You Only Live Once: Single-Life Reinforcement Learning

Annie S. Chen[1], **Archit Sharma**[1], **Sergey Levine**[2], **Chelsea Finn**[1]
Stanford University[1], UC Berkeley[2]
asc8@stanford.edu

Abstract

Reinforcement learning algorithms are typically designed to learn a performant policy that can repeatedly and autonomously complete a task, usually starting from scratch. However, in many real-world situations, the goal might not be to learn a policy that can do the task repeatedly, but simply to perform a new task successfully once in a single trial. For example, imagine a disaster relief robot tasked with retrieving an item from a fallen building, where it cannot get direct supervision from humans. It must retrieve this object within one test-time trial, and must do so while tackling unknown obstacles, though it may leverage knowledge it has of the building before the disaster. We formalize this problem setting, which we call *single-life reinforcement learning* (SLRL), where an agent must complete a task within a single episode without interventions, utilizing its prior experience while contending with some form of novelty. SLRL provides a natural setting to study the challenge of autonomously adapting to unfamiliar situations, and we find that algorithms designed for standard episodic reinforcement learning often struggle to recover from out-of-distribution states in this setting. Motivated by this observation, we propose an algorithm, Q-weighted adversarial learning (QWALE), which employs 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 states. [1]

1 Introduction

Agents operating in the real world must often contend with novel situations that differ from their prior experience. For example, a search-and-rescue disaster relief robot may encounter novel obstacles while traversing a building. The vast range of situations that could be encountered at test time cannot be anticipated in advance during training, and therefore agents that can adapt at test time will be better equipped to cope with new situations. Reinforcement learning (RL) algorithms in principle provide a suitable paradigm for such adaptation. However, much of the work in RL focuses on learning an optimal policy that can repeatedly solve a given task, whereas on-the-fly adaptation simply requires the agent to complete the task once without human interventions or supervision, presenting a different set of challenges. In this work, we formalize this problem setting as *single-life reinforcement learning* (SLRL), where the agent is evaluated on its ability to successfully and autonomously adapt to a novel scenario within a single trial.

Given some prior experience, SLRL induces the challenge of contending with novelty (e.g., a new initial state distribution or new dynamics) at test time without episodic resets. A key problem that makes this difficult is that the agent will need to recover from unfamiliar situations without any

[1]Project website: https://sites.google.com/stanford.edu/single-life-rl

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

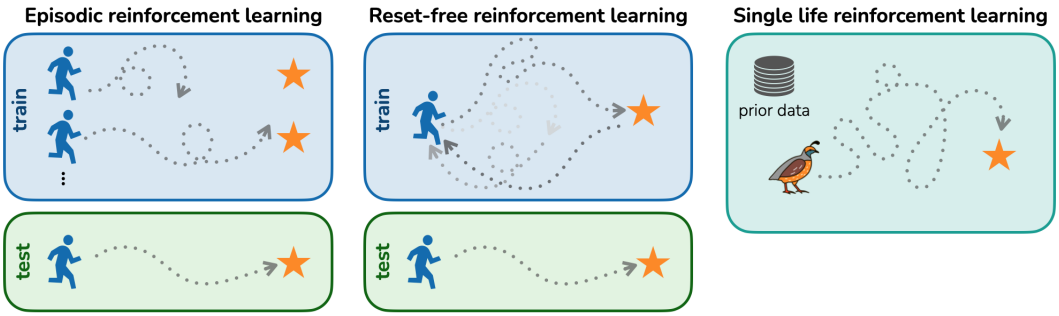

Figure 1: We study the single-life reinforcement learning (SLRL) problem, where given prior data, an agent must complete a task autonomously in a single trial in a domain with a novel distribution shift.

interventions, which can be especially challenging when informative shaped rewards are unavailable, as is often the case in the real world. In fact, task rewards are sometimes only provided at the end of the life, e.g. the search-and-rescue robot may only receive reward signal after it maneuvers to find a desired object and may therefore get stuck behind unknown debris. In episodic RL, the agent can rely on a reset to recover from an unfamiliar state. In contrast, in SLRL, the agent only has its prior experience to leverage in order to find its way back to a good state distribution. Consequently, we find that fine-tuning a pre-trained value function via online RL will struggle in SLRL settings, as it does not explicitly encourage the agent to return to its known distribution and complete the desired task.

