SQIL: Imitation Learning
via Regularized Behavioral Cloning

Siddharth Reddy, Anca D. Dragan, Sergey Levine
Department of Electrical Engineering and Computer Science
University of California, Berkeley
{sgr,anca,svlevine}@berkeley.edu

Abstract

Learning to imitate expert behavior given action demonstrations containing high-dimensional, continuous observations and unknown dynamics is a difficult problem in robotic control. Simple approaches based on behavioral cloning (BC) suffer from state distribution shift, while more complex methods that generalize to out-of-distribution states can be difficult to use, since they typically involve adversarial optimization. We propose an alternative that combines the simplicity of BC with the robustness of adversarial imitation learning. The key insight is that under the maximum entropy model of expert behavior, BC corresponds to fitting a soft Q function that maximizes the likelihood of observed actions. This perspective suggests a way to regularize BC so that it generalizes to out-of-distribution states: combine the standard maximum-likelihood objective with a penalty on the soft Bellman error of the soft Q function. We show that this penalty term gives the agent an incentive to take actions that lead it back to demonstrated states when it encounters new states. Experiments show that our method outperforms BC and GAIL on a variety of image-based and low-dimensional environments in Box2D, Atari, and MuJoCo.

1 Introduction

This paper considers the problem of training an agent to imitate an expert policy, given expert action demonstrations and access to the environment. The agent does not get to observe a reward signal or query the expert, and does not know the state transition dynamics.

Standard approaches to this problem based on behavioral cloning (BC) seek to imitate the expert's actions, but do not reason about the consequences of actions [24]. As a result, they suffer from state distribution shift, and fail to generalize to states that are very different from those seen in the demonstrations [28, 29]. Approaches based on inverse reinforcement learning (IRL) deal with this issue by fitting a reward function that represents preferences over trajectories rather than individual actions [23, 35], and using the learned reward function to train the imitation agent through RL [32, 5, 6]. This is the core idea behind generative adversarial imitation learning (GAIL), which implicitly combines IRL and RL using generative adversarial networks (GANs) [15, 9]. GAIL is the state of the art, but tends to require additional reward augmentation and feature engineering when applied to environments with high-dimensional image observations [21], and inherits the difficulty of training GANs [18].

The main idea in this paper is that instead of resorting to adversarial imitation learning to overcome state distribution shift, we can modify BC so that it generalizes to out-of-distribution states while remaining simple and easy to implement. We propose combining the standard maximum-likelihood objective of BC, which encourages the agent to imitate the expert in demonstrated states, with a

regularization term that gives the agent an incentive to take actions that lead it back to demonstrated states when it encounters new states.

Our intuition for the regularizer is that it should incorporate information about the dynamics of the environment into the objective, so that the agent can learn to get back to demonstrated states. We derive such a regularizer from the maximum entropy model of expert behavior. In this model, the logits of the imitation policy can be interpreted as soft Q values, and BC corresponds to maximum-likelihood estimation of the soft Q function given the demonstrations. The problem with BC is that the learned soft Q function may output arbitrary values in states that are out-of-distribution with respect to the demonstrations. To overcome this issue, we regularize the soft Q function by imposing a penalty on the squared soft Bellman error, which is approximated using transitions from rollouts of the imitation policy periodically sampled during training. Since we do not have access to a reward signal, we set all rewards to zero in the samples, which encourages the agent to get back to the demonstrated states. We refer to this algorithm as *regularized behavioral cloning* (RBC).

To better understand the effect of the proposed regularizer, we show that RBC is similar to an off-policy reinforcement learning (RL) algorithm that rewards the agent for reaching demonstrated states and matching demonstrated actions in those states. The off-policy RL algorithm is a variant of soft Q-learning that initializes the agent's experience replay buffer with demonstrations, sets rewards to a positive constant in the stored demonstration experiences, and sets rewards to zero in all additional experiences. We call this method *soft Q imitation learning* (SQIL). The connection between RBC and SQIL enables us to interpret the effect of the regularizer on the agent's incentives. It also makes our method easier to implement, since SQIL only requires a few small changes to existing off-policy RL code.

The main contribution of this paper is SQIL: a simple and general imitation learning algorithm that is effective in MDPs with high-dimensional, continuous observations and unknown dynamics. We run experiments in four image-based environments – Car Racing, Pong, Breakout, and Space Invaders – and three low-dimensional environments – Humanoid, HalfCheetah, and Lunar Lander – from OpenAI Gym [3], Arcade Learning Environment [1], and MuJoCo [31], to compare SQIL to two prior methods: BC and GAIL. We find that SQIL outperforms both prior methods, especially on the image-based tasks. Our experiments illustrate two key benefits of SQIL: (1) that it can overcome the state distribution shift problem of BC without adversarial training or learning a reward function, which makes it easier to use with images, and (2) that it is simple to implement using existing off-policy value-based RL algorithms.

2 Preliminaries

This work builds on the maximum causal entropy (MaxCausalEnt) model of expert behavior [34, 20]. In an infinite-horizon Markov Decision Process (MDP) with a continuous state space $\mathcal{S}$ and discrete action space $\mathcal{A}$,[1] the demonstrator is assumed to follow a policy π that maximizes reward $R(s, a)$. The policy π forms a Boltzmann distribution over actions,

$$\pi(a|s) \triangleq \exp\left(Q(s, a) - V(s)\right), \tag{1}$$

where Q is the soft Q function, and V is the soft value function,

$$V(s) \triangleq \log\left(\sum_{a \in \mathcal{A}} \exp\left(Q(s, a)\right)\right). \tag{2}$$

If s is an absorbing state, we assume $V(s) = r_{\text{abs}}/(1 - \gamma)$, where γ is the discount factor, and $r_{\text{abs}} \in \mathbb{R}$ is a constant hyperparameter that represents the reward for remaining in an absorbing state for one timestep.[2] The soft Q values are a deterministic function of the rewards and dynamics, given by the soft Bellman equation,

$$Q(s, a) \triangleq R(s, a) + \gamma\mathbb{E}[V(s')]. \tag{3}$$

---

[1]Assuming a discrete action space simplifies our analysis. SQIL can be applied to continuous control tasks using existing sampling methods [11, 12], as illustrated in Section 5.3.

