EvIL: Evolution Strategies for Generalisable Imitation Learning

Silvia Sapora [1] Gokul Swamy [2] Chris Lu [1] Yee Whye Teh [1] Jakob Nicolaus Foerster [1]

Abstract

Often times in imitation learning (IL), the environment we collect expert demonstrations in and the environment we want to deploy our learned policy in aren't exactly the same (e.g. demonstrations collected in simulation but deployment in the real world). Compared to policy-centric approaches to IL like behavioural cloning, reward-centric approaches like *inverse reinforcement learning* (IRL) often better replicate expert behaviour in new environments. This transfer is usually performed by optimising the recovered reward under the dynamics of the target environment. However, *(a)* we find that modern deep IL algorithms frequently recover rewards which induce policies far weaker than the expert, *even in the same environment the demonstrations were collected in*. Furthermore, *(b)* these rewards are often quite poorly shaped, necessitating extensive environment interaction to optimise effectively. We provide simple and scalable fixes to both of these concerns. For *(a)*, we find that *reward model ensembles* combined with a slightly different training objective significantly improves re-training and transfer performance. For *(b)*, we propose a novel *evolution-strategies* based method (`EvIL`) to optimise for a reward-shaping term that speeds up re-training in the target environment, closing a gap left open by the classical theory of IRL. On a suite of continuous control tasks, we are able to re-train policies in target (and source) environments more interaction-efficiently than prior work.

1. Introduction

Consider the problem of predicting driver behaviour across a network of roads. Let's say there are $|\mathcal{S}|$ nodes in our routing graph and $|\mathcal{A}|$ roads going out of each node. If we simply wanted to predict the path usually taken between some start node s_1 and some goal node s_2, we could collect data of drivers navigating between these two points and learn a policy by regressing from their states to their actions, an approach known as behavioural cloning (BC, Pomerleau (1988)). As we'd need to learn an action at each state, this would require learning $|\mathcal{S}||\mathcal{A}|$ parameters. Now, let's say we wanted to learn the path taken between the same start node s_1 and some new goal node s_3. We'd need to repeat this entire procedure once again. Thus, to learn how to navigate from one goal to all destinations, a policy-centric approach like BC would need $|\mathcal{S}|^2|\mathcal{A}|$ parameters. In contrast, a reward-centric approach to imitation like *inverse reinforcement learning* (IRL, Ziebart et al. (2008a)) would simply require learning the $|\mathcal{S}||\mathcal{A}|$ parameters of the underlying reward function from expert data and then could be optimised to find the shortest path between any two nodes in the graph. This ability to learn a *compact* representation that generalises well across multiple tasks / environments was one of the key reasons IRL was developed in the first place. Indeed, as Ng et al. (2000) put it, *"the entire field of reinforcement learning is founded on the presupposition that the reward function, rather than the policy is the most succinct, robust, and transferable definition of the task"*. This intuition remains true in the modern day, perhaps explaining why robust real-world mapping services like Google Maps are build on top of the bedrock of IRL (Barnes et al., 2023).

If we look at classical IRL algorithms like Maximum Entropy IRL (Ziebart et al., 2008a) or LEARCH (Ratliff et al., 2009), they usually have a double loop structure. In the outer loop, one trains a discriminator to solve a classification problem between learner and expert trajectories. The discriminator is then used as a reward function in the inner loop, where the learner optimises this adversarially chosen reward function via reinforcement learning. Observe that because we are actually optimising the reward function to completion in each inner loop iteration, we have reason to believe we will be able to retrain effectively from scratch, a point borne out in practice across a wide variety of disciplines (e.g. in robotics (Silver et al., 2010; Ratliff et al., 2009; Kolter et al., 2008; Ng et al., 2006; Zucker et al., 2011), computer vision (Kitani et al., 2012), and human-computer interaction (Ziebart et al., 2008b; 2012)).

More modern approaches to IRL like GAIL (Ho & Ermon,

[1]University of Oxford, UK [2]Carnegie Mellon University, USA. Correspondence to: Silvia Sapora <silvia.sapora@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

2016) keep the double loop structure but perform a small amount of policy optimisation in each inner loop iteration, rather than completely resolving the RL problem. While this means we expend less environment interactions in each inner loop iteration, it also means that we no longer can ensure that retraining from scratch on the recovered reward function will guarantee performance similar to that of the expert. Beyond being merely a theoretical concern, we find that the rewards recovered by modern deep IL algorithms preclude such ***effective*** retraining, *even in the environment the expert demonstrations were collected in.* This raises one our work's key questions: ***how can we preserve the computational benefits of modern IRL while matching the re-training ability of classical IRL?***

However, even the rewards recovered by classical IRL methods suffer from a weakness: there is no need for them to be nicely shaped, which can preclude ***efficient*** retraining. In fact, later in the same paper, Ng et al. (2000) raise the following question: *"shaping rewards can produce reward functions that make it dramatically easier to learn a solution to an MDP, without affecting optimality. Can we design IRL algorithms that recover 'easy' rewards?"* As we discuss in greater detail in the following sections, classical IRL algorithms are entirely agnostic to reward shaping, as they assume we have the ability to perfectly compute the optimal policy at each iteration and use a loss function for updating the reward estimate that is invariant to shaping terms. Thus, the classical theory of IRL doesn't provide any clear answers for how to make re-training more interaction-efficient.

We provide a method for efficient and effective retraining in IRL that scales to modern deep learning architectures and algorithms. First, **we find that a combination of *reward model ensembles*, random policy resets, and a more stable loss function allows for more *effective* retraining** that enables the agent to more closely match expert performance. For the second, **we introduce a novel *evolution-strategies* based method that *directly* optimises for *efficiency* in re-training**, beating out classical value function-based approaches to reward shaping. We refer to the combination of these two techniques as `EvIL`: ***an algorithm that allows effective and efficient re-training in novel environments.***

More explicitly, our contributions are four-fold:

1. We empirically demonstrate that rewards recovered by conventional IRL algorithms consistently fail in producing optimal agents when employed to retrain an agent from scratch. This holds true even when the retraining occurs within the same environment where the data was initially collected, differing from the picture in theory.

2. We provide a suite of adjustments that make retraining based on the reward recovered by IRL more effective. Our adjustments are simple to implemented require minimal additional environment interactions, preserving the efficiency of more modern approaches to IRL.

3. We introduce a novel evolution-strategies based approach to potential-based reward shaping that allows for efficient retraining. Rather than classical heuristic approaches that attempt to uniformly approximate the value function of the optimal policy, we instead use zeroth-order optimisation to *directly* optimise for interaction-efficient retraining. We also show a speedup in vanilla RL.

4. We perform extensive experimental evaluation of our proposed method across a suite of continuous control tasks and find that it leads to significantly more efficient and effective retraining in source and target environments than prior work. We show that we can combine both of these techniques to reap the benefits of efficient and effective retraining even in environments that are markedly different than what the demonstrations were collected in.

We begin with a discussion of related work.

2. Related Work

Inverse Reinforcement Learning. IRL is commonly framed as a two-player zero-sum game between a policy player and a reward function player (Syed & Schapire, 2007; Ho & Ermon, 2016; Swamy et al., 2021). Intuitively, the reward function player tries to pick out differences between the current learner policy and the expert demonstration, while the policy player attempts to maximise this reward function to move closer to expert behaviour. As pointed out by Finn et al. (2016), this setup is effectively a GAN (Goodfellow et al., 2014) in the space of trajectories.

