Hybrid Inverse Reinforcement Learning

Juntao Ren [*1] **Gokul Swamy** [*2] **Zhiwei Steven Wu** [2] **J. Andrew Bagnell** [3 2] **Sanjiban Choudhury** [1]

Abstract

The inverse reinforcement learning approach to imitation learning is a double-edged sword. On the one hand, it can enable learning from a smaller number of expert demonstrations with more robustness to error compounding than behavioral cloning approaches. On the other hand, it requires that the learner repeatedly solve a computationally expensive reinforcement learning (RL) problem. Often, much of this computation is wasted searching over policies very dissimilar to the expert's. In this work, we propose using *hybrid RL* – training on a mixture of online and expert data – to curtail unnecessary exploration. Intuitively, the expert data focuses the learner on good states during training, which reduces the amount of exploration required to compute a strong policy. Notably, such an approach doesn't need the ability to reset the learner to arbitrary states in the environment, a requirement of prior work in efficient inverse RL. More formally, we derive a reduction from inverse RL to *expert-competitive RL* (rather than globally optimal RL) that allows us to dramatically reduce interaction during the inner policy search loop while maintaining the benefits of the IRL approach. This allows us to derive both model-free and model-based hybrid inverse RL algorithms with strong policy performance guarantees. Empirically, we find that our approaches are significantly more sample efficient than standard inverse RL and several other baselines on a suite of continuous control tasks.

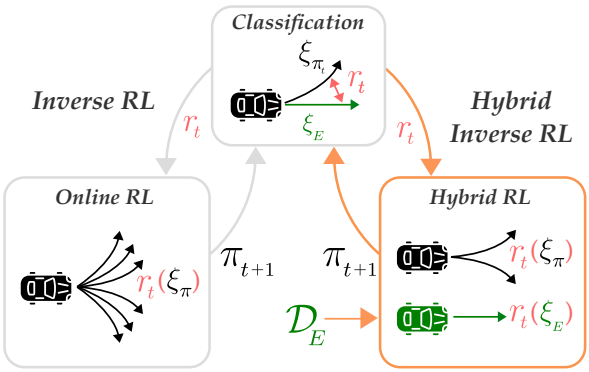

Figure 1: Standard inverse reinforcement algorithms (left) require repeatedly solving a reinforcement learning problem in their inner loop. Thus, the learner is potentially forced to explore the entire state space to find any reward. We introduce *hybrid inverse reinforcement learning*, where the learner trains on a mixture of its own and the expert's data during the policy search inner loop. This reduces the exploration burden on the learner by providing positive examples. We provide model-free and model-based algorithms that are both significantly more sample efficient than standard inverse RL approaches on continuous control tasks.

1. Introduction

Broadly speaking, we can break down the approaches to imitation learning (IL) into *offline* algorithms (e.g. behavioral cloning, (Pomerleau, 1988)) and *interactive* algorithms (e.g. inverse reinforcement learning (Ziebart et al., 2008a), DAgger (Ross et al., 2011)). Offline approaches to imitation aren't robust to the covariate shift between the expert's state distribution and the learner's *induced* state distribution; therefore, they suffer from compounding errors which results in poor test-time performance (Ross et al., 2011; Swamy et al., 2021; Wang et al., 2021). Instead, interactive algorithms allow the the learner to observe the consequences of their actions and therefore learn to recover from their own mistakes. This is the fundamental reason why interactive approaches like inverse reinforcement learning (IRL) continue to provide state-of-the-art performance for challenging tasks like autonomous driving (Bronstein et al., 2022; Igl et al., 2022; Vinitsky et al., 2022) and underlie large-scale services like Google Maps (Barnes et al., 2023).

However, inverse reinforcement learning reduces the problem of imitation to repeatedly solving a reinforcement learning problem, and thus the agent has to potentially pay the exponential interaction complexity of reinforcement learn-

[*]Equal contribution [1]Cornell University [2]Carnegie Mellon University [3]Aurora Innovation. Correspondence to: Gokul Swamy <gswamy@cmu.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

ing (Kakade, 2003). When performed in the real world, this interaction can be both unsafe and time-consuming; in simulation, it incurs great computational expense. This fact motivates our key question: *how can we reduce the amount of interaction performed in inverse reinforcement learning?*

At its core, the reason reinforcement learning is interaction-inefficient is because of global *exploration*: in the worst case, the learner needs to reach all possible states to figure out what decisions are optimal over the horizon. This means that in IRL, the learner often spends the majority of interactions trying out policies that are quite dissimilar to the expert's in the hope of finding a bit of reward. This is rather odd, given our goal is just to imitate the expert. Put differently, when optimizing a potential reward function, we should only be competing against policies with similar visitation distributions to the expert. We therefore narrow our overarching question: *how do we focus IRL policy search on policies that are similar to the expert's?*

Recent work by Swamy et al. (2023) shows that one can dramatically reduce the amount of exploration required by resetting the learner to states from the expert demonstrations during policy search. Despite this approach's strong theoretical guarantees, the ability to reset the learner to arbitrary states has limited feasibility in the real world. Thus, we focus on how we can curtail unnecessary exploration *without* assuming *generative model* access to the environment.

We provide a general reduction that allows one to use *any* RL algorithm that merely guarantees returning a policy that competes with the expert (rather than competing with the optimal policy) for policy search in IRL. This generalizes the argument of Swamy et al. (2023), allowing us to leverage efficient RL algorithms that don't require resets for IRL.

Specifically, we propose to use *hybrid* reinforcement learning (Ross & Bagnell, 2012; Song et al., 2022; Zhou et al., 2023) to speed up the policy search component of inverse reinforcement learning. In hybrid RL, one trains a policy to do well on *both* the offline data and the distribution of data it induces (e.g. by using data from both buffers when fitting a Q-function). Rather than competing against an arbitrary policy (as we do in online RL which therefore leads to having to pay for extensive exploration), this procedure asks only the learner to compete against policies *covered* by the offline dataset. Hybrid RL gives similar theoretical guarantees as offline RL without requiring explicit pessimism (which can be brittle in practice and intractable in theory) and retains a higher degree of robustness to covariate shift. Given our goal is to compete with the expert, we propose to simply use the expert demonstrations as the offline dataset for hybrid RL. Our key insight is that ***we can use hybrid RL as the policy search procedure in IRL to curtail exploration.***

More explicitly, the contributions of our work are three-fold:

1. We provide a reduction from inverse RL to expert-competitive RL. We prove that as long as our policy search procedure guarantees to output a sequence of policies that competes with the expert *on average* over a sequence of chosen rewards, we are able to compute a policy that competes with the expert on the ground truth reward. Notably, our reduction generalizes the underlying argument of Swamy et al. (2023) to methods that don't require the ability to reset the learner to arbitrary states in the environment.

2. We derive two hybrid inverse RL algorithms: model-free `HyPE` and model-based `HyPER`. `HyPE` uses the HyQ algorithm of Song et al. (2022) in its inner loop while `HyPER` uses the LAMPS algorithm of Vemula et al. (2023) as its policy search procedure. We provide performance guarantees for both algorithms and discuss their pros and cons relative to other fast inverse RL algorithms.

3. We demonstrate that on a suite of continuous control tasks, `HyPE` and `HyPER` are significantly more sample efficient than standard approaches. In addition to outperforming GAIL-like approaches (Ho & Ermon, 2016) and behavioral cloning, we also find that we are able to consistently out-perform the FILTER algorithm of (Swamy et al., 2023) and IQLearn (Garg et al., 2021).