[2]Section 4 discusses how the value of r_{abs} is chosen.

3 Imitation Learning in the Maximum Entropy Model

We aim for an imitation learning algorithm that generalizes to new states, without resorting to complex adversarial optimization procedures. We build on BC, which is a simple approach that seeks to imitate the expert's actions using supervised learning. BC does not reason about the consequences of actions, so when the agent makes small mistakes and enters states that are slightly different from those in the demonstrations, the distribution mismatch between the states in the demonstrations and those actually encountered by the agent leads to compounding errors [29]. Our solution is to add a regularization term to BC that enables it to overcome state distribution shift while remaining simple and easy to implement.

We derive the regularization term from the maximum entropy model of expert behavior, resulting in a method that infers the expert's soft Q function by maximizing the likelihood of observed actions and minimizing the squared soft Bellman error. We show that this regularized BC method is similar to an off-policy RL algorithm that rewards the agent for reaching demonstrated states and matching demonstrated actions in those states. This connection enables us to interpret the effect of the regularizer on the agent's incentives, and makes it possible to implement our method by applying a few simple modifications to any off-policy value-based RL algorithm.

3.1 Behavioral Cloning

Under the generative model in Section 2, BC corresponds to maximum-likelihood estimation of the soft Q function given the demonstrations. Let $\tau = (s_0, a_0, s_1, a_1, ..., s_T)$ denote a rollout, where s_T is an absorbing state, and let $\mathcal{D}_{\text{demo}}$ denote the set of demonstration rollouts. We define BC as fitting a parameterized soft Q function $Q_{\boldsymbol{\theta}}$ to minimize the loss,

$$
\begin{aligned}
\ell_{\text{BC}}(\boldsymbol{\theta}) &\triangleq \sum_{\tau \in \mathcal{D}_{\text{demo}}} \sum_{t=0}^{T-1} -\log \pi_{\boldsymbol{\theta}}(a_t|s_t) \\
&= \sum_{\tau \in \mathcal{D}_{\text{demo}}} \sum_{t=0}^{T-1} -(Q_{\boldsymbol{\theta}}(s_t, a_t) - V_{\boldsymbol{\theta}}(s_t)),
\end{aligned} \tag{4}
$$

where the equality follows from Equation 1, and $V_{\boldsymbol{\theta}}$ denotes the soft value function given by $Q_{\boldsymbol{\theta}}$ and Equation 2. The experiments in Section 5 use a convolutional neural network or multi-layer perceptron to model $Q_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta}$ are the weights of the neural network.

3.2 Regularized Behavioral Cloning

The issue with BC is that when the agent encounters states that are out-of-distribution with respect to $\mathcal{D}_{\text{demo}}$, $Q_{\boldsymbol{\theta}}$ may output arbitrary values. One solution is to add a regularization term to the BC objective that encourages $Q_{\boldsymbol{\theta}}$ to output reasonable default values when it encounters such states. For example, a penalty on the L2-norm of the soft Q values would encourage the soft Q values in out-of-distribution states to be zero instead of arbitrary, leading to a uniform random policy in those states. Instead of inducing a uniform policy, we would like to endow the agent with the ability to get back to demonstrated states when it encounters new states. To do so, we need to incorporate information about the dynamics of the environment into the regularization term.

The model in Section 2 suggests a natural choice of regularizer that achieves this goal: a penalty on the squared soft Bellman error, i.e., the squared difference between the LHS and RHS of Equation 3. Since we do not know the expert's reward function $R(s, a)$, we set all rewards to zero. Additionally, since the state space cannot be enumerated and the dynamics are unknown, we approximate the penalty by evaluating it on transitions (s, a, s') observed in the demonstrations $\mathcal{D}_{\text{demo}}$ as well as additional rollouts $\mathcal{D}_{\text{samp}}$ periodically sampled during training using the imitation policy. Sampling ensures that the penalty covers the state distribution actually encountered by the agent, instead of only the demonstrations.

Formally, we define the regularized BC objective as follows.

$$
\ell_{\text{RBC}}(\boldsymbol{\theta}) \triangleq \ell_{\text{BC}}(\boldsymbol{\theta}) + \lambda B_{\boldsymbol{\theta}}(\mathcal{D}_{\text{demo}} \cup \mathcal{D}_{\text{samp}}, 0), \tag{5}
$$

Algorithm 1 Soft Q Imitation Learning (SQIL)

1: Require $r_{\text{demo}}, \lambda_{\text{samp}} \in \mathbb{R}_{\geq 0}$
2: Initialize $\mathcal{D}_{\text{samp}} \leftarrow \emptyset$
3: **for** $i = 1, 2, ...$ **do**
4: $\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}}(B_{\boldsymbol{\theta}}(\mathcal{D}_{\text{demo}}, r_{\text{demo}}) + \lambda_{\text{samp}} B_{\boldsymbol{\theta}}(\mathcal{D}_{\text{samp}}, 0))$ {See Equation 6}
5: $\quad$ **if** $i \mod k \equiv 0$ **then**
6: $\qquad$ Sample rollout $\tau = (s_0, a_0, s_1, a_1, ..., s_T)$ with imitation policy $\pi_{\boldsymbol{\theta}}(a_t|s_t)$ {See Equation 1}
7: $\qquad \mathcal{D}_{\text{samp}} \leftarrow \mathcal{D}_{\text{samp}} \cup \{\tau\}$
8: $\quad$ **end if**
9: **end for**

---

where $\lambda \in \mathbb{R}_{\geq 0}$ is a constant hyperparameter, and $B_{\boldsymbol{\theta}}$ denotes the sum of squared soft Bellman errors,

$$B_{\boldsymbol{\theta}}(\mathcal{D}, r) \triangleq \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T-1} (Q_{\boldsymbol{\theta}}(s_t, a_t) - (r + \gamma V_{\boldsymbol{\theta}}(s_{t+1})))^2. \tag{6}$$

The BC objective encourages $Q_{\boldsymbol{\theta}}$ to output high values for demonstrated actions at demonstrated states, and the penalty term propagates those high values to nearby states. In other words, $Q_{\boldsymbol{\theta}}$ outputs high values for actions that lead to states from which the demonstrated states are reachable, so when the agent finds itself far from the demonstrated states, it takes actions that lead it back to the demonstrated states.