IRL algorithms can be categorised into two flavours: *primal* and *dual* (Swamy et al., 2021). In a primal algorithm, one follows a *no-regret* strategy for both the policy and reward player. Practically, this corresponds to taking a small step on the reward function before performing a small amount of policy optimisation. Most modern deep IL algorithms like GAIL (Ho & Ermon, 2016) and AIRL (Fu et al., 2018) follow this approach. In contrast, the classical approaches to IRL (e.g. MaxEnt IRL (Ziebart et al., 2008a), LEARCH (Ziebart et al., 2008a), Abbeel & Ng (2004)) are of the *dual* flavour: one follows a no-regret strategy for reward selection against a *best response* via RL for policy selection. Practically, this corresponds to optimising the reward to completion at each iteration. For example, in MaxEnt IRL, the best response is performed via soft value iteration. From the perspective of interaction efficiency, primal methods are preferable to dual methods due to lesser interaction in their inner loop. However, from the perspective of ensuring effective retraining (more formally, from the perspective of *best-iterate* convergence), dual methods are preferable due to their explicit retraining at each iteration. Our goal is to achieve the best of both worlds: we seek to preserve the retraining ability of classical dual methods while leveraging the interaction efficiency of modern primal methods.

Efficient Inverse RL. Various authors have attempted to improve IRL, both in terms of *sample efficiency* (number of expert demonstrations required) and *interaction efficiency*. Swamy et al. (2022) provide the minimax-optimal algorithm for IRL in terms of sample efficiency – this is complimentary to the interaction efficiency we focus on in our work. Swamy et al. (2023) provide the first general $\mathsf{poly}(H)$ interaction complexity algorithm for IRL. In contrast, we focus on efficient *retraining*. Furthermore, we can easily combine our approach with theirs as they are orthogonal – they focus on reducing exploration during policy search under an arbitrary reward function while we provide a method to learn well-shaped reward functions that can be used by an arbitrary downstream policy optimisation procedure. The same can be said for the *hybrid RL* based approach of Ren et al. (2023) and the BC regularisation suggested by Tiapkin et al. (2023).

Reward Shaping. Starting with the seminal work of Ng et al. (1999), reward shaping has emerged as a critical component of reducing the interaction complexity of reinforcement learning methods (Jaderberg et al., 2019; Wu & Tian, 2017). As we discuss further below, reward shaping is often thought of as reducing the *effective horizon* of the planning problem. In the search-based planning literature, this is often referred to as $A^\star$ *search* (Likhachev et al., 2003; 2005). Even in the era of deep RL, the effective horizon of a problem has remained as an accurate predictor of the performance of policy optimisation algorithms (Laidlaw et al., 2023). From this perspective, the optimal shaping term would be the value function of the optimal policy – the greedy policy would then be the optimal policy. However, ensuring that we've learned a value function that closely approximates the true optimal value function everywhere in the state space (more formally, an *admissible* heuristic (Russell & Norvig, 2010)), is a rather tall order, outside of small tabular problems like those considered in Cooke et al. (2023). We find that in practice, using the critic of a strong policy as a learned shaping term improves interaction efficiency, but that our approach is able to significantly out-perform this baseline. We hypothesise this is because a learned critic is likely only accurate on states the optimal policy visits, while our method directly optimises for reducing training time by looking for a shaping term that provides signal over the entire course of training, including on the state distribution of the initial weak policy.

Evolution Strategies for RL. Recent work leverages JAX-based (Bradbury et al., 2018) hardware acceleration for RL to massively parallelise training (Lu et al., 2022a). This has been used for large-scale multi-agent learning (Rutherford et al., 2023) and rapid population-based training (Flajolet et al., 2022). Another, more related, line of work leverages these techniques to perform evolution-based (Salimans et al., 2017) bi-level optimisation. For example, Lu et al. (2022a); Jackson et al. (2023) *evolve* surrogate objective functions to effectively discover new, performant reinforcement learning algorithms. Lu et al. (2023); Lupu et al. (2024) instead evolve and analyse adversarial environment features and data that influence long-term RL learning behaviour whilst Khan et al. (2023); Lu et al. (2022b) do the same for adversarial policies. Thus, Evolution Strategies (ES) are well-suited for long-horizon and noisy bi-level optimisation tasks, such as those involving an RL inner loop. We apply a similar evolution-based approach for reward shaping.

3. Background

We provide a quick overview of relevant background.

3.1. Inverse Reinforcement Learning as Game Solving

We consider a finite-horizon Markov Decision Process (MDP) (Puterman, 2014) parameterized by $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, H \rangle$ where $\mathcal{S}$, $\mathcal{A}$ are the state and action spaces, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition operator, and H is the horizon. In imitation learning, we see trajectories generated by an expert policy $\pi_E : \mathcal{S} \to \Delta(\mathcal{A})$, but do not know the reward function. Our goal is to find a reward function that, when optimised, recovers a policy with similar behaviour to that of the expert. We cast this problem as a zero-sum game between a policy player and an adversary that tries to penalise any difference between the expert and learner policies (Syed & Schapire, 2007; Ho & Ermon, 2016; Swamy et al., 2021). More formally, we optimise over policies $\pi : \mathcal{S} \to \Delta(\mathcal{A}) \in \Pi$ and reward functions $f : \mathcal{S} \times \mathcal{A} \to [-1, 1] \in \mathcal{F}_r$, where we assume that both Π and $\mathcal{F}_r$ are convex and compact so Sion's minimax theorem holds. We use $\xi = (s_1, a_1, r_1, \dots, s_H, a_H, r_H)$ to denote the trajectory generated by some policy. Using $J(\pi, \hat{r}) = \mathbb{E}_{\xi \sim \pi}[\sum_{h=1}^{H} \hat{r}(s_h, a_h)]$ to denote the value of policy π under reward function $\hat{r}$, we have the following:

$$\min_{\pi \in \Pi} \max_{f \in \mathcal{F}_r} J(\pi_E, f) - J(\pi, f). \tag{1}$$

3.2. Potential-Based Reward Shaping

Potential-based reward shaping (Ng et al., 1999) is a technique for speeding up policy optimisation by reducing the effective planning horizon of the problem which guarantees preserving policy optimality. More formally, we define a potential function $\Phi : \mathcal{S} \to \mathbb{R}$ that assigns real values to states $s \in \mathcal{S}$ in the environment. The potential-based shaping term $F : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ is then defined as the *change* in the potential over the course of a transition. i.e.

$$F_\Phi(s, s') = \Phi(s') - \Phi(s). \tag{2}$$

This shaping term is added to the standard reward during training: if in the original MDP M the reward is $r(s,a)$ for transitioning from s to s' with action $a \sim \pi(s)$, then in the new MDP M' the reward is updated to be

$$r'(s,a) = r(s,a) + F(s,s') \tag{3}$$

at all but the last timestep, for which $r'(s_H, a_H) = r(s_H, a_H) + F(s_H, s_1)$. Observe that for any policy $\pi \in \Pi$, $J(\pi, r) = J(\pi, r')$ as the $F(s, s')$ term telescopes. Thus, $\pi^\star = \arg\max_{\pi\in\Pi} J(\pi, r') = \arg\max_{\pi\in\Pi} J(\pi, r)$ – we preserve *policy optimality*. Thus, in theory, PBRS only affects the speed at which we converge to the optimal policy, rather than the value of the policy we end up converging to.