To better handle the SLRL setting, we need a way to provide intermediate shaped rewards that guide the adaptation process, and help the agent recover when it gets stuck. One potential option for providing the desired guidance is to use reward shaping towards the agent's distribution of prior experience. Adversarial imitation learning (AIL) approaches such as GAIL (Ho and Ermon [2016]) can potentially provide such reward shaping, but 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 Q-weighted adversarial learning (QWALE), in which different states in the prior data are weighted different amounts by their estimated Q-value. QWALE incentivizes the agent to move towards states in the prior data with higher values, so that agent is more consistently guided towards states closer to task completion.

Our contributions are as follows. We formalize the SLRL problem setting, which provides a natural setting in which to study autonomous adaptation to novel situations in reinforcement learning. Moreover, we believe SLRL is a useful framework for modeling many situations in the real world. We explore reward shaping through distribution matching as one potential choice of methods for SLRL and identify challenges that uniquely arise in the SLRL setting with existing distribution matching approaches. We 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 find that QWALE can meaningfully guide the agent to recover to the prior data distribution to complete the desired task 20-60% more successfully on four separate domains compared to existing distribution matching approaches and RL fine-tuning. QWALE provides a strong baseline for developing algorithms that can better adapt to novelty online and recover from out-of-distribution states.

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 [Han et al., 2015, Eysenbach et al., 2017, Zhu et al., 2020a, Sharma et al., 2021a, 2022, Gupta et al., 2021, 2022, Kim et al., 2022, Sharma et al., 2021b]. 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 [Mahadevan, 1996, Schwartz, 1993, Sutton and Barto, 2018, Xu and Zhu, 2018, Rolnick et al., 2019, Lomonaco et al., 2020, Wei et al., 2020] 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 with a distribution shift. 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 [Argall et al., 2009, Ross et al., 2011, Ghasemipour et al., 2020], inverse RL [Ng et al., 2000, Finn et al., 2016, Ziebart et al., 2008, 2010], RL for sparse reward settings [Brys et al., 2015, Nair et al., 2018, Rajeswaran et al., 2017, Hester et al., 2018, Vecerik et al., 2017] and offline RL [Wang et al., 2020b, Levine et al., 2020, Kumar et al., 2020, Kidambi et al., 2020, Wu et al., 2019a, Nair et al., 2018]. 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 [Ho and Ermon, 2016, Fu et al., 2017, Singh et al., 2019, Torabi et al., 2019, Kostrikov et al., 2018, Zhu et al., 2020b] 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 [Wang et al., 2018, Sun and Ma, 2019, Wu et al., 2019b, Wang et al., 2020a, 2021b,a, Cao et al., 2022, Beliaev et al., 2022]. However, as we discuss in Section 5, 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 [Sadeghi and Levine, 2016, Tobin et al., 2017, Peng et al., 2018, Mehta et al., 2020] or fast adaptation via meta-learning Finn et al. [2017], Nichol et al. [2018], Nagabandi et al. [2018], Zintgraf et al. [2019], Finn et al. [2019]. 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 a 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 Hansen et al. [2020], Yoneda et al. [2021] or through extended episodic online training Khetarpal et al. [2020], Rusu et al. [2016], Eysenbach et al. [2020], Xie et al. [2020], Xie and Finn [2021]. 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 $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho_0, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the agent's action space, $\mathcal{P}(s_{t+1}|s_t, a_t)$ represents the environment's transition dynamics, $\mathcal{R} : \mathcal{S} \to \mathbb{R}$ indicates the reward function, $\rho_0 : \mathcal{S} \to \mathbb{R}$ 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 [Ho and Ermon, 2016], 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:

