You Only Live Once: Single-Life Reinforcement Learning via Learned Reward Shaping

Anonymous Author(s)
Affiliation
Address
email

Abstract

Reinforcement learning algorithms are typically designed to learn a performant policy that can repeatedly and autonomously complete a task, typically starting from scratch. However, many real-world situations operate under a different set of assumptions: the goal might not be to learn a policy that can do the task repeatedly, but simply to perform a new task successfully once, ideally as quickly as possible, and while leveraging some prior knowledge or experience. For example, imagine a robot that is exploring another planet, where it cannot get help or supervision from humans. If it needs to navigate to a crater that it has never seen before in search of water, it does not really need to acquire a policy for reaching craters reliably, it only needs to reach this particular crater once. It must do so without the benefit of episodic resets and tackle a new, unknown terrain, but it can leverage prior experience it acquired on Earth. We formalize this problem setting, which we call *single-life reinforcement learning* (SLRL), where an agent must complete a task once while contending with some form of novelty in a single trial without interventions, given some prior data. In this setting, we find that algorithms designed for standard episodic reinforcement learning can struggle, as they have trouble recovering from novel states especially when informative rewards are not provided. Motivated by this observation, we also propose an algorithm, Q-weighted adversarial learning (QWALE), that addresses the dearth of supervision by employing a distribution matching strategy that leverages the agent's prior experience as guidance in novel situations. Our experiments on several single-life continuous control problems indicate that methods based on our distribution matching formulation are 20-60% more successful because they can more quickly recover from novel, out-of-distribution states.

1 Introduction

When building autonomous agents for the natural world, often the goal is not to learn a performant policy but rather to get something done, perhaps even suboptimally. For example, an agent exploring on Mars looking for water will only need to complete its mission a single time. As another example, a rescue robot will need to recover valuables from a particular burning building only once. While the agent may have access to prior data about its task, a challenge arises from the fact the agent is inevitably going to have to contend with some form of novelty. In the prior examples, the agent on Mars may have to contend with unknown terrain and environmental conditions, and the rescue agent may find certain paths in the building unpassable due to the fire. We formalize this setting as the *single-life reinforcement learning* (SLRL) setting, where the agent is evaluated on its ability to complete a task in a single trial autonomously without episodic resets. Importantly, the given online task contains an aspect of novelty not present in the prior data although the task objective remains the same. The agent's objective is to complete the given task as quickly as possible, rather than learn a policy that can repeatedly complete the task.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

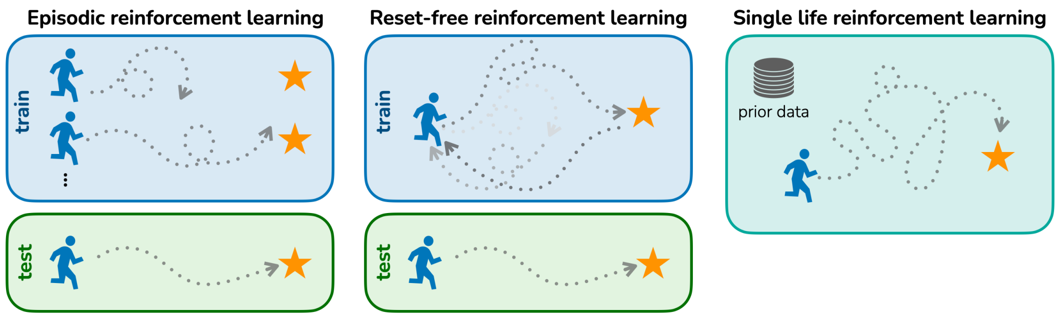

Figure 1: We study the single-life reinforcement learning (SLRL) problem, where given prior data, an agent must complete a task autonomously in a single trial in a domain with a novel distribution shift.

We find that algorithms designed for episodic policy learning can struggle to complete single-life tasks, even when initialized with prior data. These algorithms empirically struggle to recover from novel states. In episodic RL, the agent can rely on a reset to recover from an unfamiliar state. In contrast, in SLRL, the agent will inevitably fall off the distribution of prior data and must find its way back to a good state distribution on its own. We find that fine-tuning a pre-trained value function via online RL will not explicitly encourage the agent to get back on distribution. We hypothesize that biasing exploration towards the known distribution represented in prior data and incentivizing the agent to stay there may be suboptimal from a policy learning perspective but may enable the agent to get the task done more quickly, which is what we care about in SLRL. While shaped rewards may help the agent find its way back and ultimately complete the desired task, the agent may often find itself in a sparse reward environment or with access to rewards that are not informative enough to be guided towards task completion.

Adversarial imitation learning (AIL) approaches such as GAIL ([21]) can potentially provide the desired reward shaping via distribution matching. However, using existing AIL methods naively may not give the intended behavior in the SLRL setting due to two main shortcomings. First, such methods assume expert demonstrations are given as prior data, but in SLRL, we may be given suboptimal offline prior data. Second, AIL methods train the agent to match the entire distribution of prior data, which may be key to learning an optimal policy, but may be a drawback in our setting, as the agent might not be consistently guided towards task completion. To address these shortcomings, we propose a method in which different states in the prior data are weighted different amounts by their estimated Q-value. More concretely, we propose a Q-weighted AIL approach that incentivizes the agent to move towards states in the prior data with higher value than its current state, so that agent may be guided consistently towards states closer to task completion.

Our contributions are as follows. First, we formalize the SLRL problem setting, which we believe to be a useful framework for modeling many situations in the real world. We next provide an intuitive argument and empirical analysis suggesting that learned reward shaping via distribution matching is better suited for this setting than finetuning without additional reward shaping. We identify challenges that uniquely arise in the SLRL setting with existing distribution matching approaches and propose a new approach, Q-weighted adversarial learning (QWALE), which is less sensitive to the quality of prior data available and provides the agent with a shaped reward towards completing the desired task a single time. Through our experiments, we explore the performance of different approaches in SLRL. We find that QWALE can meaningfully guide the agent to explore the state distribution in its prior data to complete the desired novel task 20-60% more successfully on four separate domains compared to existing distribution matching approaches and RL fine-tuning.