3.3. Evolution Strategies

Evolution Strategies (ES) are zeroth-order, population-based, stochastic optimisation algorithms. First, they use random noise to generate a population of candidate solutions. These solutions are then evaluated using a fitness function. Lastly, the population is iteratively improved over time by assigning higher weight to better-performing population members. This causes the population to move closer and closer to the optimal solution, and the process is repeated until a satisfactory solution is found. Recently, ES has been successfully applied to a variety of tasks (Real et al., 2019; Salimans et al., 2017; Such et al., 2018). ES algorithms are gradient-free and well-suited for (meta-)optimisation problems where the objective function is noisy or non-differentiable and the search space is large or complex (Beyer, 2000; Lange, 2023; Lu et al., 2023; 2022a; Houthooft et al., 2018). This includes reward function shaping (Niekum et al., 2010) and RL hyperparameter search (Elfwing et al., 2018).

There are several types of ES algorithms, one of the most well known is the covariance matrix adaptation evolution strategy (CMA-ES) (Hansen & Ostermeier, 2001), which represents the population by a full-covariance multivariate Gaussian. Although CMA-ES can be applied to our problem, it has only proven successful in low to medium dimension optimisation spaces. Another widely applied ES algorithm is OpenAI-ES (Salimans et al., 2017) which estimates the gradient through the following function:

$$\nabla_\theta \mathbb{E}_{\epsilon\sim N(0,1)} F(\theta + \sigma\epsilon) = \frac{1}{\sigma}\mathbb{E}_{\epsilon\sim N(0,1)}\{F(\theta+\sigma\epsilon)\epsilon\}$$

This is an unbiased estimate and, in contrast to meta-gradient approaches, ES avoids the need to backpropagate the gradient through the whole training procedure, which often results in biased gradients due to truncation (Werbos, 1990; Metz et al., 2022; Liu et al., 2022). As CMA-ES struggles with higher dimensional problems, we use OpenAI-ES for optimizing shaping terms across all of our experiments.

4. Reward-Centric Challenges of Efficient IRL

We attempt to solve the IRL Game (Eq. 1) via a more interaction-efficient *primal* strategy: taking small steps on the reward function via gradient descent and on the policy via RL, as in Ho & Ermon (2016). As we described in the related work, in contrast to *dual* algorithms that repeatedly retrain the policy at each step, this means we have weaker guarantees as far as the performance of the policy trained on the recovered reward. This is more than just a theoretical concern: as we will demonstrate in the following sections, standard GAIL-like algorithms often recover rewards that, when optimised from scratch, do not lead to strong policies. Unfortunately, switching back to a dual strategy might require an infeasible amount of interaction for the scale of problems more modern primal methods seek to solve. We therefore are left with our first open question to ponder:

Challenge 1: *How do we ensure the final reward function returned by primal IRL methods permits effective (even if not efficient) retraining from scratch?*

Assuming that we successfully solve the **Challenge 1**, a key next question is that of *interaction-efficient learning*. Observe that because Eq. 1 can be written as a difference of performances, it is *invariant* to shaping potential-based terms. Thus, for the wide variety of algorithms that are essentially solving this game (e.g. MaxEnt IRL (Ziebart et al., 2008a) or GAIL (Ho & Ermon, 2016) – see Swamy et al. (2021) for a more complete list), we have no way to ensure that they are picking well-shaped rewards, and therefore have no way to encourage the discriminator to pick rewards that are suitable for re-training. This issue is well known even in the classical IRL literature (Ng et al., 2000). This leads to the next key question we seek to answer:

Challenge 2: *How should we modify the discriminator learning process to ensure that the overall IRL procedure returns well-shaped rewards for efficient retraining?*

In theory at least, given a reward function r, there is a clean example of a shaping term that permits efficient training. Consider setting $\Phi(s) = V^\star(s)$: the value function of the optimal policy $\pi^\star$ and assume deterministic dynamics. Then, by the definition of an *advantage function*, we have that

$$r'(s,a) = r(s,a) + V^\star(s') - V^\star(s) = A^\star(s,a).$$

The greedy policy with respect to this modified reward (i.e. the optimal advantage function $A^\star$) is π^*. Thus, we only need a planning horizon of 1 to compute the optimal policy. At heart, this is why potential-based reward shaping speeds up RL: it reduces the amount of planning the agent has to perform. In general, one has to pay exponentially in the horizon of the problem for RL (Kakade, 2003), so a perfectly-shaped reward function can provide an *exponential* speedup in terms of overall interaction efficiency.

In the language of search-based planning, $V^\star(s)$ can be thought of as an optimal *heuristic*. Unfortunately, without access to $\pi^\star$, it is rather difficult to compute $V^\star$ for all but small tabular problems (e.g. via value iteration). Even with access to $\pi^\star$, estimating a $\hat{V}^\star$ that is accurate *uniformly* over the state space is rather challenging, as we'd need to roll out $\pi^\star$ from a wide variety of states, including those entirely outside of its induced state visitation distribution. In practice, one can often at best hope to have access to the critic used in the training of $\pi^\star$, which is likely only accurate on states visited by $\pi^\star$ and not the states visited early on in training by weak policies, where shaping is most important. Thus, in practice, we are left with an open question.

Challenge 3: *In practice, how do we learn a potential-based shaping term that is useful throughout the course of training from scratch?*

We now discuss concrete and scalable solutions to each of these challenges before validating their empirical efficacy.

5. `EvIL`: Evolution Strategies for Generalisable Imitation Learning

We now provide solutions to the preceding challenges.

5.1. Solution 1: Improving Retrainability in IRL

In order to improve retrainability of the rewards recovered by primal IRL algorithms, we propose a trifecta of strategies that can be applied across a variety of base algorithms.

1A: Policy Buffer. To mitigate the risk of the IRL discriminator forgetting valuable signals it previously provided during an earlier policy update, we maintain an ongoing buffer containing all past policy trajectories. This continuous retention allows for the consistent retraining of the IRL discriminator on the full history of learner policies. This differs from the more common practice of taking a small gradient step on the classification loss between the expert and most recent learner policy. In a sense, this can be thought of as moving from procedure reminiscent of Online Gradient Descent (Zinkevich, 2003) to one reminiscent of Follow the Regularised Leader (McMahan, 2011). While both are no-regret algorithms in theory, prior work in imitation learning has found that the latter can sometimes produce more stable results (Ross et al., 2011).

1B: Discriminator and Policy Ensembles. To address potential errors where the discriminator might assign an unusually high value to a state not visited by the policy, we adopt an ensemble-based approach. The policy reward is calculated as the average of all the discriminators. While ensembling techniques are common in RL for approximations of pessimism (Kidambi et al., 2020), we instead use them to enlarge the portion of the state space where our discriminator provides useful feedback, which can be important during the random exploration that is usually a critical part of the start of RL training. Accordingly, to ensure that each discriminator is trained on a sufficiently different set of states, we also train an ensemble of policies, using data from one policy per discriminator.

1C: Random Policy Resets. During IRL training, it's possible for the inner learner policy to converge prematurely, limiting exploration of all relevant states. This might cause the discriminator to overfit to a specific learner state distribution, which might differ from the totality of states seen during a fresh re-training. To elide this concern, we occasionally re-initialise the learner policy during training, with linearly decreasing probability as training advances. This still allows the learner to match expert performance while enhancing retrainability. In a sense, this can be thought of as annealing from dual to primal IRL over the course of a single training run, allowing us to reap the benefits inherent to both families of approaches.

Combined, all these modifications (along with other slight changes like improved regularisation on the discriminator) significantly improve retraining performance under the recovered reward. We call the resulting algorithm IRL++. We also performed an ablation of these components and other techniques we found to provide limited benefits – see Figure 6 in the appendix for more details.