$$\begin{aligned} &\min_{\pi} \max_{D \in (0,1)^{\mathcal{S} \times \mathcal{A}}} \mathbb{E}_{\pi_E}[\log(D(s,a))] + \mathbb{E}_{\pi}[\log(1 - D(s,a))] - \lambda \mathcal{H}(\pi) \\ &= \min_{\pi} \mathcal{D}_{\text{JS}}(\rho^{\pi}(s,a) \mid\mid \rho^{*}(s,a)) - \lambda \mathcal{H}(\pi), \end{aligned} \tag{1}$$

where $\mathcal{D}_{\text{JS}}$ is the Jensen-Shannon divergence and $\rho^*(s, a)$ is the optimal state-action occupancy measure of the expert π_E (represented by demonstrations). In Section 6, 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 the *single-life reinforcement learning (SLRL)* problem. The defining characteristic of SLRL is that the agent is given a single "life" (i.e., one long 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. Furthermore, as is the case in many real-world situations, when faced with situations where a task must be completed once, informative shaped rewards are not easily obtained, and so the single life might have a reward only once after the task has been completed. However, an agent typically has some prior knowledge. E.g., an agent tasked with retrieving a valuable from a burning building may have experience finding items in that building before the fire. 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 disaster relief 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 $\mathcal{M}_{\text{target}} = (\mathcal{S}, \mathcal{A}, \tilde{\mathcal{P}}, \mathcal{R}, \tilde{\rho}_0, \gamma)$. We assume that the the target MDP has an aspect of novelty not present in the source MDP, such as different dynamics $\tilde{\mathcal{P}}(s_{t+1} \mid s_t, a_t)$ or a different initial state distribution $\tilde{\rho}$. 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.

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, as the reward may be very sparse or even awarded only upon successful completion of the task. 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 [Mahadevan, 1996]. 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 6, 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 5, we will discuss an approach based on distribution matching that attempts to address this issue.

5 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 efficiently complete the task, especially when the reward information is sparse in the target domain. The framework of AIL provides a reasonable starting point, though it is not sufficient by itself: existing AIL methods assume that the prior data

consists of expert demonstrations, and they train the agent to match the entire demo distribution. 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.

5.1 Algorithm Description

Our algorithm's desired behavior is to lead the agent towards the distribution of prior data, therefore helping it recover when it falls into out-of-distribution states, while also pushing it towards task completion. GAIL [Ho and Ermon, 2016] offers a framework to shape the state-action distribution towards that of an expert. In particular, GAIL minimizes $\mathcal{D}_{\text{JS}}(\rho^\pi(s,a) \;||\; \rho^*(s,a))$ (ignoring the causal entropy $\mathcal{H}(\pi)$). In SLRL, we relax this assumption, as we may only have access to suboptimal offline prior data. As such, we desire an algorithm that will be agnostic to the quality of prior data. Hence, instead of learning a policy that aims to uniformly match the entire state-action distribution of the prior data, we want to match the state-action pairs that will lead to task completion.

The problem of matching the state-action distribution has been studied in stochastic optimal control, particularly in REPS [Peters et al., 2010]. Roughly, this problem can be framed as the RL problem with an additional constraint encouraging the learned distribution to be close to the distribution of the prior data. The solution to this optimization problem gives us that our desired target state-action distribution has the form $\rho^*_{\text{target}}(s,a) \propto \rho_\beta(s,a)\exp(Q^{\pi_{\text{target}}}(s,a) - V^{\pi_{\text{target}}}(s))$, where the corresponding policy can be derived by using $\pi_{\text{target}} = \rho^*_{\text{target}}(s,a)/\int_{\mathcal{A}} \rho^*_{\text{target}}(s,a')\mathrm{d}a'$ using [Syed et al., 2008]. For a formal description of the problem, see Appendix A.1.