2 Related Work

Autonomous RL. In the context of deep RL, agents typically (but not always) are trained in episodic setting and are evaluated on the quality of the learned policy. Several recent works have developed algorithms that can learn without episodic resets [18, 6, 63, 42, 44, 15, 16, 24, 43]. Like our work, such methods aim to make it possible to learn without any episodic resets, but are typically still focused on acquiring an effective policy that can perform the task repeatedly, typically by training some auxiliary controller to enable the policy to "retry" the task multiple times without resets. In contrast, our aim is to develop an algorithm that can solve the task once, but as quickly as possible, which introduces a unique set of challenges as we discussed above.

Continual RL. There is a rich literature on reinforcement learning in the continuing setting [29, 41, 47, 60, 37, 28, 55] that considers maximizing the average reward accumulated over an infinite horizon without episodic resets. Such works often also consider regret minimization as the objective. SLRL is closely related, and can be viewed as a special case where the agent has access to a prior offline dataset and aims to solve a single task as quickly as possible in a new domain. While the focus in continual learning is on general "lifelong" methods or on exploration, our focus is on effectively leveraging prior data in a setting that is meant to be reflective of real-world tasks (for example, in robotics).

Leveraging offline data in online RL. Learning expert policies given prior interaction data has been extensively studied in imitation learning [1, 38, 13], inverse RL [33, 8, 65, 66], RL for sparse reward settings [3, 32, 36, 20, 50] and offline RL [27, 26, 23, 56, 32]. Across all these diverse topics, the goal is to learn a competent policy that can solve the task efficiently whereas the objective in this work is to complete the task in a single trial as quickly as possible. To this end, we build on recent adversarial approaches to inverse RL [21, 11, 45, 49, 25, 64] to encourage agent's state visitation towards expert prior data where the agent is likely to be successful. Prior methods have also studied adversarial inverse RL and imitation learning with non-expert data [51, 46, 57, 52, 54, 53, 5, 2]. However, as we discuss in Section 6, these approaches need to be adapted for the SLRL setting to be efficient at completing the task and to handle novelty, for example when the dynamics may have changed.

Transfer and adaptation in RL. Many prior works have studied the problem of adapting in presence of shifts between train and test settings, often in a specific problem setting such as sim2real transfer [40, 48, 35, 30] or fast adaptation via meta-learning [9, 34, 31, 67, 10]. A common theme in these works is that the algorithm can often train in preparation for adaptation at test-time, thus affecting the prior experiences it may collect. In contrast, the SLRL setting lays algorithmic emphasis on online exploration and adaptation, as the agent has access to fixed prior dataset of experiences. Other transfer learning approaches adapt the weights of the policy to a new environment or task, either through rapid zero-shot adaptation [19, 61] or through extended episodic online training [22, 39, 7, 59, 58]. Unlike the latter, we focus on adaptation within a single episode, but, unlike the former, with a focus on extended exploration and learning over tens of thousands of timesteps. This problem setting leads to unique challenges, namely that the agent must autonomously recover from mistakes, hence requiring a distinct approach.

3 Preliminaries

In this section, we describe some preliminaries before formalizing our problem statement in the following section. We consider an agent that operates in a Markov decision process (MDP) consisting of the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, \mathcal{R}, \rho, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the agent's action space, $p(s_{t+1}|s_t, a_t)$ represents the environment's transition dynamics, $\mathcal{R} : \mathcal{S} \to \mathbb{R}$ indicates the reward function, $\rho : \mathcal{S} \to \mathbb{R}$ denotes the initial state distribution, and $\gamma \in [0, 1)$ denotes the discount factor. In typical reinforcement learning, the objective is find a policy π that maximizes $J(\pi) = \mathbb{E}_{\tau \sim \pi}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t)]$.

Although our method is a reward-driven RL algorithm, we utilize concepts from imitation learning to overcome sparse rewards, utilizing potentially suboptimal prior data. To this end, we build on adversarial imitation learning (AIL), which uses prior data $\mathcal{D}_{\text{prior}}$ in the form of expert demonstrations (we will relax this requirement) to recover the expert's policy. One such method is GAIL [21], which finds a policy π_θ that minimizes the Jensen-Shannon divergence between its stationary distribution and the expert data. It does so by training a discriminator network $D : \mathcal{S} \times \mathcal{A} \to (0, 1)$, alternating updates with updates to the policy π. Concretely, D and π are learned by optimizing the following: $\min_\pi \max_{D \in (0,1)^{\mathcal{S} \times \mathcal{A}}} \mathbb{E}_\pi[\log(D(s, a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s, a))] - \lambda H(\pi)$. In Section 5, we will see that AIL-style discriminator-based approaches can be adapted to the SLRL problem setting *without* demonstration data, and will therefore form the basis of our method.

4 Single-Life Reinforcement Learning

In this section, we formalize our problem setting, the *single-life reinforcement learning (SLRL)* problem. The defining characteristic of SLRL is that the agent is given a single "life", i.e. trial, to complete a desired task, with the trial ending when the task is completed. The agent must complete the task autonomously, without access to any human interventions or resets.

In the real world, when faced with situations where a task must be completed once, an agent typically has some prior knowledge. E.g., an agent tasked with finding water on Mars may have experience

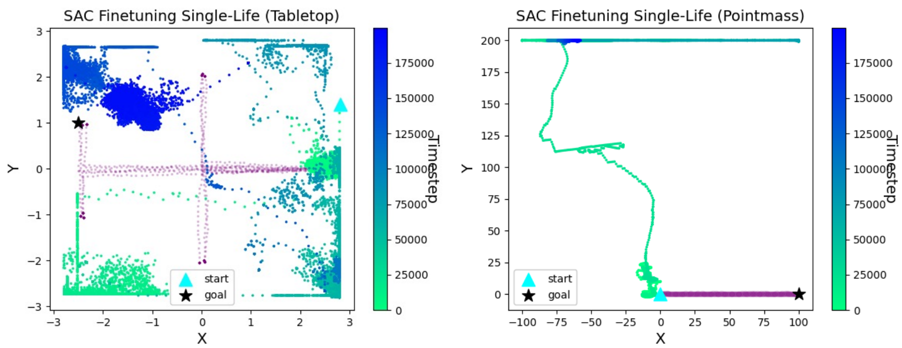

Figure 2: We visualize the online state visitation plots in the Tabletop and Pointmass environments of a single-life trial using SAC finetuning. We plot the location of the mug throughout the agent's single life for the tabletop and the location of the agent for the pointmass, colored green to blue by timestep, along with expert demo states (purple). In both environments, the agent fails to recover from novel states and complete the task.