5.2. Solution 2: Decoupling Shaping from Discrimination

We propose using a two-stage procedure to improve retraining efficiency for IRL: first, learning a reward, and second, learning a shaping term to be added in during the re-training procedure. This allows us to avoid the issue that sequence of loss functions that the discriminator sees during game-solving are invariant to how well shaped the chosen reward function. Critically, because shaping terms do not change the set of optimal policies (Ng et al., 1999), optimising the recovered reward plus the shaping term cannot affect the correctness of the overall procedure and therefore preserves the strong performance guarantees of IRL.

A bit more explicitly, we propose first running IRL++ to recover a reward function that admits effective (but not efficient) retraining. Second, we optimise for a shaping term that maximises the area under the curve of the reward recovered by IRL++ (as we have no access to the ground truth reward function). This gets directly at the objective we care about – retraining interaction efficiency – without going through the detour of trying to learn a value function network that is accurate on the set of states encountered during retraining. The area under the curve of an RL training curve as a function of a shaping term added to the reward function is clearly a complex and non-differentiable objective. We

Algorithm 1 Reward Shaping with Evolution Strategies

```
Input: Reward function $\hat{r}$, population size $N$, inner loop updates $M$, learning rate $\alpha$, noise standard deviation $\sigma$
Initialise potential parameters $\theta$
// Outer-loop, optimise shaping parameters
repeat
    Generate Gaussian noise $\epsilon_1, ...\epsilon_N \sim \mathcal{N}(0, I)$ to generate $N$ members in the population
    for $i = 1, \dots, N$ do
        $\Phi_{\theta_i} = \Phi_\theta + \sigma\epsilon_i$
        $AUC_i = 0$
        Initialise policy $\pi_{i0}$
        // Inner-loop, tracking RL training efficiency
        for $j = 1, \dots, M$ do
            $\pi_{ij} \leftarrow$ RL Step on reward $\hat{r}' = F_{\Phi_{\theta_i}} + \hat{r}$
            $AUC_i \leftarrow AUC_i + \mathbb{E}_{\xi \sim \pi_{ij}}[\sum_{h=0}^{H-1} \hat{r}'(s_h, a_h, s_{h+1})]$
        end for
    end for
    // Estimate gradient and update meta (shaping) parameters
    $\mathcal{L}_i \leftarrow -AUC_i$
    $\theta \leftarrow \theta - \alpha \frac{1}{N\sigma} \sum_{i=1}^{N} \mathcal{L}_i \epsilon_i$
until convergence
Output: Final shaping function $\Phi_\theta$
```

therefore need an shaping term optimisation procedure that is amenable to such circumstances, which leads to the last component of our overall proposed procedure.

5.3. Solution 3: ES for Potential-Based Shaping

Rather than attempting to learn $V^\star$, we propose simply *evolving* a potential Φ that leads to faster training. Specifically, we use the *area under the curve* (AUC) of the performance $J(\pi, \hat{r})$ vs. environment interactions plot as the *fitness function* for the ES optimisation. As shown in Algorithm 1, we calculate reward AUC during training by saving the policy's performance after each gradient update. After the inner loop procedure is complete, the AUC is passed onto the OpenES algorithm. This estimates the gradient to maximise AUC and updates the shaping meta parameters accordingly. We note that such a technique is of interest in learning shaping terms even for vanilla RL with known rewards and confirm its efficacy at doing so in Figure 1.

We refer to the combination of IRL++ with an evolved shaping term as EvIL: **EVolution strategies for Imitation Learning**. See Algs. 1 and 2 for full details of our method.

5.3.1. When is EvIL a good idea?

A natural question when reading the preceding section might be the following: *if we care about improving the sample efficiency of IRL retraining, doesn't expending a large interaction budget by repeatedly retraining the agent from scratch to learn a good shaping term via evolution seem counter-intuitive?* Indeed, if one were to perform such shaping-via-evolution in the real world, EvIL would likely cost more to implement than it would save in the final retraining in the novel (or same) environment.

However, for a variety of domains, we have access to a reasonably accurate simulator. Such simulators are often used to perform the interactive learning component of IRL, with the recovered reward then optimised in the real world to recover strong policies / trajectories (e.g. in the autonomous driving domain (Vinitsky et al., 2022; Gulino et al., 2024)). As discussed above, this recovered reward can be (and often is in practice) poorly shaped. Thus, *by running EvIL inside the simulator, we can trade exploration in the real world for exploration in simulation, which is likely to be cheaper and safer.* Of course, then one has to contend with the well-known sim2real gap, both in terms of the recovered reward and on the learned shaping term. We investigate this concern in depth in our experiments section, where we explore the effect of a shaping term in improving retraining performance in a novel environment (e.g. the real world instead of the simulator the shaping term was trained in) and find encouraging results across the board. Thus, we believe that on problems where a cheap approximate simulator exists, EvIL presents a feasible path for improving the interaction efficiency of retraining on the reward recovered by IRL. Even on problems where this is not the case, one could potentially *learn* a model to perform IRL in, as explored by Ren et al. (2023). It would be an interesting direction for future work to perform ES-based shaping inside a learned model (where interaction is relatively inexpensive) and see how much benefit it provides for real-world retraining.

Algorithm 2 `EvIL`: Evolution Strategies for Generalisable Imitation

```
Input: Expert trajectories $\mathcal{D}_E$, Learning rate $\alpha$, Ensemble size $K$,
// Step #1:  Run IRL++
Initialise policy $\pi_0$, reward functions $f_0^{1:K}$, and buffer $\mathcal{D}_0$.
for $i = 1, ..., N$ do
    $\mathcal{D}_i \leftarrow \mathcal{D}_{i-1} \cup \{\xi_i \sim \pi_i\}$
    $\ell_i(f) = \mathbb{E}_{\xi \sim \mathcal{D}_i}\left[\sum_h^H f(s_h, a_h)\right] - \mathbb{E}_{\xi \sim \mathcal{D}_E}\left[\sum_h^H f(s_h, a_h)\right] + ||f||_2$
    $\forall k \in [K]$, $f_{i+1}^k \leftarrow f_i^k - \alpha \nabla_f \ell_i$
    Take small RL step on $\frac{-1}{K}\sum_k^K f_i^k$ to get $\pi_{i+1}$.
end for
// Step #2:  Shape IRL reward
Run Algorithm 1 on $\hat{r} = \frac{-1}{K}\sum_k^K f_N^k$ to evolve shaping term $\Phi_\theta$.
Output: Shaped reward $F_{\Phi_\theta} + \hat{r}$.
```

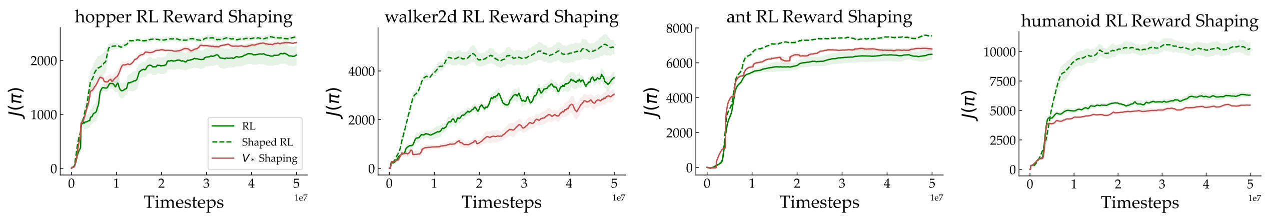

Figure 1. **Shaping RL.** Our method successfully recovers a potential-based reward function that helps an RL agent learn faster. We compare against the baseline of using the expert value function as a shaping term when retraining and non-shaped RL training on the real reward. We compute standard error across 5 seeds. $J(\pi)$ is the performance of the learner under the ground truth reward.