Thus, replacing ρ^*_{target} as the target distribution instead of ρ^*, we propose Q-weighted adversarial learning (QWALE), which minimizes $\mathcal{D}_{\text{JS}}(\rho^\pi \;||\; \rho^*_{\text{target}})$ as follows:

$$\begin{aligned}
\min_\pi \mathcal{D}_{\text{JS}}(\rho^\pi(s,a) \;||\; \rho^*_{\text{target}}(s,a)) &= \min_\pi \max_D \mathbb{E}_{s,a\sim\rho^*_{\text{target}}}\left[\log D(s,a)\right] + \mathbb{E}_{s,a\sim\rho^\pi}\left[\log(1-D(s,a))\right] \\
&= \min_\pi \max_D \mathop{\mathbb{E}}_{s,a\sim\rho^\beta}\left[\frac{\exp(Q^{\pi_{\text{target}}}(s,a) - V^{\pi_{\text{target}}}(s))}{\mathbb{E}_{\rho^\beta}\left[\exp(Q^{\pi_{\text{target}}}(s,a) - V^{\pi_{\text{target}}}(s))\right]}\log D(s,a)\right] + \mathop{\mathbb{E}}_{s,a\sim\rho^\pi}\left[\log(1-D(s,a))\right] \\
&\equiv \min_\pi \max_C \mathbb{E}_{s,a\sim\rho^\beta}\left[\exp\left(Q^{\pi_{\text{target}}}(s,a) - V^{\pi_{\text{target}}}(s)\right)\log D(s,a)\right] + \mathbb{E}_{s,a\sim\rho^\pi}\left[\log(1-D(s,a))\right],
\end{aligned}$$

where the constant $\mathbb{E}_{\rho^\beta}\left[\exp(Q^{\pi_{\text{target}}}(s,a) - V^{\pi_{\text{target}}}(s))\right]$ can be ignored.

Hence, QWALE trains a weighted discriminator using 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 offline data and negatives from online experience.

When training the discriminator, we weight the positive states s by $\exp(Q(s,a) - b)$, where b is an implementation detail treated as a hyperparameter. Following Abdolmaleki et al. [2018], we exclude $V^{\pi_{\text{target}}}(s)$ in the weighting term in our practical implementation. Including the term b changes the bias on the discriminator, which in effect just adds a constant to the reward. In practice, to avoid having to tune b, we just use the value of the most recent state as b, i.e. $b = Q(s_t, a_t)$. We include additional discussion in the Appendix. We normalize the Q-values to be between 0 and 1 by subtracting the min and dividing by the (max - min) and train the Q-weighted discriminator in alternating 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.

Intuitively, 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

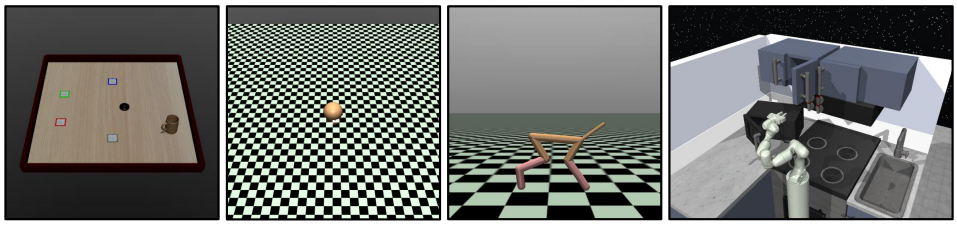

Figure 2: We evaluate in four different domains, including Tabletop-Organization, Pointmass, HalfCheetah, and a Franka-Kitchen environment with a microwave and cabinet. At test time, an aspect of novelty is introduced in each environment–new initial mug positions for Tabletop, wind for Pointmass, hurdles for the HalfCheetah, and a new combination of tasks for the Franka-Kitchen.

were simply imitating the optimal policy, which would give equal weight to all of the transitions along an optimal path.

