setting, by comparing the performance of **oracle RL** and **naïve RL**.

- **VaPRL** relies on relabeling goals which requires the ability to query the reward function for any arbitrary state and goal (as the VaPRL curriculum can task the agent to reach arbitrary goals from the demonstrations). Additionally, VaPRL has a task specific hyperparameter that controls how quickly the curriculum moves towards the initial state distribution.

In a real-world settings, where the reward function often needs to be learned as well (for example from images), these assumptions can be detrimental to their practical application. While FBRL also requires an additional reward function to reach the initial state distribution, the requirement is not as steep. Additionally, we consider a version of FBRL that learns this reward function in Section 5.3. However, the ability to query the reward function for arbitrary states and goals, as is required by VaPRL, can be infeasible in practice. The impact of these additional assumptions cannot be overstated, as the primary motivation for the autonomous RL framework is to be representative of real-world RL training.

Results. Table 5.1 shows the performance of the final forward policy, and Figure 4 shows the *deployed* performance of the forward policy versus the training time for different methods. MEDAL consistently outputs the best performing final policy, as can be seen in Table 5.1. Particularly notable is the performance on *sawyer peg insertion*, where the final policy learned by MEDAL gets 40% success rate on average, while all other methods fail completely. With the exception

of VaPRL on *tabletop organization*, MEDAL also learns more efficiently compared to any of the prior methods. Notably, MEDAL substantially reduces the sample efficiency gap between ARL methods and episodic methods on *sawyer door closing*.

We posit two reasons for the success of MEDAL: (a) Learning a backward policy that retrieves the agent close to the task distribution enables efficient exploration, producing the speedup in performance. (b) Bringing the agent closer to the state distribution implicit in the demonstrations may be easier to maximize compared to other objectives, for example, retrieving the agent to the initial state distribution.

5.2. Imitation Learning

Given that MEDAL assumes access to a set of demonstrations, a natural alternative to consider is imitation learning. In this section, we focus our experiments on the tabletop organization environment. We first test how a behavior cloning fares (**BC**). Results in Figure 5 suggest that behavior cloning does not do well on *tabletop organization*, completely failing to solve the task and leaves substantial room for improvement on *sawyer door*. This is to be expected as EARL provides only a small number of demonstrations, and errors compounding over time from imperfect policies generally leads to poor performance. How do imitation learning methods with online data collection fare? We consider an off-policy version of GAIL, Discriminator Actor-Critic (DAC) (Kostrikov et al., 2018), which matches $\rho^\pi(s, a)$ to $\rho^*(s, a)$ with an implicit distribution matching approach similar to ours. Assuming that $\rho^\pi(s, a)$ can match $\rho^*(s, a)$, the method should in principle recover the optimal policy – there is nothing specific about GAIL that restricts it to the episodic setting. However, as the results in Figure 5 suggest, there is a substantial drop in performance when running GAIL in episodic setting (**oracle GAIL**) versus the non-episodic ARL setting (**naïve GAIL**). While such a distribution matching could succeed, naïvely extending the
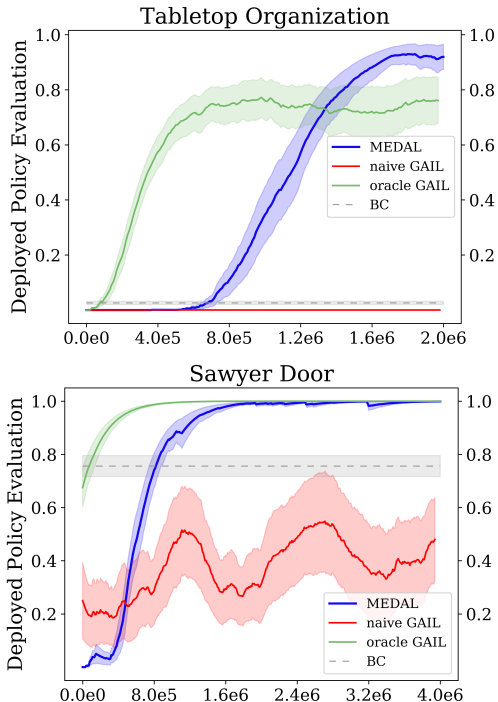
Figure 5. **MEDAL** in comparison to imitation learning methods on *tabletop organization* and *sawyer door closing*. Behavior cloning (**BC**) does not fare well, suggesting the importance of online data collection. The success of online imitation learning methods such as GAIL in episodic settings does not translate to the non-episodic ARL setting, as indicated by the substantial drop in performance of **naïve GAIL** compared to **oracle GAIL**.

methods to the ARL setting is not as successful, suggesting that it may require an additional policy (similar to the backward policy) to be more effective.