6. Experimental Results

In this section, we aim to answer the following questions:

1. **Can ES (Algorithm 1) be used to learn a shaping function that improves interaction efficiency?** We first investigate whether ES successfully learns an effective shaping function in RL with a hand-designed reward. We compare to a strong baseline: the critic of a policy trained extensively on the ground-truth reward.

2. **Does our modified IRL procedure ensure we recover reward functions that admit effective retraining?** We investigate whether, by taking the final reward function returned by our modified procedure and optimising it extensively in the training environment, we are able to find a policy with similar performance to the expert.

3. **By combining the preceding techniques (i.e. `EvIL`), are we able to efficiently and effectively retrain policies in both the source and target environments?** We investigate whether we are able to more sample-efficiently find high quality policies, both in the source and in target environments. Specifically, we consider environments where we add stochasticity to the dynamics or slightly change the link lengths of our agent.

Together these questions get at our over-arching goal with `EvIL`: *the ability to robustly transfer expert behaviour to novel environments with limited interaction.*

We conduct our experiments across three distinct MuJoCo environments: Hopper, Walker, and Ant. All learners receive 100 trajectories from the expert policy, trained using Proximal Policy Optimisation (PPO) (Schulman et al., 2017) over 5e7 timesteps. We perform two sets of transfer experiments per environment. In the first, with probability $p_{tremble}$, a randomly selected action is executed instead of the one determined by the agent's policy, potentially leading the agent to encounter previously unseen states at test time. In the second, more challenging set of experiments, we randomly sample link lengths from a distribution with significant support outside of the dynamics the expert demonstrations were collected under.

Our IRL implementation is built upon the GAIL framework proposed in Ho & Ermon (2016). To enhance its performance, we follow the recommendations of Swamy et al. (2022), such as incorporating a gradient penalty (Gulrajani et al., 2017) to stabilise the discriminator training process and a linear learning rate decay for the actor, critic, and discriminator. Within the inner loop of our optimisation strategy, we employ PPO (Schulman et al., 2017). Across all experiments, our IRL discriminator is trained on states only, rather than state-action pairs. This reflects an under-

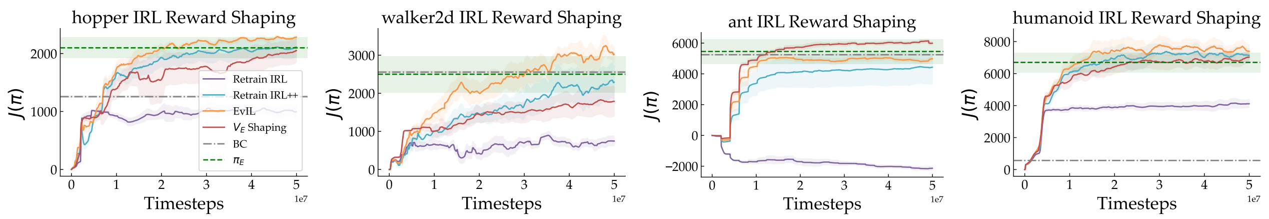

Figure 2. **Shaping IRL.** We show our method can successfully recover a potential-based reward function that makes the recovered reward function easier to learn. We use the shaping term combined with a reward recovered from IRL++. We compare against three baselines: the reward recovered from an ensemble of discriminators, without shaping, the reward recovered by a classic IRL method, and the IRL++ reward shaped using the expert value function when retraining. For each, we train on 5 seeds, with shading representing standard error.

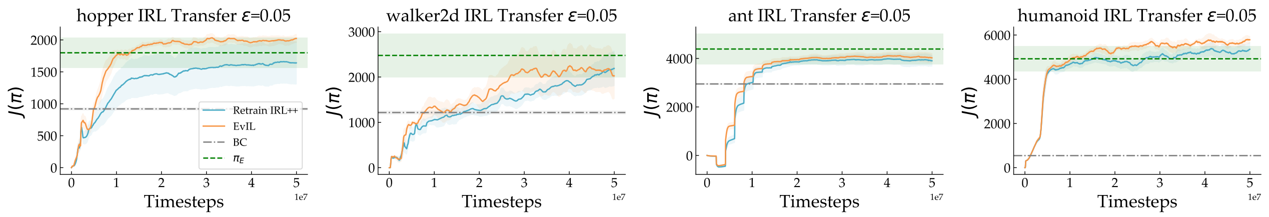

Figure 3. **`EvIL` Transfer on Trembling Hand Environment** `EvIL` outperforms both BC and IRL++ on transfer to an environment where, with ϵ probability, a random action is executed in the environment rather than the one the agent selected. IRL++ out-performs BC, highlighting the importance of interactive training for effective transfer.

appreciated benefit of IRL: the ability to perform imitation without access to action labels, which are difficult to obtain in various domains (e.g. learning from third-person videos).

All our code is implemented in JAX (Bradbury et al., 2018) using the PureJaxRL (Lu et al., 2022a), Brax (Freeman et al., 2021), and evosax (Lange, 2023) libraries to maximise parallelisation of training. For all our experiments, we use 5 parallel learners and 5 discriminators.

6.1. ES-Based Shaping of Ground-Truth Rewards

As a sanity check, we first confirm that performing ES-based shaping recovers $V^\star$ in a small tabular in Figure 7 in the appendix. Next, in Figure 1, we compare the performance of an RL agent trained with our evolved shaping function to the performance of an agent trained on the ground truth reward. We see that we are consistently able to recover a shaping that leads to better interaction efficiency, even on top of a hand-designed reward function. Furthermore, we find that we consistently compete with or exceed the performance of using the critic of a strong policy as a learned shaping term. Recall that this is a very strong baseline that assumes access to a privileged piece of information. We attribute the strength of our method to directly optimising for the desired objective, allowing one to ensure one does not waste representational capacity on accurately fitting $V^\star$ on states that are unimportant for efficient re-training.

In greater detail, we evolve the shaping term over 300 generations with a population size of 64. To expedite the optimisation procedure, we only shaped the first 50% of RL training for all environments. We note that none of our training runs had converged after 300 generations, indicating we could have kept training to further improve the efficacy of our learned shaping term, indicating the results we report are a lower bound on the maximal efficacy of our method.

6.2. IRL++: Retraining with the Recovered Reward

In Figure 2, we show that naively re-training on the final IRL reward function performs poorly. Specifically, in the Ant environment, employing this reward function results in a negative reward, while in the Hopper, Walker and Humanoid environments, the policy plateaus, ceasing to learn at a sub-optimal performance. These policies might be stuck in local minimum and never converge, regardless of how many environment interactions they get to experience. In contrast, after employing the suggestions from the preceding section (i.e. IRL++), we find that retraining from scratch (in teal) gives us significantly better performance with enough training time. This confirms that IRL++ permits effective re-training. However, in environments like Walker and Ant, training is perhaps less efficient than desirable, which segues into the results for our complete method.