3.3 Connection to Off-Policy Reinforcement Learning

The squared soft Bellman error term in Equation 5 strongly resembles a soft Q-learning objective [11], hinting at an alternative interpretation of RBC. This section shows that the RBC objective is similar to that of a soft Q-learning algorithm that gives the agent a constant positive reward for reaching a demonstrated state and matching the demonstrated action in that state, and zero reward otherwise.

Consider the following modification to the RBC objective.

$$\ell_{\text{SQIL}}(\boldsymbol{\theta}) \triangleq \ell_{\text{RBC}}(\boldsymbol{\theta}) - \sum_{\tau \in \mathcal{D}_{\text{demo}}} V_{\boldsymbol{\theta}}(s_0) \tag{7}$$

The additional $V_{\boldsymbol{\theta}}(s_0)$ terms encourage the learned soft Q values to be higher and make the imitation policy less stochastic, which can improve performance and reduce the number of hyperparameters that need to be tuned (see the ablation experiments in Section 5.4). More importantly, they lead to the following result (derived in Section A.1 of the appendix).

$$\nabla_{\boldsymbol{\theta}} \ell_{\text{SQIL}}(\boldsymbol{\theta}) \propto \nabla (B_{\boldsymbol{\theta}}(\mathcal{D}_{\text{demo}}, r_{\text{demo}}) + \lambda_{\text{samp}} B_{\boldsymbol{\theta}}(\mathcal{D}_{\text{samp}}, 0)), \tag{8}$$

where $r_{\text{demo}}, \lambda_{\text{samp}} \in \mathbb{R}_{\geq 0}$ are constant hyperparameters. Equation 8 is the gradient of a soft Q-learning algorithm that gives the agent a constant reward of r_{demo} for taking the demonstrated action in a demonstrated state, assigns a reward of zero to all non-demonstration experiences, and balances the number of demonstration experiences and non-demonstration experiences sampled for each step of stochastic gradient descent. We call this algorithm *soft Q imitation learning* (SQIL).

SQIL vs. RBC. The main benefit of using SQIL to optimize Equation 7 instead of using RBC to optimize Equation 5 is that SQIL is trivial to implement, since it only requires a few small changes to existing deep Q-learning code (see Section 4). Extending SQIL to MDPs with a continuous action space is also easy, since we can simply replace soft Q-learning with the soft actor-critic algorithm [12] (see Section 5.3). Given the difficulty of implementing deep RL algorithms correctly [13], this flexibility makes SQIL more practical to use, since it can be built on top of existing code bases. Furthermore, the ablation study in Section 5.4 suggests that SQIL actually performs better than RBC.

4 Soft Q Imitation Learning

SQIL is summarized in Algorithm 1. It performs soft Q-learning with three small, but important, modifications: (1) it initially fills the agent's experience replay buffer with demonstrations, where the rewards are set to some positive constant (e.g., $r_{\text{demo}} \triangleq 1$), (2) as the agent interacts with the world and accumulates new experiences, it adds them to the replay buffer, and sets the rewards for these

additional experiences to zero, and (3) it balances the number of demonstration experiences and new experiences in each sample from the replay buffer. Section A.3 in the appendix contains additional implementation details.

Crucially, since the agent can learn from off-policy data, *the agent does not necessarily have to visit the demonstrated states* in order to experience positive rewards. Instead, the agent replays the demonstrations that were initially added to its experience replay buffer. Thus, SQIL can be used in stochastic environments with continuous states, where the demonstration states may never actually be encountered by the agent.

Termination condition. As the imitation policy learns to behave more like the expert, a growing number of expert-like transitions get added to $\mathcal{D}_{\text{samp}}$ with an assigned reward of zero. This causes the effective reward for mimicking the expert to decay over time. Balancing the number of demonstration experiences and new experiences sampled from the replay buffer ensures that this effective reward remains at least $r_{\text{demo}}/(1 + \lambda_{\text{samp}})$ instead of decaying to zero. In practice, we find that this reward decay does not degrade performance if SQIL is halted once the squared soft Bellman error objective converges to a minimum (e.g., see Figure 8 in the appendix).

Rewards at absorbing states. The value of hyperparameter r_{abs}, which controls the reward at absorbing states (see Section 2), affects the agent's incentives. Setting r_{abs} to a constant much larger than r_{demo} would encourage the agent to terminate the episode quickly, which may be appropriate for certain tasks. Setting r_{abs} much lower than r_{demo} would encourage the agent to avoid terminating the episode. See [17] for further discussion. We set $r_{\text{abs}} \triangleq 0$ in all our experiments, which include some environments where terminating the episode is always undesirable (e.g., walking without falling down) and other environments where success requires terminating the episode (e.g., landing at a target), suggesting that SQIL is not sensitive to the choice of r_{abs}.

5   Experimental Evaluation

Our experiments aim to compare SQIL to existing imitation learning methods on a variety of tasks with high-dimensional, continuous observations, such as images, and unknown dynamics. To that end, we benchmark SQIL against BC and GAIL on four image-based games – Car Racing, Pong, Breakout, and Space Invaders – and three low-dimensional tasks – Humanoid, HalfCheetah, and Lunar Lander. We also investigate which components of SQIL contribute most to its performance via an ablation study on the Lunar Lander game. Section A.3 in the appendix contains additional experimental details.