looking for water in a desert on Earth. We will therefore assume access to offline prior data of some sort that the agent may use for pretraining and during its single life. In many cases such as the Mars example, we may not have expert prior data of the desired task in the desired environment. Hence, we emulate this setting by providing the agent with prior data from a related environment and deploying its single life on a domain with a distribution shift.

We can formalize this setting as follows. We are given prior data $\mathcal{D}_{\text{prior}}$, which consists of transitions from some source MDP $\mathcal{M}_{\text{source}}$. The agent will then interact with a target MDP defined by $\mathcal{M}_{\text{target}} = (\mathcal{S}, \mathcal{A}, p, \mathcal{R}, \rho, \gamma)$. We assume that the the target MDP has an aspect of novelty not present in the source MDP, such as different dynamics $p(s_{t+1} \mid s_t, a_t)$ or a different initial state distribution ρ. Naturally, the more similar the domains are, the easier the problem becomes, and the effectiveness of any algorithm will be strongly dependent on the degree of similarity, though formalizing a precise assumption on similarity between the source and target domain is difficult. The reward between the two MDPs is the same, meaning the agent is still trying to accomplish the same task in the target domain as in the source. The problem setting may be extended to include multiple source MDPs or a series of target MDPs.

The goal in SLRL is to accumulate as much reward as possible in a single trial in $\mathcal{M}_{\text{target}}$. Maintaining the same notation as the previous section, the SLRL problem aims to maximize $J = \sum_{t=0}^{h} \gamma^t \mathcal{R}(s_t)$, where h is the trial horizon, which may be ∞. In general, we expect the task reward to be such that learning *only* from task rewards during the single life is difficult, for example because the reward is very sparse or even awarded only upon successful completion of the task (which can only happen once in the entire single life deployment). We assume that there are no sink states beyond a terminal success state, such that it is possible for the agent to autonomously recover from any mistake.

Note that this setup is essentially the same as the widely studied regret minimization problem in exploration [29]. However, while regret minimization is typically studied in the context of RL exploration theory, our aim with SLRL is to study a particular *special case* of the more general regret minimization framework that is meant to reflect a realistic setting in real-world RL (e.g., robotics) where an agent with prior experience must solve a task in a single (potentially long) trial.

As we analyze in Section 5, algorithms designed for episodic policy learning do not perform well in the SLRL problem setting, even when the policy and replay buffer are pre-trained and seeded with the prior data, because they do not quickly recover from mistakes to get back onto a good state distribution. In Section 6, we will discuss an approach based on distribution matching that attempts to address this issue.

5 Single-Life Performance of Online RL

Now that we have described the problem setting, we now empirically analyze online RL, which is designed for episodic learning, in the SLRL setting. Namely, we consider finetuning SAC [17], which pre-trains a policy and value function in the source setting and fine-tunes during the single trial in the target environment.

As we will see in Section 7, finetuning SAC performs poorly in the SLRL setting. We save the details of the experimental setup until Section 7, but to first motivate the use of distribution-matching approaches in our problem setting, we analyze the state visitation of SAC finetuning in the online phase for the Tabletop and Pointmass domains (see Figure 2). A key challenge of SLRL (and also fully autonomous RL in general) is that if the agent falls off of the distribution, it cannot rely on resets to get back on track. Since there is a gap between the source and target domains, the agent will inevitably find itself in states that are out of distribution from the prior data. A value function pre-trained on the source data could in principle be used to evaluate states and guide the agent back towards good states, but it will be inaccurate on states outside of the prior data [12]; then, when the value of some of those out-of-distributions states is overestimated, the value function may misguide the policy away from good states. Hence, fine-tuning a pre-trained value function via online RL will not explicitly encourage the agent to get back on distribution, especially in sparse reward envs. As a result, the agent may spend a lot of time (perhaps infinite time) drifting once it falls out of distribution, which we see occurs in Figure 2.

On the other hand, distribution-matching methods like GAIL [21] will explicitly encourage the agent to get back on distribution, by giving higher rewards on distribution than off distribution. However, existing adversarial imitation learning methods assume that the prior data consists of expert demonstrations, and they train the agent to match the entire demo distribution, which is not necessarily the ideal distribution to match. In the following section, we will discuss a method that aims to address these shortcomings.

6 Q-weighted Adversarial Learning (QWALE)

In this section, we will present our method for addressing SLRL, which we call QWALE. The key insight in QWALE is to utilize the prior data $\mathcal{D}_{\text{prior}}$ to handle the sparse and uninformative reward information in the target domain. Our first observation is that the framework of AIL already provides a reasonable starting point, though it is not sufficient by itself: rather than using only the task reward, which is too sparse to be useful in a single episode, we can bias the agent to seek out states that are similar to those seen in the prior data. However, since our goal is not to learn a policy that repeatedly performs the task, but rather to solve it as quickly as possible once, we do not actually want to *learn to imitate* prior data, but rather to seek out states that resemble the *best* states in the prior data, with better states being more preferred. This is especially important when the prior data is not actually optimal, but might consist of arbitrarily suboptimal states. We will discuss how this can be accomplished with a modification of AIL which, instead of treating prior data as equally desirable, preferentially drives the agent toward states that resemble the best states in the prior data.