5.3. The Choice of State Distribution

The key element in MEDAL is matching the state distribution of the backward policy to the states in the demonstrations. To isolate the role of our proposed scheme of minimizing $\mathcal{D}_{\mathrm{JS}}(\rho^b \parallel \rho^*)$, we compare it to an alternate method that minimizes $\mathcal{D}_{\mathrm{JS}}(\rho^b \parallel \rho_0)$, i.e., matching the initial state distribution ρ_0 instead of ρ^*. This makes exactly one change to MEDAL: instead of sampling positives for the discriminator $C(s)$ from forward demonstrations $\mathcal{D}_f$, the positives are sampled from ρ_0. Interestingly, this also provides a practically realizable implementation of **FBRL**, as it removes the requirement of the additional reward function required for the learning a backward policy to reach the initial state distribution. We call this method **FBRL + VICE** as VICE ([Singh et al., 2019](#)) enables learning a goal reaching reward function using a few samples of the goal distribution, in this case the goal distribution for π_b being ρ_0. As can be seen in Figure 6, the FBRL + VICE learns slower

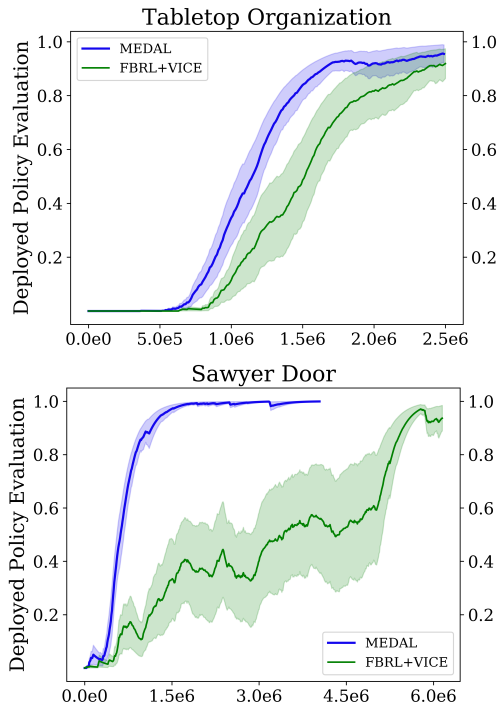than MEDAL, highlighting the importance of matching the entire state distribution as done in MEDAL.

Figure 6. Isolating the effect of matching demonstration data. The speed up of **MEDAL** compared to **FBRL + VICE**, which matches the initial state distribution, suggests that the performance gains of MEDAL can be attributed to the better initial state distribution created by the backward controller.

6. Conclusion

We propose MEDAL, an autonomous RL algorithm that learns a backward policy to match the expert state distribution using an implicit distribution matching approach. Our empirical analysis indicates that this approach creates an effective initial state distribution for the forward policy, improving both the performance and the efficiency. The simplicity of MEDAL also makes it more amenable for the real-world, not requiring access to additional reward functions.

MEDAL assumes access to a (small) set of demonstrations, which may not be feasible in several real-world scenarios. Identifying good initial state distributions without relying on a set of demonstrations would increase the applicability of MEDAL. Similarly, in applications where safe exploration is a requirement, MEDAL can be adapted to constrain the forward policy such that it stays close to the task-distribution defined by the demonstrations. While MEDAL pushes further the improvements in ARL, as exemplified by the reduction of sample efficiency gap on *sawyer door closing* results, there is still a substantial gap in performance between ARL methods and oracle RL on *sawyer peg*.

7. Acknowledgements

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Baram, N., Anschel, O., Caspi, I., and Mannor, S. End-to-end differentiable adversarial imitation learning. In *International Conference on Machine Learning*, pp. 390–399. PMLR, 2017.

Bloesch, M., Humplik, J., Patraucean, V., Hafner, R., Haarnoja, T., Byravan, A., Siegel, N. Y., Tunyasuvunakool, S., Casarini, F., Batchelor, N., et al. Towards real robot learning in the wild: A case study in bipedal locomotion. In *Conference on Robot Learning*, pp. 1502–1511. PMLR, 2022.

Brys, T., Harutyunyan, A., Suay, H. B., Chernova, S., Taylor, M. E., and Nowé, A. Reinforcement learning from demonstration through shaping. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Campos, V., Trott, A., Xiong, C., Socher, R., Giró-i Nieto, X., and Torres, J. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, pp. 1317–1327. PMLR, 2020.

Chatzilygeroudis, K., Vassiliades, V., and Mouret, J.-B. Reset-free trial-and-error learning for robot damage recovery. *Robotics and Autonomous Systems*, 100:236–250, 2018.