2. Related Work

Hybrid Reinforcement Learning. Hybrid RL — using data that covers a strong policy to speed up reinforcement learning — comes in several variants. One variant is to *reset* the learner to states from the offline distribution (Bagnell et al., 2003; Ross & Bagnell, 2014). Another is *hybrid training*: using a combination of on-policy data from the learner and the offline distribution when fitting the critic (Song et al., 2022; Zhou et al., 2023) or the model (Ross & Bagnell, 2012; Vemula et al., 2023) used in the policy update. While the former variant comes with stronger guarantees as far as interaction complexity, the latter is more applicable to a wider variety of problems due to weaker reset requirements (Hester et al., 2018a; Ball et al., 2023; Luo et al., 2023). We lift the insights from two hybrid training algorithms — the HyQ algorithm of Song et al. (2022) and the LAMPS algorithm of (Vemula et al., 2023) — to the space of imitation learning.

Sample-Efficient Inverse Reinforcement Learning. Various lines of work have attempted to address the sample-inefficiency of the RL inner loop of inverse RL. One line removes the inner loop entirely by using the difference of Q functions across timesteps as an implicit reward (Dvijotham & Todorov, 2010; Garg et al., 2021). However, since Q functions depend on the dynamics of the environment, it is unclear if such methods will produce consistent estimates of the expert's policy if data was collected across agents with

Algorithm 1 (Dual) IRL (Ziebart et al. (2008b))

Input: Demos. $\mathcal{D}_E$, Policy class Π, Reward class $\mathcal{F}_r$
Output: Trained policy π
Initialize $\pi_1 \in \Pi$
for t in $1 \dots T$ **do**
  // Use any no-regret algo to pick f
  $f_t \leftarrow f_{t-1} + \nabla_f (J(\pi_E, f) - J(\pi_t, f))$
  $\pi_{t+1} \leftarrow \texttt{RL}(r = f_t, \Pi = \Pi)$
end for
Return mixture of $\pi_{1:T}$.

slightly different dynamics (e.g. cars with different wheel frictions). On the other hand, reward-based methods like ours have repeatedly demonstrated robustness to this issue (Silver et al., 2010; Ratliff et al., 2009; Kolter et al., 2008; Ng et al., 2006; Zucker et al., 2011; Ziebart et al., 2008c).

Another line of work attempts to use resets to the expert's state distribution to curtail the exploration the learner performs during the inner loop (Swamy et al., 2023). This approach comes with strong guarantees for interaction efficiency but requires generative model access to the environment. Our approach operates under weaker reset assumptions but can only provide weaker guarantees. More explicitly, we lift the reset flavor of hybrid RL to imitation, while we lift hybrid training. We provide a more in-depth comparison efficient IRL methods in Section 3.5.

Hybrid Training for Inverse Reinforcement Learning. Perhaps the most similar approaches to our model-free HyPE algorithm are the SQIL approach of Reddy et al. (2019) and the AdRIL approach of Swamy et al. (2021). Both approaches use data from both the learner and the expert during policy updates by sampling from two separate replay buffers. However, neither of these works rigorously addresses the effect of using off-policy data in the policy optimization subroutine of inverse RL. We provide a general reduction that specifies the properties required for doing so while preserving performance guarantees. In concurrent work, Kolev et al. (2024) combine hybrid training with pessimism for more interaction-efficient model-based IRL, using a model ensemble disagreement auxiliary cost in their practical RL procedure. In contrast, we focus on the benefits hybrid training + expert resets provides for model-based IRL. This allows us to elide intractable pessimism in theory and ensemble-based approximations in practice.

3. Hybrid Inverse RL

3.1. A Game-Theoretic Perspective on Inverse RL

We consider a finite-horizon Markov Decision Process (MDP) (Puterman, 2014) parameterized by $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, H \rangle$ 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, $r : \mathcal{S} \times \mathcal{A} \to [-1, 1]$ is the reward function, and H is the horizon. In the inverse RL setup, 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 policy that, no matter what reward function we are evaluated under, performs as well as the expert. We cast this problem as a zero-sum game between a policy player and an adversary that tries to pick out differences between expert and learner policies (Syed & Schapire, 2007; Swamy et al., 2021). More formally, we optimize over policies $\pi : \mathcal{S} \to \Delta(\mathcal{A}) \in \Pi$ and reward functions $f : \mathcal{S} \times \mathcal{A} \to [-1, 1] \in \mathcal{F}_r$. We use $\xi = (s_1, a_1, r_1, \dots, s_H, a_H, r_H)$ to denote the trajectory generated by some policy. For theoretical simplicity, we assume that our strategy spaces (Π and $\mathcal{F}_r$) are convex and compact, that $\mathcal{F}_r$ is closed under negation, and that $r \in \mathcal{F}_r, \pi_E \in \Pi$. 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 can express our objective as

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

Perhaps the most common strategy for solving this game is to use a no-regret strategy for reward selection against a best-response strategy for policy selection (Ziebart et al., 2008b; Swamy et al., 2021), which we outline in Algorithm 1. Explicitly, in the *dual* flavor of IRL, one performs a *best-response* via RL at each inner iteration by computing the optimal policy for the adversarially chosen reward function.

Consider the learner operating in a tree-structured MDP and let $\mathcal{F}_r$ be the class of reward functions that are 0 everywhere except for a single leaf node. Then, to find any reward, the learner needs to explore the entire tree at each iteration, an amount of interaction that scales exponentially with the task horizon. This construction is more than a pathological example: Fig. 2 shows that even for *primal* approaches (e.g. GAIL) that don't optimize the reward to completion in their inner loop (and instead perform a small no-regret update – i.e. a gradient step), we observe empirical evidence of the adversary being able to consistently pick out differences between the expert and the learner, and drive up the value difference $J(\pi_E, f_i) - J(\pi, f_i)$. Even as the learner slowly improves upon the current reward, the adversary repeatedly shifts the reward function, thus driving up interactions and introducing instability. Effectively, any adversarial IRL algorithm (whether primal, dual or a mixture) ends up solving a potentially hard global exploration problem *at least* once.

Indeed, one may need to explore the entire state space for the optimal policy when given an arbitrary reward function. However, in inverse RL we explicitly choose reward functions that rate the expert higher than the history of learner policies. Thus, the fact that the standard recipe for both primal and dual IRL completely ignore the expert demon-

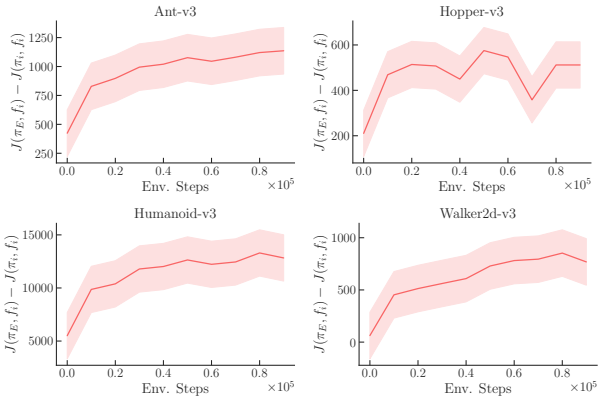

Figure 2: Difference in rewards between the learner policy π_i and expert policy π_E under the discriminator function f_i for the first 100k environment interactions in primal IRL.