For all the image-based tasks, we implement a version of GAIL that uses deep Q-learning (GAIL-DQL) instead of TRPO as in the original GAIL paper [15], since Q-learning performs better than TRPO in these environments, and because this allows for a head-to-head comparison of SQIL and GAIL: both algorithms use the same underlying RL algorithm, but provide the agent with different rewards – SQIL provides constant rewards, while GAIL provides learned rewards. We use the standard GAIL-TRPO method as a baseline for all the low-dimensional tasks, since TRPO performs better than Q-learning in these environments.

The original GAIL method implicitly encodes prior knowledge – namely, that terminating an episode is either always desirable or always undesirable. As pointed out in [17], this makes comparisons to alternative methods unfair. We implement the unbiased version of GAIL proposed by [17], and use this in all of the experiments. Comparisons to the biased version with implicit termination knowledge are included in Section A.2 in the appendix.

5.1   Testing Generalization in Image-Based Car Racing

The goal of this experiment is to study not only how well each method can mimic the expert demonstrations, but also how well they can acquire policies that generalize to new states that are not seen in the demonstrations. To do so, we train the imitation agents in an environment with a different initial state distribution $\mathcal{S}_0^{\text{train}}$ than that of the expert demonstrations $\mathcal{S}_0^{\text{demo}}$, allowing us to systematically control the mismatch between the distribution of states in the demonstrations and the states actually encountered by the agent. We run experiments on the Car Racing game from the Box2D environments in OpenAI Gym (screenshot in Figure 1). To create $\mathcal{S}_0^{\text{train}}$, the car is rotated 90 degrees so that it begins perpendicular to the track, instead of parallel to the track as in $\mathcal{S}_0^{\text{demo}}$. This

	Domain Shift ($\mathcal{S}_0^{\text{train}}$)	No Shift ($\mathcal{S}_0^{\text{demo}}$)
Random	-21 ± 56	-68 ± 4
BC (P'91)	-45 ± 18	$\mathbf{698 \pm 10}$
GAIL-DQL	-97 ± 3	-66 ± 8
SQIL (Ours)	$\mathbf{375 \pm 19}$	$\mathbf{704 \pm 6}$
Expert	480 ± 11	704 ± 79

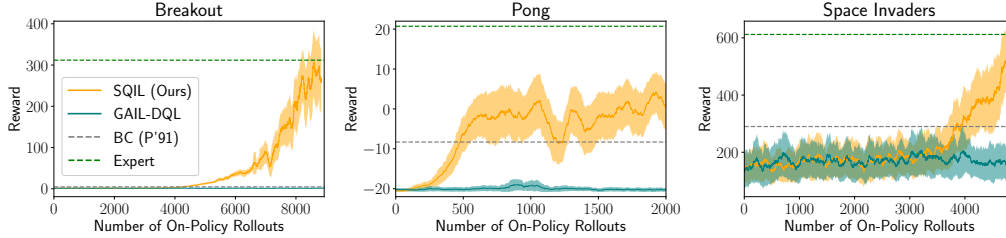Figure 1: Image-based Car Racing. Average reward on 100 episodes after training. Standard error on three random seeds.

Figure 2: Image-based Atari. Smoothed with a rolling window of 100 episodes. Standard error on three random seeds. X-axis represents amount of interaction with the environment (not expert demonstrations).

intervention presents a significant generalization challenge to the imitation learner, since the expert demonstrations do not contain any examples of states where the car is perpendicular to the road, or even significantly off the road axis. The agent must learn to make a tight turn to get back on the road, then stabilize its orientation so that it is parallel to the road, and only then proceed forward to mimic the expert demonstrations.

The results in Figure 1 show that SQIL and BC perform equally well when there is no variation in the initial state. The task is easy enough that even BC achieves a high reward. SQIL performs much better than BC when starting from $\mathcal{S}_0^{\text{train}}$, showing that SQIL is capable of generalizing to a new initial state distribution, while BC is not. SQIL learns to make a tight turn that takes the car through the grass and back onto the road, then stabilizes the car's orientation so that it is parallel to the track, and then proceeds forward like the expert does in the demonstrations. BC tends to drive straight ahead into the grass instead of turning back onto the road.

SQIL outperforms GAIL in both conditions. Since SQIL and GAIL both use deep Q-learning for RL in this experiment, the gap between them may be attributed to the difference in the reward functions they use to train the agent. SQIL benefits from providing a constant reward that does not require fitting a discriminator, while GAIL struggles to train a discriminator to provide learned rewards directly from images.

5.2 Image-Based Experiments on Atari

The results in Figure 2 show that SQIL outperforms BC on Pong, Breakout, and Space Invaders – additional evidence that BC suffers from compounding errors, while SQIL does not. SQIL also outperforms GAIL on all three games, illustrating the difficulty of using GAIL to train an image-based discriminator, as in Section 5.1.