6.1 Algorithm Description

In SLRL, we may have shifts in dynamics online, in which case matching state-action distributions as done in GAIL ([21]) may not be appropriate. GAIL with a state discriminator will help lead the agent back to the prior data distribution but by matching the entire demo state distribution, the discriminator does not necessarily incentivize the agent to go towards task completion. Our algorithm's desired behavior is to lead the agent to nearby states within distribution of the prior data if it is out of distribution and to nearby states closer to task completion if in distribution. Our proposed method for shaping relies on the intuition that as states gradually get closer to task completion, even within expert data, they should have gradually higher values as well.

We propose Q-weighted adversarial learning (QWALE), which trains a Q-weighted discriminator. In order to use the prior data effectively, we use a fixed Q-function $Q(s, a)$ trained in the source MDP to distinguish between useful transitions and ones that may be less useful. This Q-function may be obtained through RL pretraining, which is what we use for our experiments, or a variety of other ways, such as offline RL or Monte Carlo estimation. We train the discriminator in a similar manner as GAIL, where the positives come from the offline data and negatives from online experience. To take into account the varied quality of the data, we use the intuition that the closer a state is to task completion, the higher its value should be. In particular, a state in the prior data should get smaller weight if it has worse value than the agent's current state, so the agent is consistently incentivized to move towards states in the prior data with higher value than its current state. Therefore, when training the discriminator, we weight the positive states s by $\exp(Q(s, a) - b)$ and the negatives by $\exp(-Q(s, a) + b)$, where b is an implementation detail, which we discuss in the Appendix. We normalize the Q-values to be between 0 and 1 and train the Q-weighted discriminator in alternating updates with SAC updates that finetune the policy and critic in an AIL fashion. In this manner, QWALE extends AIL to the general setting with any prior data.

The goal of our weighted discriminator training procedure is to obtain a discriminator that, when used as a reward, will drive the agent toward states that it believes would lead to better outcomes than its present state, based on the prior data. The Q-function quantifies the agent's belief from the prior data that a particular state will lead to high reward, making this a natural choice for estimating how desirable a state is at any given time. Hence, using Q-values to weight the examples for the discriminator will cause the discriminator to prefer states that are closer to the goal over states that are further away. This is significantly different from the behavior we would expect to see if we were simply imitating the optimal policy, as this would give equal weight to all of the transitions along an optimal path. When the reward is more complex, using Q-values as weights generalizes this intuition.

6.2 Practical Implementation

Algorithm 1 Q-WEIGHTED ADVERSARIAL LEARNING (QWALE)

1: // Single Trial Deployment
2: **Require:** $\mathcal{D}_{\text{prior}}$, test MDP $\mathcal{M}_{\text{test}}$, pretrained critic $Q(s,a)$, and (optionally) policy π;
3: **Initialize:** replay buffer for online transitions $\mathcal{D}_{\text{online}}$; parameters ϕ for discriminator $q_\phi(\text{prior} \mid s_t)$, timestep $t = 0$
4: **while** task not complete **do**
5: Sample $a \sim \pi(\cdot \mid s_t)$
6: $\phi \leftarrow \phi - \eta \nabla_\phi L(\phi)$ // Update discriminator according to Eq. 1
7: $r'(s_t) = r(s_t) + \log q_\phi(\text{prior} \mid s_t)$
8: $Q(s,a), \pi \leftarrow \text{SAC}(Q(s,a), \pi, \mathcal{D}_{\text{prior}} \cup \mathcal{D}_{\text{online}}, r')$
9: Increment t

We optimize our objective using maximum entropy off-policy RL with the SAC algorithm ([17]), modified in a similar manner as we did with GAIL in the previous section. In particular, we learn an additional discriminator $q_\theta(\text{prior} \mid s_t)$, optimized using standard cross-entropy loss, which is weighted accordingly:

$$L(\phi) = -\mathbb{E}_{\mathcal{D}_{\text{prior}}}[\exp(Q(s,a) - b) \log q_\phi(\text{prior} \mid s)] - \mathbb{E}_{\mathcal{D}_{\text{online}}}[\exp(-Q(s,a) + b) \log q_\phi(\text{online} \mid s)]. \quad (1)$$

The discriminator is used to modify the rewards when updating off-policy from all experience–prior and online. At single trial test time, the actor and critic are optionally initialized with the pretrained weights, and the replay buffer is initialized with the offline data. For details such as network architecture and hyperparameters, see Appendix A. We present the full algorithm in Algorithm 1.

7 Experiments

The goal of our experiments is to answer the following questions: (1) How does QWALE compare to prior reinforcement learning and distribution matching approaches in single-life RL settings? (2) Do distribution matching approaches help agents learn to recover from novel situations in single-life RL? (3) How does QWALE compare to different variants of adversarial imitation learning, with different prior datasets?

7.1 Experimental Setup

To answer the above questions, we construct four single-life RL domains with varying prior datasets and sources of novelty, and then measure performance both in terms of speed of task completion and overall single-life success. In this subsection, we describe this experimental set-up in detail.

Environments. We consider the following four problem domains. First, in the Tabletop-Organization environment from the EARL benchmark [43], the agent is tasked with bringing a mug to one of four different locations designated by a goal coaster. The prior data always has the same starting position of the mug. In the target environment, the starting position is in a new location unseen in the prior data. Second, the Pointmass setting tasks an agent to move in 2D from its starting location at the origin $(0,0)$ to the point $(100,0)$. The target environment introduces a dynamics shift in the form of a strong "wind", where the agent is involuntarily pushed upward in the y-direction each step. Third, we construct a modified HalfCheetah environment, in which it is difficult but feasible for the cheetah to recover when flipped over. The target environment includes hurdles that the cheetah must jump over,

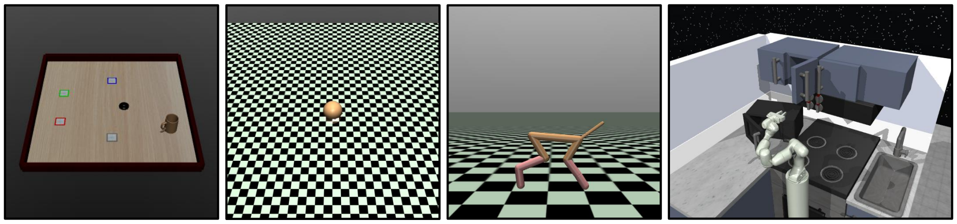

Figure 3: We evaluate in four different domains, including Tabletop-Organization, Pointmass, HalfCheetah, and a Franka-Kitchen environment with a microwave and cabinet. At test time, an aspect of novelty is introduced in each environment–new initial mug positions for Tabletop, wind for Pointmass, hurdles for the HalfCheetah, and a new combination of tasks for the Franka-Kitchen.