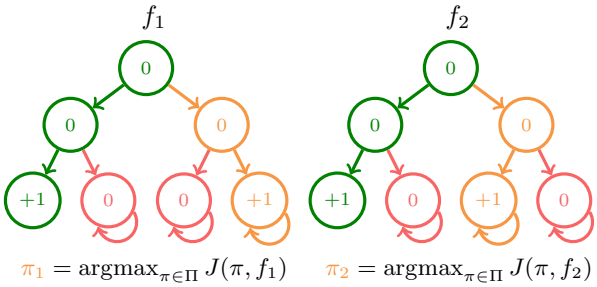

Figure 3: Consider a binary tree MDP. Define Π to be the set of all deterministic policies (paths through the tree), and $\mathcal{F}_r$ the class of rewards that always assign $+1$ to the bottom-left node and an additional $+1$ to any one of the three other leaf nodes. The expert (the green path) always takes the leftmost path. Note that the expert is not optimal under any $f \in \mathcal{F}_r$. In the first image, the learner (the orange path) has computed the best response to f_1 (the labels on the nodes). To penalize the learner, f_2 shifts the reward to a neighboring leaf node. As a result, π_2 must search through the entire tree to compute the best-response. Beyond the repeated exploration required to compute a best-response, the best responses are different across iterations, which leads to instability in policy training.

strations during policy optimization seems suboptimal. This begs the question: *how can we give the learner examples of expert behavior during policy optimization to reduce the amount of exploration required to find good states?* We now discuss the core principle that underlies multiple ways to do so while preserving performance guarantees.

3.2. Expert-Relative Regret Oracles in Inverse RL

At heart, the game-theoretic approach to inverse reinforcement learning relies on the following intuition: *we must have found a policy with performance close to the expert's if there is no reward function that can tell the difference between the learner's current policy and the expert's.* We often operationalize this principle by repeatedly picking reward functions that score the expert higher than the learner and then computing the optimal policy under this reward. Critically, however, there is no reason for the expert to be the *optimal* policy under a proposed reward function – it merely needs to score higher than the learner. We provide a simple example of this point in Fig. 3.

Thus, when computing a *best response* to this reward function (i.e. the optimal policy), the learner often needs to try out a variety of policies that visit states quite different to those the expert visits. This has two negative consequences: first, it may necessitate a large amount of interaction per iteration. Second, because the optimal policy can vary wildly across iterations, it can introduce instability into the training process. Both concerns still apply even for primal approaches like GAIL (Ho & Ermon, 2016) that take a small no-regret step at each iteration.[1] If we pause and take a step back, this is somewhat odd: given our goal is merely to compete with the expert under a variety of potential reward functions, trying to move towards the *optimal policy* under an adversarially chosen metric seems needlessly expensive. Instead, one might hope that as long as the learner can consistently compete with the *expert* under whatever metric the adversary chooses, we should be able to guarantee that we compete with the expert under the ground-truth reward.

We can formalize our preceding intuition via the notion of an ***Expert-Relative Regret Oracle*** (ERROr).

Definition 3.1 (ERROr$\{\mathsf{Reg}_\pi(T)\}$)**.** A policy-selection algorithm $\mathbb{A}_\pi$ satisfies the $\mathsf{Reg}_\pi(T)$ expert-relative regret guarantee if given any sequence of reward functions $f_{1:T}$, it produces a sequence of policies $\pi_{t+1} = \mathbb{A}_\pi(f_{1:t})$ such that

$$\sum_{t=1}^{T} J(\pi_E, f_t) - J(\pi_t, f_t) \leq \mathsf{Reg}_\pi(T). \tag{2}$$

Critically, this definition does not require us to compute or even compete against the optimal policy for each f_t. [2]

We also define a no-regret reward-selection algorithm.

[1] To see why the former point applies, observe that even if we were to restrict the adversary to only picking the ground truth r, a primal approach would eventually involve computing the optimal policy, which requires extensive exploration in the worst case.

[2] Note that this is a weaker requirement than the per-iteration guarantee the proofs of Swamy et al. (2023) require. While this distinction may seem unimportant prima facie, the model-based hybrid RL algorithms we build only guarantee competing with the expert on average (Ross & Bagnell, 2012; Vemula et al., 2023).

Definition 3.2. Define $\ell_t(f) = \frac{1}{H}(J(\pi_t, f) - J(\pi_E, f))$. $\mathbb{A}_f$ is a *no-regret reward selection algorithm* if when given a sequence of loss functions $\ell_{1:t}$ induced by a sequence of policies $\pi_{1:t}$, it produces iterates $f_{t+1} = \mathbb{A}_f(\ell_{1:t})$ such that

$$\sum_{t=1}^{T} \ell_t(f_t) - \min_{f^\star \in \mathcal{F}_r} \sum_{t=1}^{T} \ell_t(f^\star) \leq \mathsf{Reg}_f(T), \tag{3}$$

with $\lim_{T\to\infty} \frac{\mathsf{Reg}_f(T)}{T} = 0$.

Due to the linearity of ℓ_t, standard online no-regret algorithms like gradient descent satisfy this above condition (Zinkevich, 2003). We now provide a simple proof that the combination of the above two oracles allows us to efficiently compute a policy with similar performance to the expert's.

Theorem 3.3. *Assume access to an $\mathbb{A}_\pi$ and $\mathbb{A}_f$ that satisfy Definitions 3.1 and 3.2 respectively. Set $\pi_{t+1} = \mathbb{A}_\pi(f_{1:t})$ and $f_{t+1} = \mathbb{A}_f(\ell_{1:t})$. Then, $\bar{\pi}$ (the mixture of $\pi_{1:T}$) satisfies*

$$J(\pi_E, r) - J(\bar{\pi}, r) \leq \frac{\mathsf{Reg}_\pi(T)}{T} + \frac{\mathsf{Reg}_f(T)}{T} H. \tag{4}$$

Proof.

$$\begin{aligned}
J(\pi_E, r) - J(\bar{\pi}, r) &= \frac{1}{T}\sum_{t=1}^{T} J(\pi_E, r) - J(\pi_t, r) \\
&\leq \max_{f^\star \in \mathcal{F}_r} \frac{1}{T}\sum_{t=1}^{T} J(\pi_E, f^\star) - J(\pi_t, f^\star) \\
&\leq \frac{1}{T}\sum_{t=1}^{T} J(\pi_E, f_t) - J(\pi_t, f_t) \\
&\quad + \frac{\mathsf{Reg}_f(T)}{T} H \\
&\leq \frac{\mathsf{Reg}_\pi(T)}{T} + \frac{\mathsf{Reg}_f(T)}{T} H.
\end{aligned}$$

□

As $T \to \infty$, the second term in the above bound goes to 0 due to the no-regret property of $\mathbb{A}_f$. Thus, in the limit, we only pay for our average policy optimization error relative to the expert. More explicitly, the above result implies that rather than the per-iteration best response we require in dual IRL algorithms like MaxEnt IRL (Ziebart et al., 2008b),

$$\pi_t = \operatorname*{argmax}_{\pi \in \Pi} J(\pi, f_t), \tag{5}$$

or the no-regret property required to prove guarantees for primal IRL algorithms like GAIL (Ho & Ermon, 2016),

$$\lim_{T\to\infty} \max_{\pi^\star \in \Pi} \frac{1}{T}\sum_{t=1}^{T} J(\pi_t, f_t) - J(\pi^\star, f_t) = 0, \tag{6}$$

we instead merely need to compete with the expert on average to ensure that we learn a policy with strong performance.

Our above discussion begs the question of whether there are efficient algorithms that satisfy the ERROr property. Two algorithms that do are the PSDP algorithm of Bagnell et al. (2003) and the NRPI algorithm of Ross & Bagnell (2014). Swamy et al. (2023) essentially use this property in the proofs of their MMDP and NRMM algorithms. Unfortunately, both PSDP and NRPI require the ability to reset the learner to arbitrary states in the environment, which means MMDP and NRMM do as well. Thus we ask the question: *are there algorithms that satisfy the* ERROr *property without requiring generative model access to the environment?* As we detail in the following sections, hybrid RL algorithms answer this question in the affirmative, boding well for their application to the imitation learning setting.