6.3. `EvIL`: Retraining with an Evolved Shaping Term

For `EvIL`, we combine our proposed reward model ensembling and policy resets (i.e. IRL++) with an evolved shaping term. The shaping rewards were evolved over 300 genera-

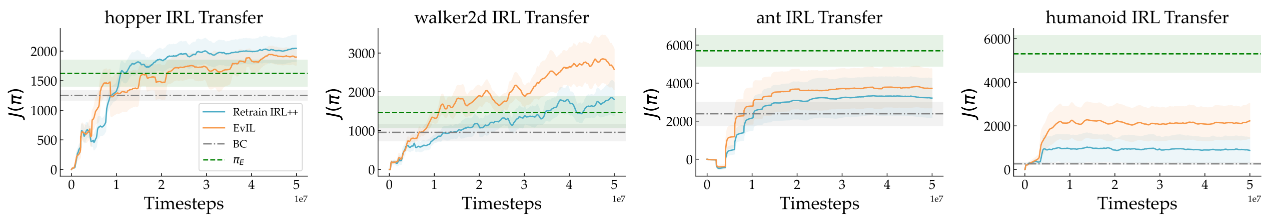

Figure 4. **`EvIL` Transfer on Randomised Dynamics Environment** `EvIL` outperforms both BC and IRL++ on transfer to an environment where link lengths and joint ranges are randomly sampled and differ from the demonstrations. As before, IRL++ out-performs BC.

tions with a population size of 64. The Hopper and Walker shaping function were optimised on the first 20% of training only, while Ant and Humanoid doubled the amount of training considered for the last 100 generations. In Figure 2, we see that across the board, `EvIL` (in orange) leads to marked improvements over optimising the unshaped reward. For both Hopper and Humanoid, we are able to match / exceed expert performance while for Ant we are able to achieve a score thousands higher than the IRL baseline. The V_E shaping baseline we consider is particularly strong due to the privileged information it has access to: an estimate expert's value function (i.e. the final critic from the RL training we used to generate the expert). Interestingly, we find that the performance of V_E shaping is inconsistent: it hinders training in the Hopper, Walker and Humanoid environments, but is effective in the Ant environment. We hypothesise this is due to differing levels of covariate shift in terms of the state distribution of the initial and expert policies.

6.4. `EvIL` Transfer Performance

To assess the transferability of the policies learned by all methods, we consider two sets of modifications to the environment. First, we consider, with probability $p_{tremble} = 0.05$, forcing the policy to execute a random action, a corruption that is not present in the expert demonstrations. We retrain under the recovered reward in this stochastic environment but do ***not*** give the learner access to the test environment during the inverse RL procedure. Echoing our argument in the introduction of the paper, we find that the interactive methods are able to more effectively transfer than offline approaches like behavioural cloning. Furthermore, we find that our learned shaping term improves both the *effectiveness* (final performance) and *efficiency* (interaction before performance peaks) of the retraining procedure.

To highlight the robustness of our method, we then consider a significantly more challenging transfer task: controlling an agent with different link lengths than the one the expert demonstrations were collected with. Specifically, we introduce random variations in the link lengths and joint ranges of the target agents (details postponed to the Appendix). Performance was evaluated by sampling 10 different agent variations from each MuJoCo environment. Our findings in Figure 4 show that `EvIL` consistently outperforms the BC and unshaped baselines. BC performance exhibits a notable decline across all three environments when compared to both the trembling hand case and the standard environment, highlighting the difficulty of this transfer learning task.

In summary, our experiments support our hypothesis that the combination of ingredients that make up `EvIL` are critical for ensuring that we are able to *effectively* and *efficiently* mimic expert behaviour in novel environments.

7. Conclusion

Summary. We empirically demonstrate that the reward functions returned by modern deep IRL algorithms do not admit effective retraining. We propose a set of fixes, IRL++, that help recover reward functions that, when optimised, lead to policies that match expert performance. However, such retraining can be rather expensive in terms of the number of interactions required. In response, we propose `EvIL`: a full-stack algorithm that combines IRL++ with an evolution strategies-based approach for learning shaping terms. On top of the retraining efficacy IRL++ gives us, `EvIL` gives us efficiency. We validate `EvIL`'s performance on challenging transfer tasks involving unseen environments and find that it is able to out-perform the prior art.

Acknowledgements

SS was supported by Google TPU Research Cloud (TRC) and Google Cloud Research Credits program.

GKS thanks Drew Bagnell for his thoughts on our core algorithm and suggestion of the value function shaping baseline.

JF is partially funded by the UKI grant EP/Y028481/1 (originally selected for funding by the ERC). JF is also supported by the JPMC Research Award and the Amazon Research Award.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

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

Barnes, M., Abueg, M., Lange, O. F., Deeds, M., Trader, J., Molitor, D., Wulfmeier, M., and O'Banion, S. Massively scalable inverse reinforcement learning in google maps. *arXiv preprint arXiv:2305.11290*, 2023.

Beyer, H.-G. Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice. *Computer Methods in Applied Mechanics and Engineering*, 186(2):239–267, 2000. ISSN 0045-7825. doi: https://doi.org/10.1016/S0045-7825(99)00386-2. URL https://www.sciencedirect.com/science/article/pii/S0045782599003862.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Cooke, L. H., Klyne, H., Zhang, E., Laidlaw, C., Tambe, M., and Doshi-Velez, F. Toward computationally efficient inverse reinforcement learning via reward shaping. *arXiv preprint arXiv:2312.09983*, 2023.

Elfwing, S., Uchibe, E., and Doya, K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2017.12.012. URL https://www.sciencedirect.com/science/article/pii/S0893608017302976. Special issue on deep reinforcement learning.

Finn, C., Christiano, P., Abbeel, P., and Levine, S. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models, 2016.

Flajolet, A., Monroc, C. B., Beguir, K., and Pierrot, T. Fast population-based reinforcement learning on a single machine. In *International Conference on Machine Learning*, pp. 6533–6547. PMLR, 2022.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning, 2018.

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

Gulino, C., Fu, J., Luo, W., Tucker, G., Bronstein, E., Lu, Y., Harb, J., Pan, X., Wang, Y., Chen, X., et al. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *Advances in Neural Information Processing Systems*, 36, 2024.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. Improved training of wasserstein gans, 2017.

Hansen, N. and Ostermeier, A. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398.

Ho, J. and Ermon, S. Generative adversarial imitation learning, 2016.

Houthooft, R., Chen, R. Y., Isola, P., Stadie, B. C., Wolski, F., Ho, J., and Abbeel, P. Evolved policy gradients, 2018.

Jackson, M. T., Lu, C., Kirsch, L., Lange, R. T., Whiteson, S., and Foerster, J. N. Discovering temporally-aware reinforcement learning algorithms. In *Second Agent Learning in Open-Endedness Workshop*, 2023.

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castañeda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., Sonnerat, N., Green, T., Deason, L., Leibo, J. Z., Silver, D., Hassabis, D., Kavukcuoglu, K., and Graepel, T. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, May 2019. ISSN 1095-9203. doi: 10.1126/science.aau6249. URL http://dx.doi.org/10.1126/science.aau6249.

Kakade, S. M. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

Khan, A., Willi, T., Kwan, N., Tacchetti, A., Lu, C., Grefenstette, E., Rocktäschel, T., and Foerster, J. Scaling opponent shaping to high dimensional games. *arXiv preprint arXiv:2312.12568*, 2023.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823, 2020.

Kitani, K. M., Ziebart, B. D., Bagnell, J. A., and Hebert, M. Activity forecasting. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part IV 12*, pp. 201–214. Springer, 2012.