as the prior data does not include these obstacles. Finally, we evaluate on a modified Franka-Kitchen environment, adapted from [14], where the task is to close a microwave and a hinged cabinet. The prior data only contains trajectories of closing the microwave and the hinged cabinet separately, so the agent must figure out online how to complete both tasks in a row. In other words, both objects are open at the start of single-life RL, and the agent has only previously seen instances where only one is open. For the latter two environments, dense rewards are given, and the discriminator-based reward is added to the extrinsic reward during single-life training. These environments are shown in Figure 3. Further details on the environments are given in Appendix A.

Comparisons. To answer question (1), we compare QWALE to three alternative methods: (a) SAC fine-tuning, which pre-trains a policy and value function in the source setting and fine-tunes for a single, long episode in the target environment, (b) SAC-RND, which additionally includes an RND exploration bonus [4] during single-life fine-tuning, and (c) GAIL-s, which runs generative adversarial imitation learning [21, 25] where the discriminator only operates on the current state s. We choose for the discriminator to only look at s so that it is less susceptible to dynamics shift between the source data and target environment. We additionally compare to GAIL-sa, which passes both the current state and action to the discriminator. All methods use soft actor-critic (SAC) [17] as the base RL algorithm.

Prior datasets. For all four environments, we evaluate SLRL using data collected through RL as our prior data. More specifically, we run SAC in the source MDP in the standard episodic RL setting for K steps and take the last 50,000 transitions as the prior data. K is chosen such that the prior data contains some good transitions but has not converged to an optimal policy yet. While we are able to run episodic RL in the source MDP, this is not a requirement for SLRL, as long as prior data in the source MDP is available. For all methods, including QWALE, GAIL variants, and SAC fine-tuning variants, the policy and value function are pretrained in this manner for the initialization of single-life RL. We note that AIL methods like GAIL typically assume that the prior data consists of expert demonstrations but we apply the algorithm only using mixed quality prior data, unless otherwise noted. In particular, Section 7.4 further evaluates AIL methods using demos as prior data, using 10 demonstrations for the Tabletop environment and 3 demonstrations for the Pointmass domain. We include such experiments to answer question (3), i.e. to investigate how the quality of prior data may affect performance.

Evaluation Metrics. To evaluate each method in each environment, we report the average and median number of steps taken before task completion across 10 seeds along with the standard error and success rate (out of 10). During single-life RL, for all environments, the agent is given a maximum of 200,000 steps to complete the task. If it has not completed the task after 200k steps, then 200k is logged as the total number of steps, and the run is marked as unsuccessful.

7.2 Results using mixed data as prior data

In this subsection, we aim to answer our first experimental question and study how QWALE performs compared to prior reinforcement learning and distribution matching approaches in single-life RL settings. As seen in Figure 4 and Table 1, we find that QWALE achieves the lowest average and median number of steps as well as highest number of successes on three out of the four domains, and performs comparably to the other methods on the fourth environment, Franka-Kitchen. On the Tabletop and Pointmass environments, QWALE takes less than half of the number of steps on average as the next best performing method. It is possible that the method does not work as well on the Franka-Kitchen environment because the pretrained Q-function may be quite inaccurate, as there is

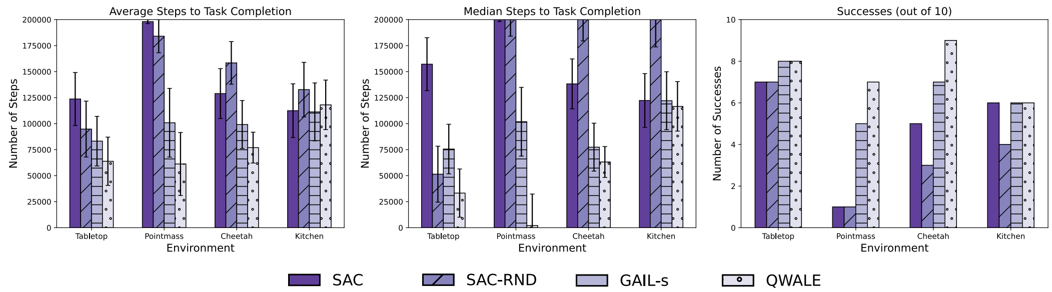

Figure 4: We evaluate the performance of QWALE to finetuning SAC and GAIL in our four environments using mixed data collected through RL as prior data. We omit the results of Behavior Cloning (BC) in the plots, as it is unsuccessful at completing the task in every domain due to the distribution shift. We plot the average and median number of steps to task completion along with the number of successes, taken over 10 seeds. We find that GAIL outperforms SAC in 3 out of 4 domains, and QWALE significantly outperforms GAIL on 3 out of 4 domains and performs comparably on the fourth.