Co-Reyes, J. D., Sanjeev, S., Berseth, G., Gupta, A., and Levine, S. Ecological reinforcement learning. *arXiv preprint arXiv:2006.12478*, 2020.

Even-Dar, E., Kakade, S. M., and Mansour, Y. Reinforcement learning in pomdps without resets. 2005.

Eysenbach, B., Gu, S., Ibarz, J., and Levine, S. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Finn, C., Levine, S., and Abbeel, P. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pp. 49–58. PMLR, 2016.

Fu, J., Singh, A., Ghosh, D., Yang, L., and Levine, S. Variational inverse control with events: A general framework for data-driven reward definition. *arXiv preprint arXiv:1805.11686*, 2018.

Ghasemipour, S. K. S., Zemel, R., and Gu, S. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*, pp. 1259–1277. PMLR, 2020.

Goodfellow, I. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Gupta, A., Yu, J., Zhao, T. Z., Kumar, V., Rovinsky, A., Xu, K., Devlin, T., and Levine, S. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. *arXiv preprint arXiv:2104.11203*, 2021.

Ha, S., Xu, P., Tan, Z., Levine, S., and Tan, J. Learning to walk in the real world with minimal human effort. *arXiv preprint arXiv:2002.08550*, 2020.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Han, W., Levine, S., and Abbeel, P. Learning compound multi-step controllers under unknown dynamics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6435–6442. IEEE, 2015.

Hazan, E., Kakade, S., Singh, K., and Van Soest, A. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pp. 2681–2691. PMLR, 2019.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Ho, J. and Ermon, S. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29:4565–4573, 2016.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.

Kostrikov, I., Agrawal, K. K., Dwibedi, D., Levine, S., and Tompson, J. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*, 2018.

Lee, L., Eysenbach, B., Parisotto, E., Xing, E., Levine, S., and Salakhutdinov, R. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

Lu, K., Grover, A., Abbeel, P., and Mordatch, I. Reset-free lifelong learning with skill-space planning. *arXiv preprint arXiv:2012.03548*, 2020.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 6292–6299. IEEE, 2018.

Ng, A. Y., Russell, S. J., et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Nowozin, S., Cseke, B., and Tomioka, R. f-gan: Training generative neural samplers using variational divergence minimization. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 271–279, 2016.

Rafailov, R., Yu, T., Rajeswaran, A., and Finn, C. Visual adversarial imitation learning using variational models. *Advances in Neural Information Processing Systems*, 34, 2021.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning

and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Rivest, R. L. and Schapire, R. E. Inference of finite automata using homing sequences. *Information and Computation*, 103(2):299–347, 1993.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

Sharma, A., Gupta, A., Levine, S., Hausman, K., and Finn, C. Autonomous reinforcement learning via subgoal curricula. *Advances in Neural Information Processing Systems*, 34, 2021.

Sharma, A., Xu, K., Sardana, N., Gupta, A., Hausman, K., Levine, S., and Finn, C. Autonomous reinforcement learning: Formalism and benchmarking. *International Conference on Learning Representations*, 2022.

Singh, A., Yang, L., Hartikainen, K., Finn, C., and Levine, S. End-to-end robotic reinforcement learning without reward engineering. *arXiv preprint arXiv:1904.07854*, 2019.

Smith, L., Kew, J. C., Peng, X. B., Ha, S., Tan, J., and Levine, S. Legged robots that keep on learning: Fine-tuning locomotion policies in the real world. *arXiv preprint arXiv:2110.05457*, 2021.

Torabi, F., Warnell, G., and Stone, P. Adversarial imitation learning from state-only demonstrations. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2229–2231, 2019.

Vecerik, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., and Riedmiller, M. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

Xu, K., Verma, S., Finn, C., and Levine, S. Continual learning of control primitives: Skill discovery via reset-games. *arXiv preprint arXiv:2011.05286*, 2020.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Zhu, H., Yu, J., Gupta, A., Shah, D., Hartikainen, K., Singh, A., Kumar, V., and Levine, S. The ingredients of real world robotic reinforcement learning. In *International Conference on Learning Representations*, 2020a.