5.2 Practical Implementation

Algorithm 1 Q-WEIGHTED ADVERSARIAL LEARNING (QWALE)

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

We optimize our objective using maximum entropy off-policy RL with the SAC algorithm (Haarnoja et al. [2018]), modified in a similar manner as we did with GAIL in the previous section. In particular, we learn an additional discriminator $q_\phi(\text{prior} \mid s_t)$, optimized using standard cross-entropy loss, which is weighted accordingly.

$$L(\phi) = -\mathbb{E}_{\mathcal{D}_{\text{prior}}}[\exp(Q(s, a) - b) \log q_\phi(\text{prior} \mid s)] - \mathbb{E}_{\mathcal{D}_{\text{online}}}[\log q_\phi(\text{online} \mid s)]. \tag{2}$$

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.

6 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?

6.1 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 Sharma et al. [2021b], 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, as the prior data does not include these obstacles. Finally, we evaluate on a modified Franka-Kitchen environment, adapted from Gupta et al. [2019], 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 2. 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 Burda et al. [2018] during single-life fine-tuning, and (c) GAIL-s, which runs generative adversarial imitation learning Ho and Ermon [2016], Kostrikov et al. [2018] 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) Haarnoja et al. [2018] as the base RL algorithm. We present additional comparisons in the Appendix.

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 6.4 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.

6.2 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 Table 1, we find that QWALE achieves the lowest average and median number of steps as well as highest number of successes on all four domains. In particular, QWALE takes 20-40% fewer steps on average as the next best performing method. GAIL-s and GAIL-sa also outperform 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 (Burda et al. [2018]). From Table 1, 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

	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
	SAC	123.7k ± 25.5k	7	157.2k		SAC	128.9k ± 23.9k	5	138.2k
	SAC-RND	94.8k ± 26.9k	7	51.4k		SAC-RND	158.4k ± 20.5k	3	200.0k
	QWALE (ours)	**44.4k ± 24.6k**	**9**	8.9k		QWALE (ours)	**74.3k ± 9.5k**	**10**	**68.9k**
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
	SAC	198.1k ± 1.9k	1	200.0k		SAC	112.5k ± 25.8k	6	122.3k
	SAC-RND	184.1k ± 15.9k	1	200.0k		SAC-RND	132.7k ± 26.2k	4	200.0k
	QWALE (ours)	**61.0k ± 28.8k**	**8**	**1.7k**		QWALE (ours)	**79.9k ± 21.7k**	**8**	**59.3k**

Table 1: 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), as it is unsuccessful at completing the task in every domain due to the distribution shift. We find that GAIL outperforms finetuning SAC and SAC-RND in 3 out of 4 domains, and QWALE outperforms both GAIL variants on all 4 domains on the average number of steps before task completion. All methods are evaluated over 10 seeds.

Q-value, leading to more efficient and reliable single-life task completion. Additional comparisons are presented in the Appendix.

6.3 Recovering from novel situations in SLRL

Next, to answer question (2), we analyze how QWALE helps agents learn to recover from novel situations in SLRL. First, we analyze the state visitation of SAC finetuning in the online phase for the Tabletop and Pointmass domains. 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 Fu et al. [2019]; 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 3.

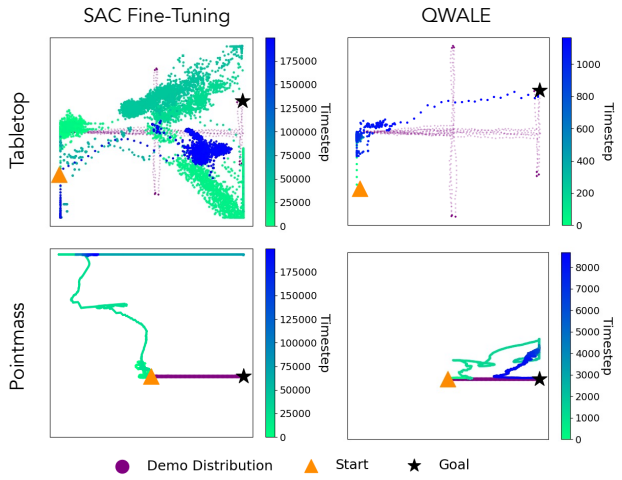

Figure 3: We visualize the online state visitation of the mug in the Tabletop (top) and the agent in the Pointmass (bottom) environments of a single-life trial using SAC finetuning (left) and QWALE (right). The expert states are in purple, and we color the trajectories green to blue according to timestep for both methods. See Section 5 for plots colored according to reward. In both environments, the SAC fine-tuning agent fails to recover from novel states, but is guided towards task completion with QWALE.