3.3. `HyPE`: Model-Free Hybrid Inverse RL

We now consider how to construct a *model-free* hybrid IRL algorithm. We begin by considering the forward problem. Many model-free hybrid RL algorithms follow the rough structure we outline in Algorithm 3: use a mixture of on-policy data from the learner and off-policy data from the expert to fit a Q function that we then use for a policy update. For example, the HyQ algorithm of Song et al. (2022) runs Fitted Q Iteration for the critic update and then returns the greedy policy from that critic update for the actor update. The HNPG / HAC algorithms of Zhou et al. (2023) performs a similar critic update before using the Natural Policy Gradient algorithm of Kakade (2001) / soft policy iteration algorithm of Ziebart et al. (2008b) as the actor update. In practice, it is common to just run an off-policy RL algorithms like Soft Actor Critic (Haarnoja et al., 2018) with expert data in the replay buffer (Ball et al., 2023). For the purposes of analysis however, we assume the learner picks policies by running HyQ with the expert demonstrations $\mathcal{D}_E$ as the offline dataset and $-f_t$ as the reward function. As argued by Song et al. (2022), under certain assumptions (e.g. Bellman Completeness of $\mathcal{F}_Q$ and an MDP with low Bilinear Rank (Du et al., 2021)), running HyQ for M steps guarantees that the average of the M policies $\bar{\pi}_t$ satisfies

$$J(\pi_E, f_t) - J(\bar{\pi}_t, f_t) \leq H^2 O\left(\frac{1}{\sqrt{M}}\right). \tag{7}$$

Observe that if we were to use HyQ to select each π_t, we would satisfy the ERRoR property with $\mathsf{Reg}_\pi(T) \leq TH^2O\left(\frac{1}{\sqrt{M}}\right)$. Critically, we do not need the ability to reset the learner to arbitrary states in the environment to run hybrid training algorithms like HyQ, allowing us to curtail exploration *without* generative model access. We refer to this approach as `HyPE`: **Hy**brid **P**olicy **E**mulation and outline it in Algorithm 2. Running `HyPE` with HyQ as hybrid RL oracle results in the following performance bound.

Algorithm 2 **Hy**brid **P**olicy **E**mulation (`HyPE`)

Input: Expert demonstrations $\mathcal{D}_E$, Policy class Π, Reward class $\mathcal{F}_r$, Critic class $\mathcal{F}_Q$, Learning rate η.
Output: Trained policy π
Initialize $f_1 \in \mathcal{F}_r, \pi_1 \in \Pi, Q_1 \in \mathcal{F}_Q$
for t in $1 \dots T$ **do**
  `// No-regret step over rewards`
  $f_{t+1} \leftarrow f_t + \eta \nabla_f (J(\pi_t, f) - J(\pi_E, f))$
  `// Update policy via hybrid RL`
  $\pi_{t+1}, Q_{t+1} \leftarrow \texttt{HyRL}(\pi_t, Q_t, f_{t+1}, \mathcal{D}_E, \dots)$
end for
Return best of $\pi_{1:T}$ on validation data.

Algorithm 3 **Hy**brid **RL** (`HyRL`)

Input: Expert demonstrations $\mathcal{D}_E$, Policy class Π, Critic class $\mathcal{F}_Q$, Batch size B, Inner steps N, Current policy π_t, Current critic Q_t, Current cost f_t.
Output: Trained policy π
Initialize $\pi_1 = \pi_t, Q_1 = Q_t, \mathcal{D}_{mix} = \{\}$
for i in $1 \dots N$ **do**
  `// Collect on-policy data`
  $\mathcal{D}_i \leftarrow \{\xi_{1:B} \sim \pi_i\}$
  $\mathcal{D}_{mix} \leftarrow \mathcal{D}_{mix} \cup \mathcal{D}_i \cup \{\xi_{1:B} \sim \mathcal{D}_E\}$
  `// Perform hybrid updates`
  $Q_{i+1} \leftarrow \mathsf{critic\_update}(Q_i, \pi_i, -f_t, \mathcal{D}_{mix}, \mathcal{F}_Q)$
  $\pi_{i+1} \leftarrow \mathsf{actor\_update}(\pi_i, Q_{i+1}, -f_t, \mathcal{D}_{mix}, \Pi)$
end for
Return Best of $\pi_{1:N}, Q_{1:N}$ on validation data.

Corollary 3.4 (`HyPE` Performance Bound)**.** *Consider running `HyPE` (Algorithm 2) with M iterations of HyQ as the hybrid RL subroutine. Then, we have the following:*

$$J(\pi_E, r) - J(\bar{\pi}, r) \leq H^2 O\left(\frac{1}{\sqrt{M}}\right) + \frac{\mathsf{Reg}_f(T)}{T} H. \quad (8)$$

Proof. This follows directly from Theorem 3.3 and the fact that $\mathsf{Reg}_\pi(T) \leq TH^2 O\left(\frac{1}{\sqrt{M}}\right)$, which follows from the fact that the per-iteration sub-optimality with respect to the expert is upper bounded by $H^2 O\left(\frac{1}{\sqrt{M}}\right)$. □

Intuitively, this bound tells us that with sufficient inner loop (M) and outer loop (T) iterations, we can guarantee that we will find a policy with similar performance to that of the expert under the ground-truth reward function. More precisely, it tells us that we need to perform $O(H^2)$ inner loop steps to avoid compounding errors, assuming realizability of the expert policy. Critically, unlike FILTER, `HyPE` does not require generative model access to the environment. [3]

[3] An open question for future work is the robustness of the hybrid approach to compounding errors when the expert policy *isn't* realizable; existing interactive algorithms like MaxEntIRL (Ziebart et al., 2008b; Swamy et al., 2021) and DAgger (Ross et al., 2011) are known to be robust to such mis-specification, and we conjecture the same for the hybrid approach espoused here.

Algorithm 4 **Hy**brid **P**olicy **E**mulation w/ **R**esets (`HyPER`)

Input: Expert demos. $\mathcal{D}_E$, Policy class Π, Reward class $\mathcal{F}_r$, Batch size B, Model class $\mathcal{F}_M$, Learning rate η.
Output: Trained policy π
Initialize $f_1 \in \mathcal{F}_r, \pi_1 \in \Pi, M_1 \in \mathcal{F}_M$
for t in $1 \dots T$ **do**
  `// No-regret step over rewards`
  $f_{t+1} \leftarrow f_t + \eta \nabla_f (J(\pi_t, f) - J(\pi_E, f))$
  `// No-regret hybrid step over models`
  $\mathcal{D}_t \leftarrow \{\xi_{1:B} \sim \pi_t\}, \mathcal{D}_{mix} \leftarrow \mathcal{D}_t \cup \{\xi_{1:B} \sim \mathcal{D}_E\}$
  $M_{t+1} \leftarrow M_t - \eta \nabla_M \mathbb{E}_{\mathcal{D}_{mix}}[-\log(M(s' \mid s, a))]$
  `// Update policy via MBRL w/ resets`
  $\pi_{t+1} \leftarrow \mathrm{argmax}_{\pi \in \Pi} \mathbb{E}_{\substack{h \sim \mathsf{Unif}[1,H] \\ s_h \sim \mathcal{D}_E[h]}} [V^\pi(s_h \mid -f_t, M_t)]$
end for
Return best of $\pi_{1:T}$ on validation data.