	Method	Avg ± Std error	Success / 10	Median		Method	Avg ± Std error	Success / 10	Median
Tabletop	GAIL-s	83.2k ± 23.8k	**8**	75.6k	Cheetah	GAIL-s	99.2k ± 23.0k	7	77.4k
	GAIL-sa	61.5k ± 28.7k	7	**2.4k**		GAIL-sa	102.0k ± 19.3k	8	85.6k
	QWALE (ours)	**23.1k ± 7.9k**	**8**	15.5k		QWALE (ours)	**77.0k ± 14.8k**	**9**	**63.2k**
Pointmass	GAIL-s	100.9k ± 33.0k	5	101.9k	Kitchen	GAIL-s	**111.3k ± 27.9k**	**6**	122.1k
	GAIL-sa	140.4k ± 30.3k	3	200.0k		GAIL-sa	**127.8k ± 26.9k**	5	189.1k
	QWALE (ours)	**61.2k ± 30.2k**	**7**	**2.1k**		QWALE (ours)	**118.1k ± 23.8k**	**6**	**116.6k**

Table 1: Discriminator-based Approaches on Mixed Data. We see that in each of the four experimental domains, the three QWALE methods typically outperform GAIL and GAIL-sa on the average number of steps needed before task completion, and on 3 out of the 4 environments, QWALE substantially improves performance over both GAIL variants. All methods are evaluated over 10 runs.

a global distribution shift in the state space at test time–the agent has never seen both objects open before in the prior data. GAIL also outperforms finetuning SAC across three of the four domains. These results demonstrate the suitability of distribution-matching approaches over RL finetuning in the SLRL setting. We see that guidance particularly towards a good state distribution is important, as we compare to finetuning SAC with an exploration bonus through random network distillation ([4]). From Figure 4, although RND may improve performance, particularly in the Tabletop domain, it generally does not perform as well as the distribution matching approaches, especially QWALE, showing that simply increasing exploration is not enough. Furthermore, these results show that the additional shaping provided by weighting the prior data by Q-value when training the discriminator can significantly improve guidance towards the goal. While GAIL gives equal weight to all transitions along an optimal path, the agent in QWALE is consistently guided towards states in the prior data with higher Q-value, leading to more efficient and reliable single-life task completion.

7.3 Analysis of distribution matching approaches

Next, to answer question (2), we analyze how QWALE helps agents learn to recover from novel situations in SLRL. To do so, we visualize QWALE's state visitation in the Tabletop and Pointmass domains throughout a single lifetime. We color the trajectories according to timestep for both methods as well as by reward (discriminator score). The coloring in the timestep-colored plots is highly correlated with that in the reward-colored plots, showing how the reward gradually guides the agent towards the goal. In particular, when the agent is out of distribution, the agent is incentivized to explore states that will lead it closer back to the prior state distribution, and when the agent is within distribution, it is incentivized to move to states closer to the goal, leading to efficient task completion.

7.4 Using demos as prior data

Finally, we evaluate the performance of different discriminator-based approaches in the two SLRL problem settings–Tabletop and Pointmass–where demonstration data is available as prior data. We compare using AIL with a state-only discriminator (GAIL-s) as well as with a state-action discriminator (GAIL-sa) to our proposed Q-weighted discriminator method (QWALE). With the latter method, we have access to the same Q-function pretrained when collecting prior data using standard RL for the mixed data experiments above, but we do not initialize any of the algorithms at test time with the pretrained policy and critic weights.

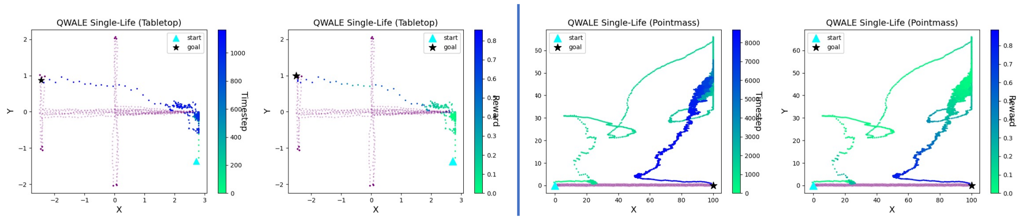

Figure 5: We visualize the online state visitation plots in the Tabletop (left) and Pointmass (right) environments of a single-life trial using QWALE. We plot the location of the mug throughout the agent's single life for the tabletop and the location of the agent for the pointmass as well as expert demo states (purple). We color the trajectories green to blue according to timestep as well by reward (discriminator score) for the distribution-matching approach.

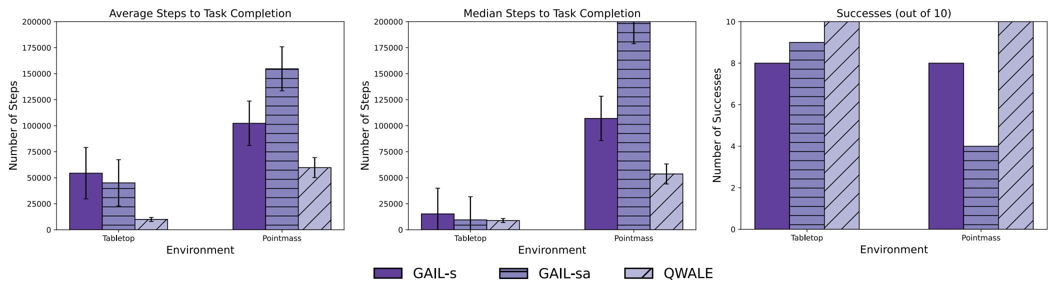

Figure 6: Discriminator-based Approaches using Expert Demo Data. Given demonstration data, in both the Tabletop and Pointmass domains, QWALE significantly outperforms both GAIL variants in almost all metrics.

From Figure 6, the GAIL variants are both able to consistently solve the task in the Tabletop domain, but unsurprisingly, GAIL-sa does especially poorly in the Pointmass domain, where the dynamics have changed at test time. Compared to the two GAIL variants, QWALE gives a significant improvement in both domains. These results demonstrate how more detailed reward shaping towards the completion of the desired task can be helpful in the SLRL setting even with demonstrations as prior data. Moreover, comparing these results with those in Table 1, while access to expert data as prior data unsurprisingly improves the performance of GAIL methods, it can also improve the performance of QWALE.

8 Conclusion

In this paper, we formalized and studied a problem setting underlying single-life reinforcement learning: settings where an agent needs to autonomously complete a task once while drawing upon prior experience from a related environment. We found that standard fine-tuning via RL is ill-suited for this problem because the algorithm struggles to recover from mistakes and novel situations. We hypothesized that this observation stems from the fact that resets in episodic RL prevent algorithms from needing to recover, whereas single-life RL and continuing settings in general do demand the agent to find its way back to good states on its own. We then postulated that distribution matching methods that aim to match the distribution of related prior data may help agents recover via reward shaping, and presented a new distribution matching method, QWALE, that weights examples by their Q-value. Our experiments verified that distribution matching approaches indeed do make better use of prior data, and that QWALE is competitive with or outperforms prior distribution matching methods on four single-life RL problems.