On the other hand, distribution-matching methods like QWALE will explicitly encourage the agent to get back on distribution, by giving higher rewards on distribution than off distribution. In Figure 3, 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, and include the trajectories colored by reward (discriminator score) in Figure 5. The coloring in the timestep-colored plots is highly correlated with that in the reward-colored plots, showing how the reward helps the agent recover from unfamiliar states and 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

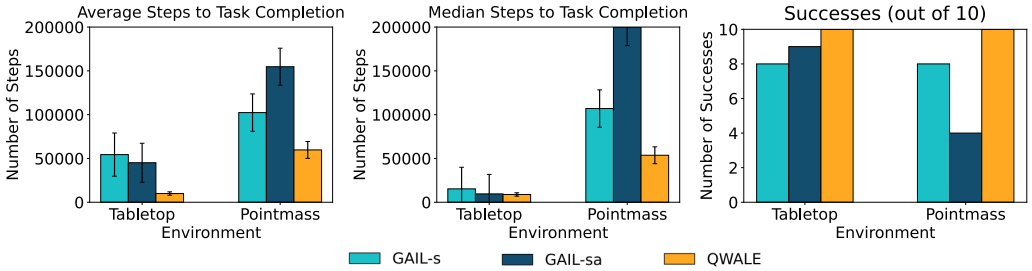

Figure 4: Discriminator-based approaches to single-life RL with expert demonstrations as the prior data. Given demonstration data, in both the Tabletop and Pointmass domains, QWALE matches or significantly outperforms both GAIL variants on all metrics.

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.

6.4 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.

From Figure 4, 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.

7 Conclusion

In this paper, we formalized and studied the problem setting of single-life reinforcement learning: a setting where an agent needs to autonomously complete a task once in a single trial 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 happens since 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 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. In our experimental domains, the reward and state space between the source and target MDPs were assumed to stay the same. Eventually, we hope that this work may be extended to the case where the reward may be different in the online trial from the prior data. We expect that such settings would place even greater importance on autonomy and exploration, requiring sophisticated strategies for adapting

online to novelty. We hope that future work can explore these interesting questions and continue to make progress on allowing RL agents to autonomously complete tasks within a single lifetime.

Acknowledgments

We thank members of the IRIS lab for helpful discussions on this project. This research was supported by Google and the Office of Naval Research (ONR) via grants N00014-21-1-2685 and N00014-20-1-2675. Annie Chen is supported by an NSF graduate student fellowship. Chelsea Finn is a CIFAR fellow.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

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.

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

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.

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

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

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.

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.

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.

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.

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

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

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.

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.

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.

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.

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.

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.

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.

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.

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.

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

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

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.

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

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.

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.

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.

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.

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

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.

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.

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.

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

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

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.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

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.

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`.

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.

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.

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

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.

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., 2021a. URL `https://proceedings.neurips.cc/paper/2021/file/99c83c904d0d64fbef50d919a5c66a80-Paper.pdf`.

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*, 2021b.

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

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.

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

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

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

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.

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.

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.

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.

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

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, 2021a.

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

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *Advances in Neural Information Processing Systems*, 33:7768–7778, 2020b.

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.

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

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, 2019b.

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

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

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`.

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

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

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*, 2020a.

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)*, 2020b.

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.

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

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.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We do not expect any negative societal impacts that are unique to this work compared to other works on reinforcement learning.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Included in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Included in the supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See experiments in Section **??**.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Included in the supplemental material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Additional Method Justification

The key idea of QWALE 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. GAIL finds a policy whose occupancy measure minimizes the Jensen-Shannon divergence $\mathcal{D}_{\text{JS}}$ to the prior data. Since our objective in SLRL is to finish the task as soon as possible, and we may not be given expert demonstrations as prior data, we want to match the state-action pairs to those that lead to task completion.