3.4. `HyPER`: Model-Based Hybrid Inverse RL

We now consider how best to design a *model-based* hybrid IRL algorithm, again first considering the forward problem. A common recipe for hybrid model-based RL is to *(1)* fit a model on a mixture of learner and expert data, *(2)* compute the optimal policy in this model, and *(3)* go back to Step *(1)* (Ross & Bagnell, 2012). While Step *(2)* doesn't require any real-world interaction, it can still involve an amount of computation in the model that scales with $\exp(H)$. To deal with this concern, Vemula et al. (2023) suggest running the No Regret Policy Iteration (NRPI) algorithm of Ross & Bagnell (2014) *inside* the model, which comes with strong $\mathsf{poly}(H)$ interaction complexity guarantees. Practically, this looks like model-based RL but with resets to states from the expert's state distribution – as we've fit this model ourselves, we clearly have generative model access to it to reset.

`HyPER` (**Hy**brid **P**olicy **E**mulation with **R**esets) lift this idea to the space of imitation learning as outlined in Algorithm 4. In each iteration, `HyPER` picks both an adversarial reward function and a model, and updates the policy using model-based RL with resets. `HyPER` can be thought of as running the LAMPS algorithm of Vemula et al. (2023) with adversarially chosen rewards, or, equivalently, as running the FILTER algorithm of Swamy et al. (2023) inside a model fit in a hybrid fashion. We now prove that the LAMPS algorithm of Vemula et al. (2023) satisfies the ERROr property.

Lemma 3.5. *Given any sequence of reward functions $f_{1:T}$, LAMPS picks a sequence of policies $\pi_{1:T}$ that satisfies*

$$\frac{1}{T} \sum_t^T (J(\pi_E, f_t) - J(\pi_t, f_t)) \leq (\bar{\epsilon}_\pi + 2\sqrt{\bar{\epsilon}_M}) H^2$$

where $\bar{\epsilon}_\pi$ and $\bar{\epsilon}_M$ are the average regret of the policy and model selection subroutines. [Proof]

As was the case for HyPE, LAMPS satisfying the ERRoR property directly implies a performance bound for HyPER.

Corollary 3.6 (HyPER Performance Bound)**.** *Consider running HyPER (Algorithm 4) for T iterations. Then, we have the following performance guarantee for average policy $\bar{\pi}$:*

$$J(\pi_E, r) - J(\bar{\pi}, r) \leq (\bar{\epsilon}_\pi + 2\sqrt{\bar{\epsilon}_M})H^2 + \frac{\mathsf{Reg}_f(T)}{T}H \quad (9)$$

Proof. Follows directly from Theorem 3.3 and Lemma 3.5. □

Observe that this algorithm allows us to gain the computational efficiency benefits of expert resets without needing generative model access to the environment like FILTER. Compared to HyPE, we have to pay $O(H^2)$ in terms of our model learning error. However, this comes with the benefit of only needing to perform policy evaluation rather than policy search in the real world. Thus, we would expect that for problems where we can accurately model the dynamics, HyPER would be more interaction efficient than HyPE.

3.5. Efficient IRL Battle Royale

How does one choose from different flavors of efficient IRL algorithms? The best choice depends on the degree of environment access, type of demonstrations, and whether there are multiple tasks to be performed in a single environment.

If we assume generative model access to the environment, then FILTER (Swamy et al., 2023) seems like the best approach to employ. It comes with strong $\mathsf{poly}(H)$ guarantees on interaction complexity and can also handle scenarios where the demonstrations do not have action labels. However, for many real-world applications such as household robotics, resets to arbitrary states are unrealistic.

If we assume we can model the environment well, then HyPER mitigates the need for expensive, potentially unsafe exploration in the real world. A learned model also allows for resetting, thus providing strong computational efficiency guarantees similar to FILTER. HyPER is particularly useful in multi-task settings where tasks may differ in terms of reward but share common dynamics, e.g. a home robot solving multiple tasks that all share a common physical setup like a kitchen, as explored in Kim et al. (2023).

HyPE requires neither of these assumptions, rendering it the most broadly applicable of the efficient IRL algorithms. However, its interaction efficiency guarantees are strongly tied to the underlying hybrid RL algorithm. For example, to argue that HyQ is more efficient than off-the-shelf fitted Q iteration, one needs strong assumptions like Bellman Completeness of $\mathcal{F}_Q$ and low Bellman Rank (Song et al., 2022). Thus, one needs to be base their selection of policy search procedure on the precise characteristics of the problem they are attempting to solve to be assured of efficiency.

We note that HyPE / HyPER are complimentary to FILTER and can be applied in combination to further boost performance, as demonstrated in our antmaze experiments.

Outside of these, there are other techniques to boost efficiency in IRL. A simple approach is to KL-regularize the learner to a behavior cloning policy (Tiapkin et al., 2023), which has proven empirically successful in a variety of problems. However, there are settings where KL regularization leads to undesirable behavior, for example, averaging across differing behavioral modes in the demonstrations. On problems where it is beneficial however, BC regularization can be combined with any of the strategies for improved sample efficiency. Other approaches involve bypassing the need for a reward model altogether and instead using the difference of Q values to implicitly represent rewards (Dvijotham & Todorov, 2010; Garg et al., 2021) Unfortunately, since Q values depends on the dynamics of the environment, if the dynamics were to change across demonstrations, such approaches may fail to recover consistent estimates of the expert's policy. However, for certain problems where the dynamics are always the same (e.g. language modeling), such approaches can perform well (Cundy & Ermon, 2023).

4. Experiments

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

1. **Are HyPE and HyPER more sample efficient than prior IRL methods?** Since HyPE and HyPER see expert data during their updates, we expect them to converge to expert performance with fewer environment interactions than a standard IRL approaches.
2. **Do HyPE and HyPER handle environments with hard exploration challenges better than prior IRL methods?** We conduct experiments on antmaze-large, where the learner must control a four-legged agent to navigate to a goal within a maze.
3. **Are there cases where a model-based approach provides performance or efficiency gains?** In HyPER, environment interaction is used only for policy evaluation rather than policy search. Thus, we expect HyPER to be even more sample efficient than HyPE.

We implement HyPE by updating the policy and critic networks in Soft Actor Critic (Haarnoja et al., 2018) with expert and learner samples. We implement HyPER by running model-based policy optimization (Janner et al., 2019) and resetting to expert states in the learned model. No reward information is provided in either case, so we also train a

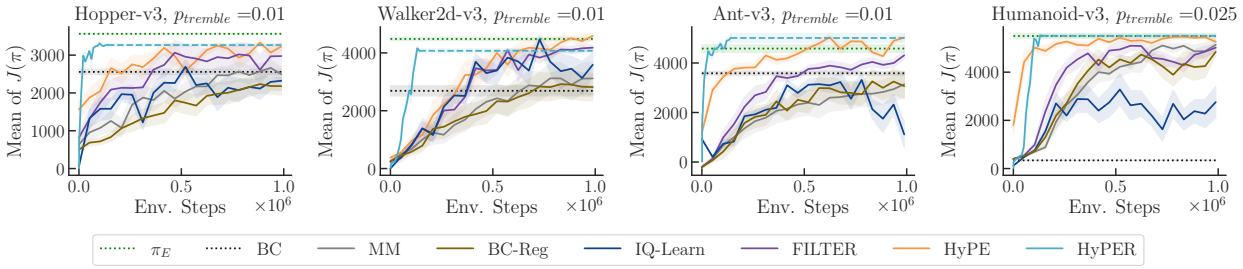

Figure 4: We see HyPER and HyPE achieve the highest reward on the MuJoCo locomotion benchmark. Further, the performance gap increases with the difficulty of the environment (i.e. how far right a plot is in the above figure). We run all model-free algorithms for 1 million environment steps. Due to the higher interaction efficiency of model-based approaches, we only run HyPER for 150k environment steps, after which the last reward is extended horizontally across. We compute standard error across 5 seeds for HyPER, and across 10 seeds for all other algorithms.

discriminator network. We compare against five baselines: BC (behavioral cloning, Pomerleau (1988)), IQLearn (Garg et al., 2021), MM (a baseline moment-matching algorithm that uses an integral probability metric instead of Jensen-Shannon divergence as suggested by Swamy et al. (2022)), BC-Reg (MM with an added Mean-Square Error Loss on the actor update), and FILTER (IRL with resets to expert states, Swamy et al. (2023)). Appendix B includes additional implementation details and hyperparameters. We release the code we used for all of our experiments at **https://github.com/jren03/garage**.