While QWALE can efficiently complete novel target tasks in a single episode without any interventions, important limitations remain. No algorithm, including QWALE, was able to complete the target task with 100% success, indicating that future works should aim to improve an algorithm's ability to solve tasks consistently. Moreover, the methods that we evaluated all used a pre-trained policy and value function from the source domain, which may be difficult to obtain in some source scenarios, as opposed to only obtaining some demonstrations or offline data. Finally, it would be interesting to explore problems with greater degrees of novelty between the source and target environments. We expect that such settings would place even greater importance on autonomy and exploration, requiring sophisticated strategies for both recovering to known states and exploring new strategies. By publicly releasing and open-sourcing the environments and code upon publication, we hope that future work can more easily explore these interesting questions and continue to make progress on allowing RL agents to autonomously complete tasks within a single lifetime.

References

[1] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[2] Mark Beliaev, Andy Shih, Stefano Ermon, Dorsa Sadigh, and Ramtin Pedarsani. Imitation learning by estimating expertise of demonstrators. *arXiv preprint arXiv:2202.01288*, 2022.

[3] Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E Taylor, and Ann Nowé. Reinforcement learning from demonstration through shaping. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

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

[5] Zhangjie Cao, Zihan Wang, and Dorsa Sadigh. Learning from imperfect demonstrations via adversarial confidence transfer. *arXiv preprint arXiv:2202.02967*, 2022.

[6] Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017.

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

[8] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[10] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International Conference on Machine Learning*, pages 1920–1930. PMLR, 2019.

[11] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[12] Justin Fu, Aviral Kumar, Matthew Soh, and Sergey Levine. Diagnosing bottlenecks in deep q-learning algorithms. In *International Conference on Machine Learning*, pages 2021–2030. PMLR, 2019.

[13] Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*, pages 1259–1277. PMLR, 2020.

[14] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

[15] Abhishek Gupta, Justin Yu, Tony Z Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6664–6671. IEEE, 2021.

[16] Abhishek Gupta, Corey Lynch, Brandon Kinman, Garrett Peake, Sergey Levine, and Karol Hausman. Bootstrapped autonomous practicing via multi-task reinforcement learning. *arXiv preprint arXiv:2203.15755*, 2022.

[17] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[18] Weiqiao Han, Sergey Levine, and Pieter Abbeel. Learning compound multi-step controllers under unknown dynamics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6435–6442. IEEE, 2015.

[19] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. *arXiv preprint arXiv:2007.04309*, 2020.

[20] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[21] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[22] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*, 2020.

[23] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

[24] Jigang Kim, J hyeon Park, Daesol Cho, and H Jin Kim. Automating reinforcement learning with example-based resets. *IEEE Robotics and Automation Letters*, 2022.

[25] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*, 2018.

[26] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

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

[28] Vincenzo Lomonaco, Karan Desai, Eugenio Culurciello, and Davide Maltoni. Continual reinforcement learning in 3d non-stationary environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[29] Sridhar Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine learning*, 22(1):159–195, 1996.

[30] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.

[31] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

[32] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[33] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[34] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[35] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[36] 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.

[37] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Paper.pdf`.

[38] 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*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[39] 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.

[40] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[41] Anton Schwartz. A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the tenth international conference on machine learning*, volume 298, pages 298–305, 1993.

[42] Archit Sharma, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Autonomous reinforcement learning via subgoal curricula. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18474–18486. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/99c83c904d0d64fbef50d919a5c66a80-Paper.pdf`.

[43] Archit Sharma, Kelvin Xu, Nikhil Sardana, Abhishek Gupta, Karol Hausman, Sergey Levine, and Chelsea Finn. Autonomous reinforcement learning: Formalism and benchmarking. *arXiv preprint arXiv:2112.09605*, 2021.

[44] Archit Sharma, Rehaan Ahmad, and Chelsea Finn. A state-distribution matching approach to non-episodic reinforcement learning. *arXiv preprint arXiv:2205.05212*, 2022.

[45] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. *arXiv preprint arXiv:1904.07854*, 2019.

[46] Mingfei Sun and Xiaojuan Ma. Adversarial imitation learning from incomplete demonstrations. *arXiv preprint arXiv:1905.12310*, 2019.

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

[48] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[49] Faraz Torabi, Garrett Warnell, and Peter Stone. Adversarial imitation learning from state-only demonstrations. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2229–2231, 2019.

[50] Mel Vecerik, 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.

[51] Qing Wang, Jiechao Xiong, Lei Han, Han Liu, Tong Zhang, et al. Exponentially weighted imitation learning for batched historical data. *Advances in Neural Information Processing Systems*, 31, 2018.

[52] Ruohan Wang, Carlo Ciliberto, Pierluigi Amadori, and Yiannis Demiris. Support-weighted adversarial imitation learning. *arXiv preprint arXiv:2002.08803*, 2020.

[53] Yunke Wang, Chang Xu, and Bo Du. Robust adversarial imitation learning via adaptively-selected demonstrations. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 3155–3161. International Joint Conferences on Artificial Intelligence Organization, 2021.

[54] Yunke Wang, Chang Xu, Bo Du, and Honglak Lee. Learning to weight imperfect demonstrations. In *International Conference on Machine Learning*, pages 10961–10970. PMLR, 2021.

[55] Chen-Yu Wei, Mehdi Jafarnia Jahromi, Haipeng Luo, Hiteshi Sharma, and Rahul Jain. Model-free reinforcement learning in infinite-horizon average-reward Markov decision processes. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10170–10180. PMLR, 13–18 Jul 2020.

[56] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[57] Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. Imitation learning from imperfect demonstration. In *International Conference on Machine Learning*, pages 6818–6827. PMLR, 2019.

[58] Annie Xie and Chelsea Finn. Lifelong robotic reinforcement learning by retaining experiences. *arXiv preprint arXiv:2109.09180*, 2021.