5.3 Instantiating SQIL for Continuous Control in Low-Dimensional MuJoCo

The experiments in the previous sections evaluate SQIL in MDPs with a discrete action space. This section illustrates how SQIL can be adapted to continuous actions. We instantiate SQIL using soft actor-critic (SAC) – an off-policy RL algorithm that can solve continuous control tasks [12]. In particular, SAC is modified in the following ways: (1) the agent's experience replay buffer is initially filled with expert demonstrations, where rewards are set to a positive constant, (2) when taking gradient steps to fit the agent's soft Q function, a balanced number of demonstration experiences and new experiences are sampled from the replay buffer, and (3) the agent observes rewards of zero during
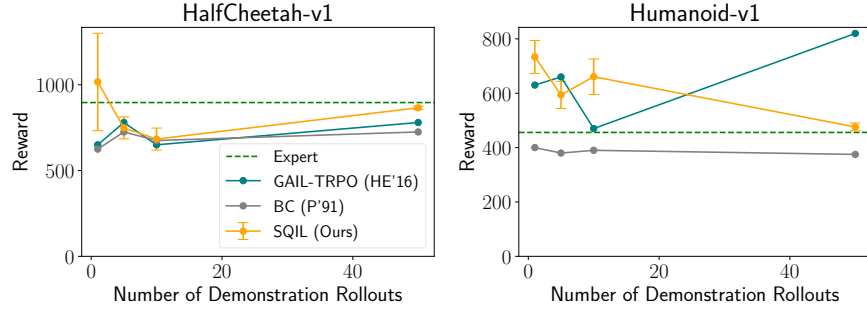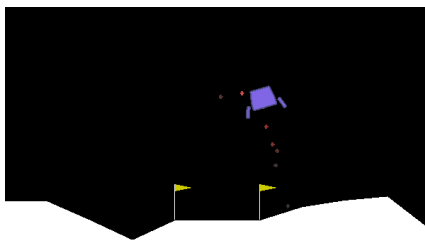
6

Figure 3: Low-dimensional MuJoCo. SQIL: best performance on 10 consecutive training episodes. BC, GAIL: results from [4].

		Domain Shift ($\mathcal{S}_0^{\text{train}}$)	No Shift ($\mathcal{S}_0^{\text{demo}}$)
	Random	0.10 ± 0.30	0.04 ± 0.02
	BC (P'91)	0.07 ± 0.03	$\mathbf{0.93 \pm 0.03}$
	GAIL-TRPO	0.67 ± 0.04	$\mathbf{0.93 \pm 0.03}$
	SQIL (Ours)	$\mathbf{0.89 \pm 0.02}$	$\mathbf{0.88 \pm 0.03}$
Ablation	$\lambda_{\text{samp}} = 0$	0.12 ± 0.02	$\mathbf{0.87 \pm 0.02}$
	$\gamma = 0$	0.41 ± 0.02	$\mathbf{0.84 \pm 0.02}$
	$\pi = \text{Unif}$	0.47 ± 0.02	0.82 ± 0.02
	RBC	0.66 ± 0.02	$\mathbf{0.89 \pm 0.01}$
	Expert	0.93 ± 0.03	0.89 ± 0.31

Figure 4: Low-dimensional Lunar Lander. Best success rate on 100 consecutive episodes during training. Standard error on five random seeds. Performance bolded if at least within one standard error of expert.

its interactions with the environment, instead of an extrinsic reward signal that specifies the desired task. This instantiation of SQIL is compared to GAIL on the Humanoid (17 DoF) and HalfCheetah (6 DoF) tasks from MuJoCo.

The results in Figure 3 show that SQIL outperforms BC and performs comparably to GAIL on both tasks, demonstrating that SQIL can be successfully deployed on problems with continuous actions, and that SQIL can perform well even with a small number of demonstrations. This experiment also illustrates how SQIL can be run on top of SAC or any other off-policy value-based RL algorithm.

5.4 Ablation Study on Low-Dimensional Lunar Lander

We hypothesize that SQIL works well because it combines information about the expert's policy from demonstrations with information about the environment dynamics from rollouts of the imitation policy periodically sampled during training. We also expect RBC to perform comparably to SQIL, since their objectives are similar. To test these hypotheses, we conduct an ablation study using the Lunar Lander game from the Box2D environments in OpenAI Gym (screenshot in Figure 4). As in Section 5.1, we control the mismatch between the distribution of states in the demonstrations and the states encountered by the agent by manipulating the initial state distribution. To create $\mathcal{S}_0^{\text{train}}$, the agent is placed in a starting position never visited in the demonstrations.

In the first variant of SQIL, λ_{samp} is set to zero, to prevent SQIL from using additional samples drawn from the environment (see line 4 of Algorithm 1). This comparison tests if SQIL really needs to interact with the environment, or if it can rely solely on the demonstrations. In the second condition, γ is set to zero to prevent SQIL from accessing information about state transitions (see Equation 6 and line 4 of Algorithm 1). This comparison tests if SQIL is actually extracting information about the dynamics from the samples, or if it can perform just as well with a naïve regularizer (setting γ to zero effectively imposes a penalty on the L2-norm of the soft Q values instead of the squared soft Bellman error). In the third condition, a uniform random policy is used to sample additional rollouts, instead of the imitation policy π_θ (see line 6 of Algorithm 1). This comparison tests how important it is that the samples cover the states encountered by the agent during training. In the fourth condition, we use

RBC to optimize Equation 5 instead of using SQIL to optimize Equation 7. This comparison tests the effect of the additional $-V_{\boldsymbol{\theta}}(s_0)$ terms in Equation 7 vs. Equation 5.

The results in Figure 4 show that all methods perform well when there is no variation in the initial state. When the initial state is varied, SQIL performs significantly better than BC, GAIL, and the ablated variants of SQIL. This confirms our hypothesis that SQIL needs to sample from the environment using the imitation policy, and relies on information about the dynamics encoded in the samples.

Surprisingly, SQIL outperforms RBC by a large margin, suggesting that the additional $-V_{\boldsymbol{\theta}}(s_0)$ terms in Equation 7 do in fact improve performance by encouraging the learned soft Q values to be higher and making the imitation policy less stochastic (discussed in Section 3.3). We expect that with additional tuning of the temperature hyperparameter in RBC, we could achieve the same effect as the additional $-V_{\boldsymbol{\theta}}(s_0)$ terms, and RBC would perform the same as SQIL.

6 Related Work

Various approaches have been developed to address state distribution shift in BC, without relying on IRL or adversarial optimization. Hand-engineering a domain-specific loss function and carefully designing the demonstration collection process have enabled researchers to train effective imitation policies for self-driving cars [2], autonomous drones [8], and robotic manipulators [33, 26, 25]. DAgger-based methods query the expert for on-policy action labels [29, 19]. These approaches either require domain knowledge or the ability to query the expert, while SQIL requires neither.