MuJoCo Experiments. On the MuJoCo locomotion benchmark environments (Brockman et al., 2016), all learners are provided with 64 demonstration trajectories [4] from an RL expert to mitigate finite sample concerns. Given simple behavior cloning can match expert performance under these conditions (Swamy et al., 2021), we harden the problem, noise is injected into the environment in the following manner: with probability $p_{tremble}$, a random action is executed instead of the one chosen by the policy. As seen in Figure 4, HyPE and HyPER converge much quicker to expert performance compared to baselines, and the gap increases as environments get more difficult (further right in the figure). While our algorithms have the same worst-case performance bound as FILTER, HyPE and HyPER show to be empirically more sample efficient. We hypothesize this is because HyPE and HyPER use expert states and actions, while FILTER only uses expert states. We find these results to be particularly exciting, as HyPE and HyPER algorithms show competitive performance and sample efficiency without needing expert resets. Finally, IQLearn fails to match BC performance on some environments, and shows signs of unstable training on others. We suspect this is due to the need to model environment stochasticity implicitly.

[4] We perform an ablation of all methods considered in the limited demonstration regime in Appendix D.2.

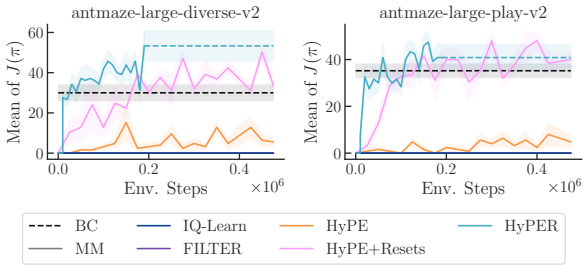

Figure 5: Results on D4RL antmaze-large environment. All interactive baselines achieve 0 reward. While HyPE outperforms prior interactive methods, it does require resets in the environment to beat BC. HyPER is able to surpass BC *without* needing to reset to expert states and match BC performance with roughly 1/10*th* the amount of environment interaction that HyPE + Resets requires. Standard errors are reported across 5 seeds for all algorithms.

D4RL Experiments. Our next set of experiments consider the D4RL (Fu et al., 2020) antmaze-large environments, which is challenging for interactive algorithms that don't use any reward information. We use the standard D4RL dataset and use TD3+BC (Fu et al., 2018) as our policy optimizer. We use a PSDP-inspired reset strategy for HyPER, where if T is total training steps and H the task horizon, then for each iteration t we reset to the set of expert states falling within a sliding window of size $\kappa \in [0, 1]$:

$$\left[H \cdot \left(1 - \frac{t}{T}\right), H \cdot \min\left(1, 1 - \frac{t}{T} + \kappa\right)\right]. \quad (10)$$

In Figure 5, we see that all baseline interactive algorithms (including FILTER [5]) achieve zero reward. This underscores the important of leveraging expert actions for these environ-

[5] This does not conflict with the results of Swamy et al. (2023). While they do not explicitly mention it in their paper, they use hybrid training for their strongest results on these environments.

ments. While HyPE gets off the ground, it fails to match BC performance. If we combine HyPE with resets in the real environment, we find that we are able to improve past BC performance. However, HyPER not only matches the performance of HyPE + Resets with less online interaction, it does so *without needing to do resets in the real environment*. Instead, we perform expert resets within our learned world model. To our knowledge, this is the highest performance achieved by an inverse reinforcement learning algorithm on antmaze, including those that require generative model access to the environment. Thus, we believe that HyPER may have interesting applications to real-world robotics tasks.

5. Conclusion

Despite the many benefits interactive approaches to imitation learning like inverse RL provide, they suffer from a high level of interaction complexity due to their reduction to *repeated* RL. In response, we derive a new paradigm of efficient IRL algorithms via a novel reduction to *expert-competitive* RL. We then instantiate this reduction with *hybrid*, deriving both model-based and model-free algorithms, and show that we can dramatically reduce the amount of interaction required to compute a strong policy by utilizing expert demonstrations during policy search. By preserving the benefits of interaction in imitation while reducing unnecessary exploration, we believe that we have provided the proper algorithmic foundations to take advantage of recent developments in hybrid RL and model-based RL on problems of interest in domains like robotics.

Acknowledgments

JR and SC are supported by the NSF RI #2312956. ZSW is supported in part by the NSF FAI Award #1939606, a Google Faculty Research Award, a J.P. Morgan Faculty Award, a Facebook Research Award, an Okawa Foundation Research Grant, and a Mozilla Research Grant. GS would like to thank Yuda Song for many helpful edits and being a hybrid RL oracle throughout the course of this project and Ken Nakamura for comments on proof style.

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.

Contribution Statements

- **JR** implemented both algorithms and baselines, performed all experiments and ablations, and released a high-quality open-source implementation.
- **GS** proposed the project, derived the key novel reduction, came up with the algorithms, proved all theorems, wrote most of the paper, and provided extensive assistance with debugging all of the experiments.
- **SC** came up with the proof of Lemma 3.5 and wrote Section 3.5 of the paper.
- **ZSW, JAB,** and **SC** advised the project.

References

Bagnell, J. A., Kakade, S. M., Ng, A., and Schneider, J. G. Policy search by dynamic programming. In *NIPS*, 2003.

Ball, P. J., Smith, L., Kostrikov, I., and Levine, S. Efficient online reinforcement learning with offline data. *arXiv preprint arXiv:2302.02948*, 2023.

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, 2023.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Bronstein, E., Palatucci, M., Notz, D., White, B., Kuefler, A., Lu, Y., Paul, S., Nikdel, P., Mougin, P., Chen, H., et al. Hierarchical model-based imitation learning for planning in autonomous driving. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8652–8659. IEEE, 2022.

Cundy, C. and Ermon, S. Sequencematch: Imitation learning for autoregressive sequence modelling with backtracking. *arXiv preprint arXiv:2306.05426*, 2023.

Daskalakis, C., Ilyas, A., Syrgkanis, V., and Zeng, H. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2017.

Du, S., Kakade, S., Lee, J., Lovett, S., Mahajan, G., Sun, W., and Wang, R. Bilinear classes: A structural framework for provable generalization in rl. In *International Conference on Machine Learning*, pp. 2826–2836. PMLR, 2021.

Dvijotham, K. and Todorov, E. Inverse optimal control with linearly-solvable mdps. In *International Conference on Machine Learning*, 2010.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rkHywl-A-.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Garg, D., Chakraborty, S., Cundy, C., Song, J., and Ermon, S. Iq-learn: Inverse soft-q learning for imitation. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Aeo-xqtb5p.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

Hester, T., Vecerík, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., Dulac-Arnold, G., Agapiou, J. P., Leibo, J. Z., and Gruslys, A. Deep q-learning from demonstrations. In *AAAI*, 2018a.

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

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *30th Conference on Neural Information Processing Systems*, 2016.