[59] Annie Xie, James Harrison, and Chelsea Finn. Deep reinforcement learning amidst lifelong non-stationarity. *arXiv preprint arXiv:2006.10701*, 2020.

[60] Ju Xu and Zhanxing Zhu. Reinforced continual learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/cee631121c2ec9232f3a2f028ad5c89b-Paper.pdf`.

[61] Takuma Yoneda, Ge Yang, Matthew R Walter, and Bradly Stadie. Invariance through inference. *arXiv preprint arXiv:2112.08526*, 2021.

[62] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[63] Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. The ingredients of real-world robotic reinforcement learning. *arXiv preprint arXiv:2004.12570*, 2020.

[64] Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. Off-policy imitation learning from observations. In *the Thirty-fourth Annual Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

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

[66] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Modeling interaction via the principle of maximum causal entropy. In *ICML*, 2010.

[67] 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.

A Appendix

A.1 Implementation Details and Hyperparameters

In our experiments, we use soft actor-critic [17] as our base RL algorithm. We use default hyperparameter values: a learning rate of 3e-4 for all networks, optimized using Adam, with a batch size of 256 sampled from the entire replay buffer (both prior and online data), a discount factor of 0.99. The policy and critic networks are MLPs with 2 fully-connected hidden layers of size 256. For all methods training a discriminator, it is parameterized as an MLP with 1 fully-connected hidden layer of size 128 and trained with a batch size of 512. During the online trial, 1000 steps are taken as initial collection steps before network updates begin. For all methods training a discriminator, we use mixup regularization [62] to reduce the brittleness of the discriminator.

Following [43], we use a biased TD update, where $Q(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1})$ if t is not a multiple of 100, and $Q(s_t, a_t) \leftarrow r(s_t, a_t)$ if it is. We use this update for all our evaluated methods online in order to improve stability. Since the online trial of single-life RL may have a large training horizon with hundreds of thousands of steps, this may lead to unstable bootstrapping, as for each t, $Q(s_t, a_t)$ bootstraps on $Q(s_{t+1}, a_{t+1})$. Following [44], for the auxiliary reward given by the discriminator D, we use $r(s, a) = -\log(1 - D(s))$ instead of $r(s, a) = \log D(s)$ to further improve stability.

For QWALE, the weighting of states is offset by a value b. This value may be treated like a constant hyperparameter and tuned. Adding this value changes the bias on the discriminator, which in effect adds a constant to the reward, though that constant changes over the course of training. In practice, to avoid having to tune b, we just use the value of the most recent state as b, i.e. $b = Q(s_t, a_t)$. To better interpret this value, with this weighting, b is a baseline value capturing some notion of current progress. Prior data tends to get small weights if they have worse value than the current state, so the agent is consistently incentivized to move towards states with higher value than its current state.

For all experiments using prior data collected through RL, the agent was initialized at test time with the pretrained policy and critic. For QWALE, a copy of that critic was frozen and used when calculating the weights for discriminator training. For all of the experiments with demonstration data in Section 7.4, the policy and critic were not initialized with any pretrained weights.

A.2 Environment & Evaluation Details

Tabletop-Organization. The details for this environment are in [43]. The state space consists of the gripper's (x, y) position, the mug's (x, y) position, the gripper's state (whether attached to the mug or not), and the current goal, for a total of 12 dimensions. The action space is 3 dimensional, consisting of a delta in the gripper's (x, y) position as well as an automatic gripper that will attach to the mug if the gripper is close enough. The tabletop extends from -2.8 to 2.8 in both the x and y directions. In the prior data, which consists either of 10 demonstrations or 50000 transitions collected through RL after 350000 steps of training, the initial state always places the mug at position (2.5, 0.0), and the goal is to place the mug at one of the following locations: (-2.5, -1.0), (-2.5, 1.0), (0, 2.0), (0, -2). For the online trial when evaluating SLRL, the mug is placed either at (2.7, 1.5) or (2.7, -1.5) with additional uniform randomness between (-0.15, 0.15) in both directions. This environment is also goal-conditioned at test time and the goal is randomly set to be either (-2.5, -1.0) or (-2.5, 1.0). The reward is 1 when the mug is within 0.15 distance of its goal position (at which point the single life ends) and 0 everywhere else.

Pointmass. The Pointmass environment has a 6-dimensional state space consisting of the agent's (x, y) position, its (x, y) velocity, and the (x, y) coordinates of the goal. The environment extends between -100 and 100 along the x axis and between -200 and 200 along the y axis. The action space is 2-D, consisting of the delta in both directions, clipped between -1 and 1 for a single action. The prior data consists of 3 demonstrations or 50000 transitions collected through RL after 350000 steps of training. The agent starts at (0, 0) and the goal is at (100, 0) for the prior data and online trial. During the online trial, a strong "wind" is introduced, where a random amount between 0.8 and 0.9 is added to the agent's y coordinate and 0.2 is subtracted from the agent's x coordinate at each step. The reward is 1 when the agent is within a distance of 2 of the goal position and 0 everywhere else.

HalfCheetah. The HalfCheetah environment has a state space with 18 dimensions, consisting of the position and velocity of each joint. The prior data consists of 50000 transitions collected through RL after 150000 steps of training. The reward is $r_t = \Delta x_t - 0.1 * ||a_t||_2^2$. At test time, 10 hurdles are included in the environment, spread between the x-coordinate of 7 and 260. The cheetah starts at 0 and its single life is considered successful when it gets to the coordinate 300, although the information about the hurdles or goal are not included in the state space.

Franka-Kitchen. The Franka-Kitchen is adapted from [14, 43]. The state space consists of a 9 DoF position-controlled Franka-robot with a microwave and hinged cabinet. The prior data consists of 50000 transitions collected through standard episodic RL after 950000 steps of training, where one of the microwave or cabinet is open, and the task is to close that object. At test time, both are open, and the task is to close both objects. The reward function is equal to the sum of the Euclidean distance between the objects and their goal positions and the distance between the arm and its goal position.