Zhu, Z., Lin, K., Dai, B., and Zhou, J. Off-policy imitation learning from observations. In *the Thirty-fourth Annual Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020b.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

Ziebart, B. D., Bagnell, J. A., and Dey, A. K. Modeling interaction via the principle of maximum causal entropy. In *ICML*, 2010.
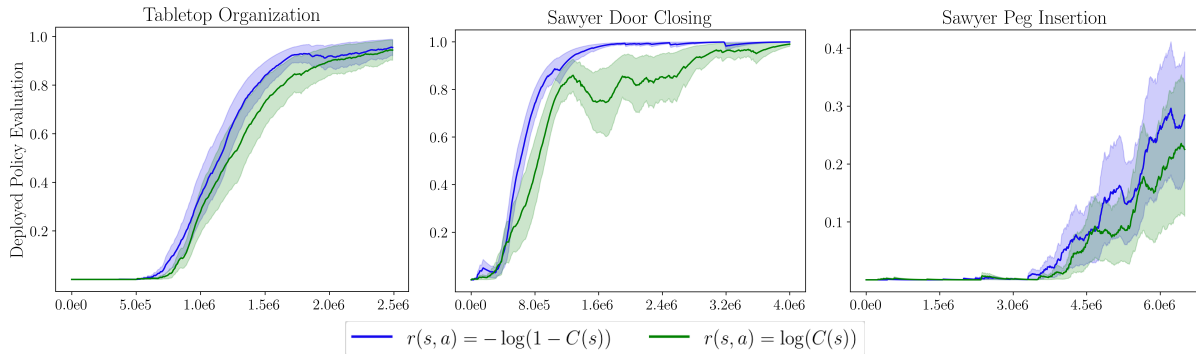
Figure 7. Performance comparison of $r(s, a) = \log(C(s))$ and $r(s, a) = -\log(1 - C(s))$ on (*left*) the *table-top organization* environment, (*center*) the *sawyer door closing* environment, and (*right*) the *sawyer peg* environment. Plots show learning curves with mean and standard error over 5 random seeds.

A. MEDAL Implementation

MEDAL is implemented with TF-Agents, built on SAC as the base RL algorithm. Hyperparameters follow the default values: `initial collect steps`: 10,000, `batch size` sampled from replay buffer for updating policy and critic: 256, steps `collected per iteration`: 1, `trained per iteration`: 1, `discount factor`: 0.99, `learning rate`: $3e - 4$ (for critics, actors, and discriminator). The actor and critic network were parameterized as neural networks with two hidden layers each of size 256. For the discriminator, it was parameterized as a neural network with one hidden layer of size 128. This discriminator is updated once every 10 collection steps for all environments. Due to a small positive dataset, mixup (Zhang et al., 2017) is used as a regularization technique on the discriminator for all environments. Additionally, the batch size for the discriminator is set to 800 for all environments as this significantly larger value was found to stabilize training.

Another choice that improved the stability was the choice of reward function for the backward controller: both $r(s, a) = -\log(1 - C(s))$ and $r(s, a) = \log(C(s))$ preserve the saddle point $(\rho^*, 0.5)$ for the optimization in Equation 3. However, as can be seen in Figure 7, $r(s, a) = -\log(1 - C(s))$ leads to both better and stable performance. We hypothesize that this is due to smaller gradients of the $-\log(1 - C(s))$ when $C(s) \leq 0.5$, which is where the discriminator is expected to be for most of the training as the discriminator can easily distinguish between expert states and those of the backward policy to begin with.

B. Environments

The environment details can be found in (Sharma et al., 2022). We briefly describe environments for completeness. For every environment, H_T defines the number of steps after which the environment is reset to a state $s_0 \sim \rho_0$, and H_E defines the evaluation horizon over which the return is computed for deployed policy evaluation:

`table-top organization`: Table-top organization is run with a training horizon of $H_T = 200,000$ and $H_E = 200$. The sparse reward function is given by:

$$r(s, g) = \mathbb{I}(\|s - g\|_2 \leq 0.2),$$

where $\mathbb{I}$ denotes the indicator function. The environment has 4 possible goal locations for the mug, and goal location for the gripper is in the center. EARL provides a total of 12 forward demonstrations and 12 backward demonstrations (3 per goal).

`sawyer door closing`: Sawyer door closing is run with a training horizon of $H_T = 200,000$ and an episode horizon of $H_E = 300$. The sparse reward function is:

$$r(s, g) = \mathbb{I}(\|s - g\|_2 \leq 0.02),$$

where $\mathbb{I}$ again denotes the indicator function. The goal for the door and the robot arm is the closed door position. EARL provides 5 forward demonstrations and 5 backward demonstrations.

`sawyer peg`: Sawyer peg is run with a training horizon of $H_T = 100,000$ and an episode horizon of $H_E = 200$. The

sparse reward function is:

$$r(s, g) = \mathbb{I}(\|s - g\|_2 \leq 0.05),$$

where $\mathbb{I}$ again denotes the indicator function. The goal for the peg is to be placed in the goal slot. EARL provides 10 forward demonstrations and 20 backward demonstrations.