Igl, M., Kim, D., Kuefler, A., Mougin, P., Shah, P., Shiarlis, K., Anguelov, D., Palatucci, M., White, B., and Whiteson, S. Symphony: Learning realistic and diverse agents for autonomous driving simulation. *arXiv preprint arXiv:2205.03195*, 2022.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Kakade, S. M. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

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

Kim, K., Swamy, G., Liu, Z., Zhao, D., Choudhury, S., and Wu, Z. S. Learning shared safety constraints from multi-task demonstrations. *arXiv preprint arXiv:2309.00711*, 2023.

Kolev, V., Rafailov, R., Hatch, K., Wu, J., and Finn, C. Efficient imitation learning with conservative world models. *arXiv preprint arXiv:2405.13193*, 2024.

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.

Luo, J., Hu, Z., Xu, C., Gadipudi, S., Sharma, A., Ahmad, R., Schaal, S., Finn, C., Gupta, A., and Levine, S. Serl: A software suite for sample-efficient robotic reinforcement learning. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.

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.

Pineda, L., Amos, B., Zhang, A., Lambert, N. O., and Calandra, R. Mbrl-lib: A modular library for model-based reinforcement learning. *arXiv preprint arXiv:2104.10159*, 2021.

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.

Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., and Dormann, N. Stable baselines3, 2019.

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.

Reddy, S., Dragan, A. D., and Levine, S. Sqil: Imitation learning via regularized behavioral cloning. *ArXiv*, abs/1905.11108, 2019.

Ross, S. and Bagnell, J. A. Agnostic system identification for model-based reinforcement learning. *arXiv preprint arXiv:1203.1007*, 2012.

Ross, S. and Bagnell, J. A. Reinforcement and imitation learning via interactive no-regret learning. *ArXiv*, abs/1406.5979, 2014.

Ross, S., Gordon, G. J., and Bagnell, J. A. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 2011.

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.

Song, Y., Zhou, Y., Sekhari, A., Bagnell, J. A., Krishnamurthy, A., and Sun, W. Hybrid rl: Using both offline and online data can make rl efficient. *arXiv preprint arXiv:2210.06718*, 2022.

Swamy, G., Choudhury, S., Bagnell, J. A., and Wu, Z. S. Of moments and matching: A game-theoretic framework for closing the imitation gap. *Proceedings of the 38th International Conference on Machine Learning*, 2021. URL https://arxiv.org/abs/2103.03236.

Swamy, G., Rajaraman, N., Peng, M., Choudhury, S., Bagnell, D., Wu, S., Jiao, J., and Ramchandran, K. Minimax optimal online imitation learning via replay estimation. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=1mFfKXYMg5a.

Swamy, G., Choudhury, S., Bagnell, J. A., and Wu, Z. S. Inverse reinforcement learning without reinforcement learning. *ArXiv*, abs/2303.14623, 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.

Vemula, A., Song, Y., Singh, A., Bagnell, D., and Choudhury, S. The virtues of laziness in model-based rl: A unified objective and algorithms. In *International Conference on Machine Learning*, pp. 34978–35005. PMLR, 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. *arXiv preprint arXiv:2206.09889*, 2022.

Wang, R., Wu, Y., Salakhutdinov, R., and Kakade, S. Instabilities of offline rl with pre-trained neural representation. In *International Conference on Machine Learning*, pp. 10948–10960. PMLR, 2021.

Xie, T. and Jiang, N. Q* approximation schemes for batch reinforcement learning: A theoretical comparison. In *Conference on Uncertainty in Artificial Intelligence*, pp. 550–559. PMLR, 2020.

Zhou, Y., Sekhari, A., Song, Y., and Sun, W. Offline data enhanced on-policy policy gradient with provable guarantees. *arXiv preprint arXiv:2311.08384*, 2023.

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2008a.

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, 2008b.

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, 2008c.

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. Proofs

A.1. Miscellaneous Lemmas

We begin by proving several lemmas that will be helpful in the following proofs.

Lemma A.1 (Policy Evaluation Lemma, (Xie & Jiang, 2020)). *For any policy π and state-action function Q, we have that*

$$\mathbb{E}_{s_0\sim\rho}[\mathbb{E}_{a\sim\pi(s_0)}[Q(s,a)]] - J(\pi) = \mathbb{E}_{\xi\sim\pi}\left[\sum_h^H (Q - \mathcal{T}_r^\pi Q)(s_h, a_h)\right], \tag{11}$$

where $\mathcal{T}_r^\pi$ is the Bellman operator under π and r.

Proof. Xie & Jiang (2020) consider the infinite horizon, discounted setting while we consider the finite horizon, undiscounted setting so we require a slightly different proof.

$$\begin{aligned}
\mathbb{E}_{s_0\sim\rho}[\mathbb{E}_{a\sim\pi(s_0)}[Q(s,a)]] - J(\pi) &= \mathbb{E}_{s_0\sim\rho}[\mathbb{E}_{a\sim\pi(s_0)}[Q(s,a)]] - \mathbb{E}_{\xi\sim\pi}\left[\sum_{h=0}^H r(s_h,a_h)\right] \\
&= \mathbb{E}_{s_0\sim\rho}[\mathbb{E}_{a\sim\pi(s_0)}[Q(s,a)]] - \mathbb{E}_{\xi\sim\pi}\left[\sum_{h=0}^H r(s_h,a_h)\right] \\
&+ \left(\mathbb{E}_{\xi\sim\pi}\left[\sum_{h=1}^H Q(s_h,a_h) - Q(s_h,a_h)\right]\right) \\
&= \mathbb{E}_{\xi\sim\pi}\left[\sum_{h=0}^H Q(s_h,a_h) - (r(s_h,a_h) + Q(s_{h+1},a_{h+1}))\right] \\
&= \mathbb{E}_{\xi\sim\pi}\left[\sum_{h=0}^H (Q - \mathcal{T}_r^\pi Q)(s_h,a_h)\right],
\end{aligned}$$

where we define $Q(s_{H+1}, a_{H+1}) = 0$. □

Lemma A.2 (Performance Difference via Advantage in Model (PDAM, (Vemula et al., 2023))). *Given two policies π_E, π and a model $\hat{M}$, we have that*

$$\begin{aligned}
J_{M^\star}(\pi_E, r) - J_{M^\star}(\pi, r) &= \mathbb{E}_{\xi\sim\pi_E, M^\star}\left[\sum_h^H \mathbb{E}_{a\sim\pi_E(s_h)}[\hat{Q}_r(s_h,a)] - \mathbb{E}_{a\sim\pi(s_h)}[\hat{Q}_r(s_h,a)]\right] \\
&+ \mathbb{E}_{\xi\sim\pi_E, M^\star}\left[\sum_h^H \mathbb{E}_{\substack{s^\star_{h+1}\sim M^\star \\ a\sim\pi(s^\star_{h+1})}}[\hat{Q}_r(s^\star_{h+1},a)] - \mathbb{E}_{\substack{\hat{s}_{h+1}\sim\hat{M} \\ a\sim\pi(\hat{s}_{h+1})}}[\hat{Q}_r(\hat{s}_{h+1},a)]\right] \\
&+ \mathbb{E}_{\xi\sim\pi, M^\star}\left[\sum_h^H \mathbb{E}_{\substack{\hat{s}_{h+1}\sim\hat{M} \\ a\sim\pi(\hat{s}_{h+1})}}[\hat{Q}_r(\hat{s}_{h+1},a)] - \mathbb{E}_{\substack{s^\star_{h+1}\sim M^\star \\ a\sim\pi(s^\star_{h+1})}}[\hat{Q}_r(s^\star_{h+1},a)]\right].
\end{aligned}$$

Proof. We significantly simply the proof of (Vemula et al., 2023). We first break up the performance difference into a sum of three terms.