To this end, we consider the following optimization problem:

$$\max_{\rho} \mathbb{E}_{\rho}\left[r(s,a)\right] = \max_{\rho} \int_{\mathcal{S}\times\mathcal{A}} \rho(s,a) r(s,a) \mathrm{d}s\mathrm{d}a \tag{3}$$

$$\text{s.t.} \quad \int_{\mathcal{S}\times\mathcal{A}} \rho(s,a)\mathrm{d}s\mathrm{d}a = 1, \quad \rho(s,a) \geq 0 \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A}, \tag{4}$$

$$\int_{\mathcal{A}} \rho(s,a)\mathrm{d}a = (1-\gamma)\rho_0(s) + \gamma \int_{\mathcal{S}\times\mathcal{A}} \rho(s',a')\mathcal{P}(s \mid s',a')\mathrm{d}s'\mathrm{d}a' \quad \forall s \in \mathcal{S}, \tag{5}$$

$$\mathrm{KL}\left(\rho(s,a) \mid\mid \rho_{\beta}(s,a)\right) \leq \epsilon. \tag{6}$$

Equations (3- 5) represent the reinforcement learning problem stated in the formalism of convex optimization, where ρ represents the state-action distribution being optimized, Eq 4 constrains ρ to be a valid probability distribution and Eq 5 constrains ρ to be consistent with the MDP. The state-distribution ρ^* of the expert policy π_E is a solution of this problem. By introducing the constraint in Eq 6, we want to encourage the optimal distribution $\rho^*_{\text{target}}(s,a)$ to be *soft* by being close to the prior state-action distribution ρ_β (the state-action distribution of $\mathcal{D}_{\text{prior}}$ corresponding to some behavior policy β). This problem has been studied in stochastic optimal control, particularly REPS [Peters et al., 2010]. REPS shows that the solution to the optimization problem in Eq(3- 6) has the form $\rho^*_{\text{target}}(s,a) \propto \rho_\beta(s,a)\exp(Q^{\pi_{\text{target}}}(s,a) - V^{\pi_{\text{target}}}(s))$, where the corresponding policy can be derived by using $\pi_{\text{target}} = \rho^*_{\text{target}}(s,a)/\int_{\mathcal{A}} \rho^*_{\text{target}}(s,a')\mathrm{d}a'$ from [Syed et al., 2008].

A.2 Implementation Details and Hyperparameters

In our experiments, we use soft actor-critic [Haarnoja et al., 2018] 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 Zhang et al. [2017] to reduce the brittleness of the discriminator.

Following [Sharma et al., 2021b], 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 [Sharma et al., 2022], 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 6.4, the policy and critic were not initialized with any pretrained weights.

A.3 Environment & Evaluation Details

Tabletop-Organization. The details for this environment are in [Sharma et al., 2021b]. 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 [Gupta et al., 2019, Sharma et al., 2021b]. 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.

A.4 Additional Experiments

We run three additional variants of SAC finetuning on all four experimental domains using mixed prior data. More specifically, we compare to (a) SAC-no online, in which we pretrain SAC in the source environment and evaluate in the target domain without additional finetuning, (b) SAC-scratch, where we train SAC completely from scratch in the target domain during its single episode, and (c) SAC-BC, where we fine-tune SAC like the comparison in Section 6 but add an additional behavior cloning loss on the offline data. QWALE performs significantly better than three additional variants across all four domains. We find that SAC-no online performs poorly on all four test domains, which is unsurprising due to the distribution shift between the source and target environments. Adding a behavioral cloning loss does help in some domains but only if the dynamics are not changed–the method struggles in the Pointmass environment due to the changing dynamics at test time. SAC-scratch struggles compared to QWALE and distribution-matching approaches but sometimes performs better than finetuning SAC, such as in the Tabletop domain, showing that the distribution shift is so significant in some domains that pretraining in the source domain is counterproductive when deployed in the target domain.