Concurrently with SQIL, another imitation learning algorithm that uses off-policy RL with constant rewards instead of a learned reward function was developed [30]. We see our paper as contributing additional evidence to support this core idea, rather than proposing a competing method. First, SQIL is derived as an extension of BC, while the prior method is derived from an alternative formulation of the IRL objective, showing that two different theoretical approaches independently lead to using off-policy RL with constant rewards as an alternative to adversarial training – a sign that this idea may be a promising direction for future work. Second, SQIL is shown to outperform BC and GAIL in domains that were not evaluated in [30] – in particular, tasks with image observations and significant shift in the state distribution between the demonstrations and the training environment. This suggests that the results of the low-dimensional MuJoCo experiments in [30], which show the prior method outperforms BC and GAIL, may extend to more complex tasks.

SQIL resembles the Deep Q-learning from Demonstrations (DQfD) [14] and Normalized Actor-Critic (NAC) algorithms [7], in that all three algorithms fill the agent's experience replay buffer with demonstrations and include an imitation loss in the agent's objective. The key difference between SQIL and these prior methods is that DQfD and NAC are RL algorithms that assume access to a reward signal, while SQIL is an imitation learning algorithm that does not require an extrinsic reward signal from the environment. Instead, SQIL automatically constructs a reward signal from the demonstrations.

The SQIL objective is similar to that of the inverse soft Q-learning (ISQL) algorithm [27]. Their details and motivations are, however, significantly different. ISQL is an internal dynamics estimation algorithm, while SQIL is for imitation learning. ISQL also assumes that the demonstrations include observations of the expert's reward signal, while SQIL does not.

7 Conclusions and Future Work

We contribute the SQIL algorithm: a general method for learning to imitate an expert given action demonstrations and access to the environment. Simulation experiments on tasks with high-dimensional, continuous observations and unknown dynamics show that our method outperforms both BC and GAIL, while being simple to implement on top of existing off-policy RL code.

SQIL might be used to recover not just the expert's policy, but also their reward function; for example, by using a parameterized reward function to model rewards in the soft Bellman error terms, instead of using constant rewards. This could provide a simpler alternative to existing adversarial IRL algorithms [6].

8 Acknowledgements

References

[1] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

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

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

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

[5] 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, 2016.

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

[7] Yang Gao, Ji Lin, Fisher Yu, Sergey Levine, Trevor Darrell, et al. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

[8] Alessandro Giusti, Jérôme Guzzi, Dan C Ciresan, Fang-Lin He, Juan P Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, 2016.

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

[10] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *arXiv preprint arXiv:1809.01999*, 2018.

[11] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

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

[13] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

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

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

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] 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. In *International Conference on Learning Representations*, 2019.

[18] Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. The gan landscape: Losses, architectures, regularization, and normalization. *arXiv preprint arXiv:1807.04720*, 2018.

[19] Michael Laskey, Sam Staszak, Wesley Yu-Shu Hsieh, Jeffrey Mahler, Florian T Pokorny, Anca D Dragan, and Ken Goldberg. Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 462–469. IEEE, 2016.

[20] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[21] Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In *Advances in Neural Information Processing Systems*, pages 3812–3822, 2017.

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

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

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

[25] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Aman Behal, and Ladislau Bölöni. Learning real manipulation tasks from virtual demonstrations using lstm. *arXiv preprint*, 2016.

[26] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multitask manipulation for inexpensive robots using end-to-end learning from demonstration. *arXiv preprint arXiv:1707.02920*, 2017.

[27] Siddharth Reddy, Anca D Dragan, and Sergey Levine. Where do you think you're going?: Inferring beliefs about dynamics from behavior. *arXiv preprint arXiv:1805.08010*, 2018.

[28] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668, 2010.

[29] 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, 2011.

[30] Fumihiro Sasaki, Tetsuya Yohira, and Atsuo Kawaguchi. Sample efficient imitation learning for continuous control. In *International Conference on Learning Representations*, 2019.

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

[32] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

[33] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. *arXiv preprint arXiv:1710.04615*, 2017.

[34] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Modeling interaction via the principle of maximum causal entropy. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 1255–1262. Omnipress, 2010.

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

A   Appendix

A.1   Derivation of SQIL Gradient

Splitting up the sum of squared soft Bellman error terms for $\mathcal{D}_{\mathrm{demo}}$ and $\mathcal{D}_{\mathrm{samp}}$,

$$\nabla_{\boldsymbol{\theta}}\ell_{\mathrm{SQIL}}(\boldsymbol{\theta}) = \sum_{\tau\in\mathcal{D}_{\mathrm{demo}}}\left(-\nabla V_{\boldsymbol{\theta}}(s_0) + \sum_{t=0}^{T-1} -(\nabla Q_{\boldsymbol{\theta}}(s_t,a_t) - \nabla V_{\boldsymbol{\theta}}(s_t))\right)$$

$$+ \lambda_{\mathrm{demo}}\sum_{\tau\in\mathcal{D}_{\mathrm{demo}}}\sum_{t=0}^{T-1}\nabla(Q_{\boldsymbol{\theta}}(s_t,a_t) - \gamma V_{\boldsymbol{\theta}}(s_{t+1}))^2 + \lambda_{\mathrm{samp}}\nabla B_{\boldsymbol{\theta}}(\mathcal{D}_{\mathrm{samp}},0)$$

$$= \sum_{\tau\in\mathcal{D}_{\mathrm{demo}}}\left(-\nabla V_{\boldsymbol{\theta}}(s_0) + \sum_{t=0}^{T-1}\nabla V_{\boldsymbol{\theta}}(s_t) - \gamma\nabla V_{\boldsymbol{\theta}}(s_{t+1})\right)$$

$$+ \lambda_{\mathrm{demo}}\nabla B_{\boldsymbol{\theta}}\left(\mathcal{D}_{\mathrm{demo}},\frac{1}{2\lambda_{\mathrm{demo}}}\right) + \lambda_{\mathrm{samp}}\nabla B_{\boldsymbol{\theta}}(\mathcal{D}_{\mathrm{samp}},0). \tag{9}$$