Kolter, J. Z., Rodgers, M. P., and Ng, A. Y. A control architecture for quadruped locomotion over rough terrain. In *2008 IEEE International Conference on Robotics and Automation*, pp. 811–818. IEEE, 2008.

Laidlaw, C., Russell, S., and Dragan, A. Bridging rl theory and practice with the effective horizon. *arXiv preprint arXiv:2304.09853*, 2023.

Lange, R. T. evosax: Jax-based evolution strategies. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pp. 659–662, 2023.

Likhachev, M., Gordon, G. J., and Thrun, S. Ara*: Anytime a* with provable bounds on sub-optimality. *Advances in neural information processing systems*, 16, 2003.

Likhachev, M., Stentz, A., and Thrun, S. Anytime dynamic a*: An anytime, replanning algorithm. 2005.

Liu, B., Feng, X., Ren, J., Mai, L., Zhu, R., Zhang, H., Wang, J., and Yang, Y. A theoretical understanding of gradient bias in meta-reinforcement learning, 2022.

Lu, C., Kuba, J., Letcher, A., Metz, L., Schroeder de Witt, C., and Foerster, J. Discovered policy optimisation. *Advances in Neural Information Processing Systems*, 35: 16455–16468, 2022a.

Lu, C., Willi, T., De Witt, C. A. S., and Foerster, J. Model-free opponent shaping. In *International Conference on Machine Learning*, pp. 14398–14411. PMLR, 2022b.

Lu, C., Willi, T., Letcher, A., and Foerster, J. N. Adversarial cheap talk. In *International Conference on Machine Learning*, pp. 22917–22941. PMLR, 2023.

Lupu, A., Lu, C., Liesen, J. L., Lange, R. T., and Foerster, J. N. Behaviour distillation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=qup9xD8mW4.

McMahan, B. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 525–533. JMLR Workshop and Conference Proceedings, 2011.

Metz, L., Freeman, C. D., Schoenholz, S. S., and Kachman, T. Gradients are not all you need, 2022.

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

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

Ng, A. Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., and Liang, E. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental robotics IX*, pp. 363–372. Springer, 2006.

Niekum, S., Barto, A., and Spector, L. Genetic programming for reward function search. *Autonomous Mental Development, IEEE Transactions on*, 2:83 – 90, 07 2010. doi: 10.1109/TAMD.2010.2051436.

Pomerleau, D. A. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Ratliff, N. D., Silver, D., and Bagnell, J. A. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27(1):25–53, 2009.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4780–4789, Jul. 2019. doi: 10.1609/aaai.v33i01.33014780. URL https://ojs.aaai.org/index.php/AAAI/article/view/4405.

Ren, J., Swamy, G., Wu, Z. S., Bagnell, J. A., and Choudhury, S. Hybrid inverse reinforcement learning. 2023. URL https://www.robot-learning.ml/2023/files/paper42.pdf.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Russell, S. J. and Norvig, P. *Artificial intelligence a modern approach*. 2010.

Rutherford, A., Ellis, B., Gallici, M., Cook, J., Lupu, A., Ingvarsson, G., Willi, T., Khan, A., de Witt, C. S., Souly, A., et al. Jaxmarl: Multi-agent rl environments in jax. *arXiv preprint arXiv:2311.10090*, 2023.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning, 2017.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017.

Silver, D., Bagnell, J. A., and Stentz, A. Learning from demonstration for autonomous navigation in complex unstructured terrain. *The International Journal of Robotics Research*, 29(12):1565–1592, 2010.

Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., and Clune, J. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning, 2018.

Swamy, G., Choudhury, S., Bagnell, J. A., and Wu, S. Of moments and matching: A game-theoretic framework for closing the imitation gap. In *International Conference on Machine Learning*, pp. 10022–10032. PMLR, 2021.

Swamy, G., Rajaraman, N., Peng, M., Choudhury, S., Bagnell, J., Wu, S. Z., Jiao, J., and Ramchandran, K. Minimax optimal online imitation learning via replay estimation. *Advances in Neural Information Processing Systems*, 35: 7077–7088, 2022.

Swamy, G., Choudhury, S., Bagnell, J. A., and Wu, Z. S. Inverse reinforcement learning without reinforcement learning, 2023.

Syed, U. and Schapire, R. E. A game-theoretic approach to apprenticeship learning. *Advances in neural information processing systems*, 20, 2007.

Tiapkin, D., Belomestny, D., Calandriello, D., Moulines, E., Naumov, A., Perrault, P., Valko, M., and Menard, P. Regularized rl. *arXiv preprint arXiv:2310.17303*, 2023.

Vinitsky, E., Lichtlé, N., Yang, X., Amos, B., and Foerster, J. Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world. *Advances in Neural Information Processing Systems*, 35: 3962–3974, 2022.

Werbos, P. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. doi: 10.1109/5.58337.

Wu, Y. and Tian, Y. Training agent for first-person shooter game with actor-critic curriculum learning. 2017.

Ziebart, B., Dey, A., and Bagnell, J. A. Probabilistic pointing target prediction via inverse optimal control. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, pp. 1–10, 2012.

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

Ziebart, B. D., Maas, A. L., Dey, A. K., and Bagnell, J. A. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *Proceedings of the 10th international conference on Ubiquitous computing*, pp. 322–331, 2008b.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pp. 928–936, 2003.

Zucker, M., Ratliff, N., Stolle, M., Chestnutt, J., Bagnell, J. A., Atkeson, C. G., and Kuffner, J. Optimization and learning for rough terrain legged locomotion. *The International Journal of Robotics Research*, 30(2):175–191, 2011.

A. Appendix

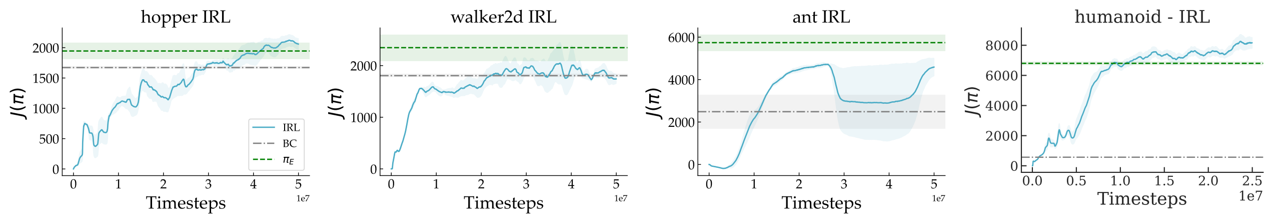

Figure 5. **IRL** IRL training across MuJoCo environments compared to expert and BC performance. 5 seeds each, shading represents standard error.