In addition, we run a comparison with a state-of-the-art reset-free algorithm, MEDAL Sharma et al. [2022], as well as an additional comparison with Progressive Networks Rusu et al. [2016], a strong continual learning method, in the SLRL setting in the Tabletop domain. We include these comparisons

	Method	Avg ± Std error	Success / 10	Median		Method	Avg ± Std error	Success / 10	Median
Tabletop	SAC-no online	181.5k ± 17.9k	2	200.0k	Cheetah	SAC-no online	143.7k ± 28.6k	3	200.0k
	SAC-scratch	99.5k ± 22.5k	8	98.8k		SAC-scratch	200.0k ± 0	0	200.0k
	SAC-BC	80.8k ± 32.5k	6	1.5k		SAC-BC	93.5k ± 23.6k	7	60.6k
Pointmass	SAC-no online	200.0k ± 0	0	200.0k	Kitchen	SAC-no online	200.0k ± 0	0	200.0k
	SAC-scratch	120.8 ± 18.5k	8	112.9k		SAC-scratch	155.0k ± 24.5k	3	200.0k
	SAC-BC	200.0k ± 0	0	200.0k		SAC-BC	101.1k ± 28.1k	6	71.7k

Table 2: We evaluate the performance of three additional SAC variants on our four experimental domains using mixed data collected through RL as prior data. QWALE significantly outperforms all variants on all 4 domains on the average number of steps before task completion. We find that SAC without online learning performs particularly poorly and finetuning SAC with a behavioral cloning loss only helps if the dynamics are not changed. All methods are evaluated over 10 seeds.

	Method	Avg ± Std error	Success / 10	Median
Tabletop	MEDAL	176.2k ± 15.9k	3	200.0k
	Progressive Networks	159.0k ± 21.2k	3	200.0k

Table 3: We evaluate the performance of a state-of-the-art reset-free algorithm, MEDAL Sharma et al. [2022], and a continual learning approach, Progressive Networks Rusu et al. [2016], in the SLRL setting in the Tabletop domain. We find that both methods are not geared towards completing the desired task as quickly as possible.

to demonstrate how the SLRL setting differs from typical autonomous RL and continual learning settings. Both methods perform poorly in SLRL setting, which is unsurprising because SLRL has the goal of completing the desired task a single time as quickly as possible, which is different from the other two settings. Thus, typical reset-free and continual learning algorithms may not be effective in the single-life setting.

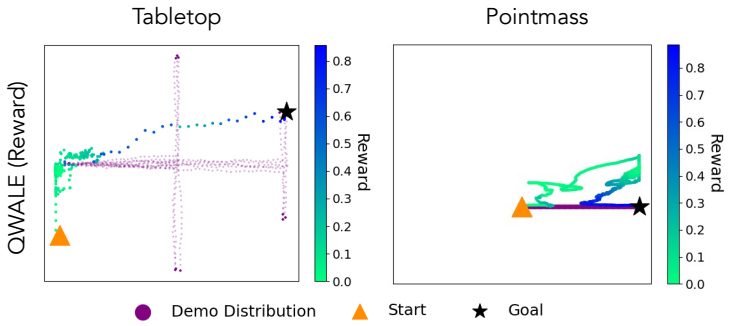

Figure 5: We visualize the online state visitation of the mug (w/ mirrored x-coordinate) in the Tabletop (left) and the agent in the Pointmass (right) environments of a single-life trial with QWALE. The expert demo states are colored in purple, and we color the trajectories green to blue according to reward. The weighted discriminator allots higher reward to states closer to the goal.

Finally, to supplement fig:visitation, we visualize the online state visitation of the mug in the Tabletop and the agent in the Pointmass environments of a single-life trial with QWALE in fig:visitation-reward. These plots show how QWALE helps the agent recover from out-of-distribution states, as the reward gradually increases as the agent gets closer to the goal.