Setting $\gamma \triangleq 1$ turns the inner sum in the first term into a telescoping sum.

$$(9) = \sum_{\tau\in\mathcal{D}_{\mathrm{demo}}}\nabla V_{\boldsymbol{\theta}}(s_T) + \lambda_{\mathrm{demo}}\nabla B_{\boldsymbol{\theta}}\left(\mathcal{D}_{\mathrm{demo}},\frac{1}{2\lambda_{\mathrm{demo}}}\right) + \lambda_{\mathrm{samp}}\nabla B_{\boldsymbol{\theta}}(\mathcal{D}_{\mathrm{samp}},0). \tag{10}$$

Since s_T is assumed to be absorbing (see Section 3.1), $V_{\boldsymbol{\theta}}(s_T)$ is a constant (see Section 2). Thus,

$$(10) = \lambda_{\mathrm{demo}}\nabla B_{\boldsymbol{\theta}}\left(\mathcal{D}_{\mathrm{demo}},\frac{1}{2\lambda_{\mathrm{demo}}}\right) + \lambda_{\mathrm{samp}}\nabla B_{\boldsymbol{\theta}}(\mathcal{D}_{\mathrm{samp}},0)$$

$$\propto \nabla\left(B_{\boldsymbol{\theta}}(\mathcal{D}_{\mathrm{demo}},r_{\mathrm{demo}}) + \lambda_{\mathrm{samp}}B_{\boldsymbol{\theta}}(\mathcal{D}_{\mathrm{samp}},0)\right), \tag{11}$$

where $r_{\mathrm{demo}}, \lambda_{\mathrm{samp}} \in \mathbb{R}_{\geq 0}$ are constant hyperparameters.

A.2   Comparing the Biased and Unbiased Variants of GAIL

As discussed in Section 5, to correct the original GAIL method's biased handling of rewards at absorbing states, we implement the suggested changes to GAIL in Section 4.2 of [17]: adding a transition to an absorbing state and a self-loop at the absorbing state to the end of each rollout sampled from the environment, and adding a binary feature to the observations indicating whether or not a state is absorbing. We refer to the original, biased GAIL method as GAIL-DQL-B and GAIL-TRPO-B, and the unbiased version as GAIL-DQL-U and GAIL-TRPO-U.

Car Racing. The results in Figure 5 show that both the biased (GAIL-DQL-B) and unbiased (GAIL-DQL-U) versions of GAIL perform equally poorly. The problem of training an image-based discriminator for this task may be difficult enough that even with an unfair bias toward avoiding crashes that terminate the episode, GAIL-DQL-B does not perform better than GAIL-DQL-U.

Atari. The results in Figure 6 show that SQIL outperforms both variants of GAIL on Pong and the unbiased version of GAIL (GAIL-DQL-U) on Breakout and Space Invaders, but performs comparably to the biased version of GAIL (GAIL-DQL-B) on Space Invaders and worse than it on Breakout. This may be due to the fact that in Breakout and Space Invaders, the agent has multiple lives – five in Breakout, and three in Space Invaders – and receives a termination signal that the episode has ended after losing each life. Thus, the agent experiences many more episode terminations than in Pong, exacerbating the bias in the way the original GAIL method handles rewards at absorbing states. Our implementation of GAIL-DQL-B in this experiment provides a learned reward of $r(s,a) = -\log(1 - D(s,a))$, where D is the discriminator (see Section A.3 in the appendix for details). The learned reward is always positive, while the implicit reward at an absorbing state is zero. Thus, the agent is inadvertently encouraged to avoid terminating the episode. For Breakout and Space Invaders, this just happens to be the right incentive, since the objective is to stay alive as long as possible. GAIL-DQL-B outperforms SQIL in Breakout and performs comparably to SQIL in Space Invaders because GAIL-DQL-B is accidentally biased in the right way.

Lunar Lander. The results in Figure 7 show that when the initial state is varied, SQIL outperforms the unbiased variant of GAIL (GAIL-TRPO-U), but underperforms against the biased version of

	Domain Shift ($\mathcal{S}_0^{\text{train}}$)	No Shift ($\mathcal{S}_0^{\text{demo}}$)
Random	-21 ± 56	-68 ± 4
BC (P'91)	-45 ± 18	$\mathbf{698 \pm 10}$
GAIL-DQL-B	-91 ± 4	-34 ± 21
GAIL-DQL-U	-97 ± 3	-66 ± 8
SQIL (Ours)	$\mathbf{375 \pm 19}$	$\mathbf{704 \pm 6}$
Expert	480 ± 11	704 ± 79

Figure 5: Image-based Car Racing. Average reward on 100 episodes after training. Standard error on three random seeds.
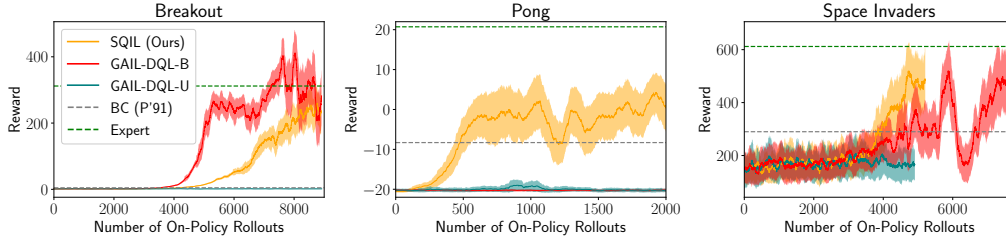

Figure 6: Image-based Atari. Smoothed with a rolling window of 100 episodes. Standard error on three random seeds. X-axis represents amount of interaction with the environment (not expert demonstrations).