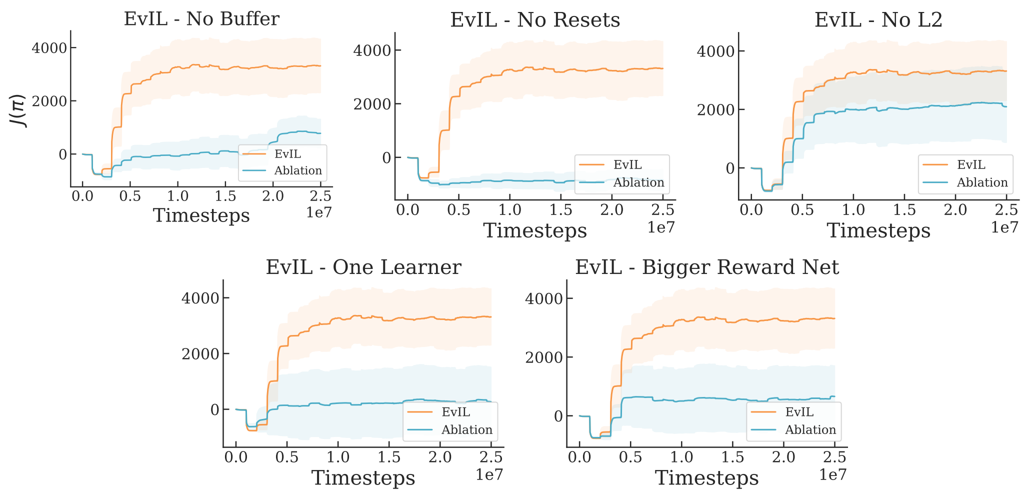

Figure 6. Ablation for `EvIL` on the Ant environment. `EvIL` uses an ensemble of size 5. **a**. We compare `EvIL` with an ensemble that does not utilise a replay buffer. In this case, the discriminator only observes samples from the most recent policy version. **b**. We compare against a version of `EvIL` that doesn't implement resets of the learner policies. **c**. We check the effect of ℓ_2 regularisation on the reward network **d**. The ablation explores the effect of using a single learner seed (and 5 discriminators) instead of multiple seeds. **e**. We use a bigger reward network (two hidden layers of size 512) to verify the issue doesn't lie in our reward network not having enough representational capacity to accurately describe the expert behaviour.

B. Environment Dynamics Randomisation

To vary the source tasks, we sample the links' size and the joint range parameters as follows:

1. Hopper:
   - Foot Link Size: $U[0.05, 0.07] \times U[0.1755, 0.2145]$
   - Leg Link Size: $U[0.03, 0.05] \times U[0.23, 0.27]$
   - Thigh Link Size: $U[0.04, 0.06] \times U[0.2, 0.25]$
   - Torso Link Size: $U[0.04, 0.06] \times U[0.2, 0.25]$
   - Thigh Joint $U[-2.79253, -2.44346] \times U[0, 0]$
2. Walker:
   - Left Foot Link Size: $U[0.05, 0.07] \times U[0.09, 0.11]$
   - Right Foot Link Size: $U[0.05, 0.07] \times U[0.09, 0.11]$
   - Left Leg Link Size: $U[0.03, 0.05] \times U[0.23, 0.27]$
   - Right Leg Link Size: $U[0.03, 0.05] \times U[0.23, 0.27]$

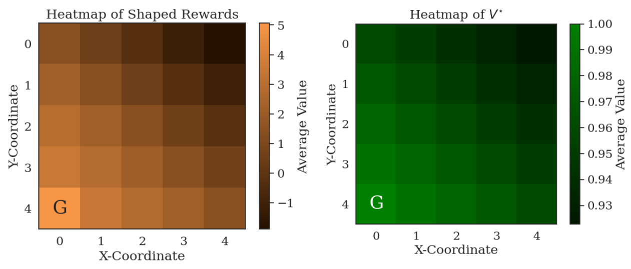

Figure 7. Heatmaps representing the values of the shaped reward (left) and the optimal value function (right) corresponding to each position the agent could move to in the 5x5 grid. The goal (represented with the G) is in the bottom left corner of the grid.

Method	**Number of Agents**	
	1 Agent	**5 Agents**
`EvIL`	0.4082 ($\pm$ 0.1166)	**0.6696** ($\pm$ 0.0176)
`EvIL` - No buffer	0.2702 ($\pm$ 0.1204)	0.5739 ($\pm$ 0.0192)
`EvIL` - No L2	0.3942 ($\pm$ 0.0861)	0.6124 ($\pm$ 0.0233)
`EvIL` - Bigger Reward Net	0.2603 ($\pm$ 0.0901)	0.5058 ($\pm$ 0.0057)
`EvIL` - No Reset	0.0951 ($\pm$ 0.1111)	0.2544 ($\pm$ 0.0170)
IRL	-0.0180 ($\pm$ 0.0895)	0.1424 ($\pm$ 0.0153)
IRL + Resets	0.2912 ($\pm$ 0.1573)	0.6277 ($\pm$ 0.0279)
IRL + L2	-0.0579 ($\pm$ 0.2612)	0.2724 ($\pm$ 0.1020)
IRL + Bigger Reward Net	-0.0940 ($\pm$ 0.1415)	-0.0216 ($\pm$ 0.0444)
IRL + Buffer	0.2053 ($\pm$ 0.0413)	0.4310 ($\pm$ 0.0180)

Table 1. Correlation between the recovered reward and the real reward at the end of the IRL procedure. Ablation for `EvIL` and IRL. We report mean and standard error on the states visited by the learner.

- Left Thigh Link Size: $U[0.04, 0.06] \times U[0.2, 0.25]$
- Right Thigh Link Size: $U[0.04, 0.06] \times U[0.2, 0.25]$
- Left Thigh Joint $U[-2.79253, -2.44346] \times U[0, 0]$

3. Ant:
   - Left Leg Link Size: $U[0.05, 0.11]$
   - Right Leg Link Size: $U[0.05, 0.11]$
   - Left Leg Link Size: $U[0.05, 0.11]$
   - Back Right Leg Link Size: $U[0.05, 0.11]$
   - Aux 1 Link Size: $U[0.05, 0.11]$
   - Aux 2 Link Size: $U[0.05, 0.11]$
   - Aux 3 Link Size: $U[0.05, 0.11]$
   - Aux 4 Link Size: $U[0.05, 0.11]$
   - Left Ankle Link Size: $U[0.05, 0.11]$
   - Right Ankle Link Size: $U[0.05, 0.11]$
   - Back Left Ankle Link Size: $U[0.05, 0.11]$
   - Back Right Ankle Link Size: $U[0.05, 0.11]$
   - Joint Hip 1 $U[-40, -20] \times U[20, 40]$
   - Joint Hip 2 $U[-40, -20] \times U[20, 40]$
   - Joint Hip 3 $U[-40, -20] \times U[20, 40]$
   - Joint Hip 4 $U[-40, -20] \times U[20, 40]$
   - Joint Ankle 1 $U[-80, -60] \times U[-40, -20]$
   - Joint Ankle 2 $U[-80, -60] \times U[-40, -20]$

- Joint Ankle 3 $U[-80, -60] \times U[-40, -20]$
- Joint Ankle 4 $U[-80, -60] \times U[-40, -20]$

C. Hyperparameters

Table 2. Hyperparameters for Training IRL

Parameter	Value
Number of Reward Hidden Layers	2
Size of Reward Hidden Layer	128
Reward Activation	tanh
Number of Outer Loop Steps	2441
Inner Loop Learning Rate	4e-3
Inner Loop Gradient Updates	1
Inner Loop Num Steps	10
Inner Loop Num Envs	2048
Num Trajectories Sampled each Policy Update	10
Gradient Penalty Coefficient	10
Outer Loop Learning Rate Schedule	Linear
Number of Discriminators	5
ℓ_2 coefficient	0.0
Outer Loop Learning Rate Start	1e-2
Outer Loop Learning Rate End	1e-5

For the Ant environment, the Final Outer Loop Learning Rate is 1e-6.

Table 3. Important parameters for Training Reward Shaping with ES

Parameter	Value
Population Size	64
Number of Reward Hidden Layers	2
Size of Reward Hidden Layer	128
Reward Activation	tanh
Number of Generations	600
ES Sigma Init	0.03
ES Sigma Decay	1.00
ES LR	1e-3
Outer Loop Learning Rate	5e-3