GAIL (GAIL-TRPO-B). The latter result is likely due to the fact that the implementation of GAIL-TRPO-B we used in this experiment provides a learned reward of $r(s, a) = \log(D(s, a))$, where D is the discriminator (see Section A.3 in the appendix for details). The learned reward is always negative, while the implicit reward at an absorbing state is zero. Thus, the agent is inadvertently encouraged to terminate the episode quickly. For the Lunar Lander game, this just happens to be the right incentive, since the objective is to land on the ground and thereby terminate the episode. As in the Atari experiments, GAIL-TRPO-B performs better than SQIL in this experiment because GAIL-TRPO-B is accidentally biased in the right way.

A.3 Implementation Details

To ensure fair comparisons, the same network architectures were used to evaluate SQIL, GAIL, and BC. For Lunar Lander, we used a network architecture with two fully-connected layers containing 128 hidden units each to represent the Q network in SQIL, the policy and discriminator networks in GAIL, and the policy network in BC. For Car Racing, we used four convolutional layers (following [10]) and two fully-connected layers containing 256 hidden units each. For Humanoid and HalfCheetah, we used two fully-connected layers containing 256 hidden units each. For Atari, we used the convolutional neural network described in [22] to represent the Q network in SQIL, as well as the Q network and discriminator network in GAIL.

To ensure fair comparisons, the same demonstration data were used to train SQIL, GAIL, and BC. For Lunar Lander, we collected 100 demonstration rollouts. For Car Racing, Pong, Breakout, and Space Invaders, we collected 20 demonstration rollouts. Expert demonstrations were generated from scratch for Lunar Lander using DQN [22], and collected from open-source pre-trained policies for Car Racing [10] as well as Humanoid and HalfCheetah [4]. The Humanoid demonstrations were generated by a stochastic expert policy, while the HalfCheetah demonstrations were generated by a deterministic expert policy; both experts were trained using TRPO.[3] We used two open-source implementations of GAIL: [6] for Lunar Lander, and [4] for MuJoCo. We adapted the OpenAI Baselines implementation of GAIL to use soft Q-learning for Car Racing and Atari. Expert demonstrations were generated from scratch for Atari using DQN.

---

[3]`https://drive.google.com/drive/folders/1h3H4AY_ZBx08hz-Ct0Nxxus-V1melu1U`

		Domain Shift ($\mathcal{S}_0^{\text{train}}$)	No Shift ($\mathcal{S}_0^{\text{demo}}$)
	Random	0.10 ± 0.30	0.04 ± 0.02
	BC (P'91)	0.07 ± 0.03	$\mathbf{0.93 \pm 0.03}$
	GAIL-TRPO-B (HE'16)	$\mathbf{0.98 \pm 0.01}$	$\mathbf{0.95 \pm 0.02}$
	GAIL-TRPO-U	0.67 ± 0.04	$\mathbf{0.93 \pm 0.03}$
	SQIL (Ours)	$\mathbf{0.89 \pm 0.02}$	$\mathbf{0.88 \pm 0.03}$
Ablation	$\lambda_{\text{samp}} = 0$	0.12 ± 0.02	$\mathbf{0.87 \pm 0.02}$
	$\gamma = 0$	0.41 ± 0.02	$\mathbf{0.84 \pm 0.02}$
	$\pi = \text{Unif}$	0.47 ± 0.02	0.82 ± 0.02
	RBC	0.66 ± 0.02	$\mathbf{0.89 \pm 0.01}$
	Expert	0.93 ± 0.03	0.89 ± 0.31

Figure 7: Low-dimensional Lunar Lander. Best success rate on 100 consecutive episodes during training. Standard error on five random seeds. Performance bolded if at least within one standard error of expert.
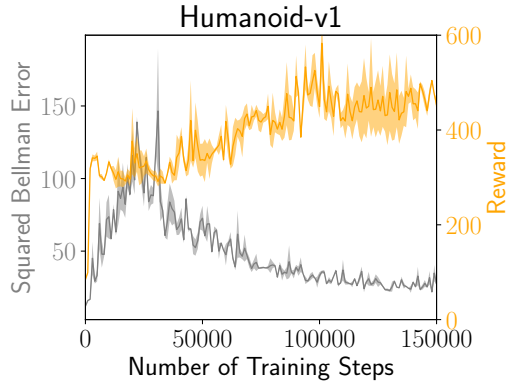

Figure 8: Standard error over two random seeds. No smoothing across training steps.

For Lunar Lander, we set $r_{\text{demo}} = 1$, $k = 200$, and $\lambda_{\text{samp}} = 10^{-6}$. For Car Racing, we set $k = 50$, and $\lambda_{\text{samp}} = 0.01$. For Humanoid, we set $r_{\text{demo}} = 5$ and $\lambda_{\text{samp}} = 1$. For HalfCheetah, we set $r_{\text{demo}} = 10$ and $\lambda_{\text{samp}} = 1$. For Atari, we set $r_{\text{demo}} = 1$, $k = 1$, and $\lambda_{\text{samp}} = 1$.

SQIL was not pre-trained in any of the experiments. GAIL was pre-trained using BC for HalfCheetah, but was not pre-trained in any other experiments.

In standard implementations of soft Q-learning and SAC, the agent's experience replay buffer typically has a fixed size, and once the buffer is full, old experiences are deleted to make room for new experiences. In SQIL, we never delete demonstration experiences from the replay buffer, but otherwise follow the standard implementation.

We use Adam [16] to take the gradient step in line 4 of Algorithm 1.

The BC and GAIL performance metrics in Section 5.3 are taken from [4].[4]

The GAIL and SQIL policies in Section 5.3 are set to be deterministic during the evaluation rollouts used to measure performance.