$$\begin{aligned}
J_{M^\star}(\pi_E, r) - J_{M^\star}(\pi, r) &= \left(J_{M^\star}(\pi_E, r) - \mathbb{E}_{s_0\sim\rho}[\mathbb{E}_{a\sim\pi_E(s_0)}[\hat{Q}_r(s_0,a)]]\right) && \text{(T1)} \\
&- \left(J_{M^\star}(\pi, r) - \mathbb{E}_{s_0\sim\rho}[\mathbb{E}_{a\sim\pi(s_0)}[\hat{Q}_r(s_0,a)]]\right) && \text{(T2)} \\
&+ \left(\mathbb{E}_{s_0\sim\rho}[\mathbb{E}_{a\sim\pi_E(s_0)}[\hat{Q}_r(s_0,a)]] - \mathbb{E}_{s_0\sim\rho}[\mathbb{E}_{a\sim\pi(s_0)}[\hat{Q}_r(s_0,a)]]\right) && \text{(T3)}
\end{aligned}$$

We consider each term separately. First, by Lemma A.1, we have that

$$\begin{aligned}
\text{(T1)} &= \mathbb{E}_{\xi\sim\pi_E,M^\star}\left[\sum_{h=0}^{H}(\mathcal{T}_r^E\hat{Q}_r - \hat{Q}_r)(s_h,a_h)\right] \\
&= \mathbb{E}_{\xi\sim\pi_E,M^\star}\left[\sum_{h=0}^{H}(\cancel{r(s_h,a_h)} + \mathbb{E}_{\substack{s^\star_{h+1}\sim M^\star(s_h,a_h)\\ a\sim\pi_E(s^\star_{h+1})}}[\hat{Q}_r(s^\star_{h+1},a)]) - (\cancel{r(s_h,a_h)} + \mathbb{E}_{\substack{\hat{s}_{h+1}\sim \hat{M}(s_h,a_h)\\ a\sim\pi_E(\hat{s}_{h+1})}}[\hat{Q}_r(\hat{s}_{h+1},a)])\right] \\
&= \mathbb{E}_{\xi\sim\pi_E,M^\star}\left[\sum_{h=0}^{H}\left(\mathbb{E}_{\substack{s^\star_{h+1}\sim M^\star(s_h,a_h)\\ a\sim\pi_E(s^\star_{h+1})}}[\hat{Q}_r(s^\star_{h+1},a)] - \mathbb{E}_{\substack{s^\star_{h+1}\sim M^\star(s_h,a_h)\\ a\sim\pi(s^\star_{h+1})}}[\hat{Q}_r(s^\star_{h+1},a)]\right)\right. \\
&\left. - \sum_{h=0}^{H}\left(\mathbb{E}_{\substack{\hat{s}_{h+1}\sim \hat{M}(s_h,a_h)\\ a\sim\pi_E(\hat{s}_{h+1})}}[\hat{Q}_r(\hat{s}_{h+1},a)] - \mathbb{E}_{\substack{s^\star_{h+1}\sim M^\star(s_h,a_h)\\ a\sim\pi(s^\star_{h+1})}}[\hat{Q}_r(s^\star_{h+1},a)]\right)\right].
\end{aligned}$$

Adding in (T3) to this expression gives us

$$\begin{aligned}
\text{(T1) + (T3)} = \mathbb{E}_{\xi\sim\pi_E,M^\star}&\left[\sum_{h=0}^{H}\left(\mathbb{E}_{a\sim\pi_E(s_h)}[\hat{Q}_r(s^\star_{h+1},a)] - \mathbb{E}_{a\sim\pi(s_h)}[\hat{Q}_r(s^\star_{h+1},a)]\right)\right. \\
&\left. + \sum_{h=0}^{H}\left(\mathbb{E}_{\substack{s^\star_{h+1}\sim M^\star(s_h,a_h)\\ a\sim\pi(s^\star_{h+1})}}[\hat{Q}_r(s^\star_{h+1},a)] - \mathbb{E}_{\substack{\hat{s}_{h+1}\sim \hat{M}(s_h,a_h)\\ a\sim\pi_E(\hat{s}_{h+1})}}[\hat{Q}_r(\hat{s}_{h+1},a)]\right)\right].
\end{aligned}$$

Next, again by Lemma A.1, we have

$$\begin{aligned}
\text{(T2)} &= \mathbb{E}_{\xi\sim\pi,M^\star}\left[\sum_{h=0}^{H}(\hat{Q}_r - \mathcal{T}_r^\pi\hat{Q}_r)(s_h,a_h)\right] \\
&= \mathbb{E}_{\xi\sim\pi,M^\star}\left[\sum_{h=0}^{H}(\cancel{r(s_h,a_h)} + \mathbb{E}_{\substack{\hat{s}_{h+1}\sim \hat{M}(s_h,a_h)\\ a\sim\pi(\hat{s}_{h+1})}}[\hat{Q}_r(\hat{s}_{h+1},a)]) - (\cancel{r(s_h,a_h)} + \mathbb{E}_{\substack{s^\star_{h+1}\sim M^\star(s_h,a_h)\\ a\sim\pi(s^\star_{h+1})}}[\hat{Q}_r(s^\star_{h+1},a)])\right] \\
&= \mathbb{E}_{\xi\sim\pi,M^\star}\left[\sum_{h=0}^{H}(\mathbb{E}_{\substack{\hat{s}_{h+1}\sim \hat{M}(s_h,a_h)\\ a\sim\pi(\hat{s}_{h+1})}}[\hat{Q}_r(\hat{s}_{h+1},a)]) - (\mathbb{E}_{\substack{s^\star_{h+1}\sim M^\star(s_h,a_h)\\ a\sim\pi(s^\star_{h+1})}}[\hat{Q}_r(s^\star_{h+1},a)])\right].
\end{aligned}$$

Adding together the preceding results gives the desired bound. □

A direct corollary of this lemma via applying Hölder's inequality to the last two terms is as follows.

Corollary A.3. *Define $\tilde{\pi}$ as the trajectory-level average of π and π_E. Then, we have that*

$$\begin{aligned}
J_{M^\star}(\pi_E,r) - J_{M^\star}(\pi,r) \le \mathbb{E}_{\xi\sim\pi_E,M^\star}&\left[\sum_{h}^{H}\mathbb{E}_{a\sim\pi_E(s_h)}[\hat{Q}_r(s_h,a)] - \mathbb{E}_{a\sim\pi(s_h)}[\hat{Q}_r(s_h,a)]\right] \\
&+ 2H^2\mathbb{E}_{s,a\sim d_{\tilde{\pi}}}\left[D_{TV}(M^\star(s,a),\hat{M}(s,a))\right],
\end{aligned}$$

where D_{TV} denotes the total variation distance between two distributions.

A.2. Proof of Lemma 3.5

Proof. We define two losses, one for each player:

$$\ell_{t+1}(M) = \mathbb{E}_{s,a\sim\tilde{\pi}_i}\left[D_{KL}(M^\star(s,a)||\hat{M}(s,a))\right] \tag{12}$$

$$\ell_{t+1}(\pi) = \frac{1}{H^2}\left(\mathbb{E}_{\xi\sim\pi_E,M^\star}\left[\sum_h^H \mathbb{E}_{a\sim\pi(s_h)}\left[Q^{\pi_t}_{M_{t+1},f_{t+1}}(s_h,a)\right]\right]\right) \tag{13}$$

Observe that minimizing the first involves an online convex optimization oracle over $\mathcal{M}$, and the second an online cost-sensitive classification oracle over Π. Also observe that, when summed, the policy and model losses bound the PDAM. To satisfy the ERRoR property, we need to be able to upper bound

$$\frac{1}{H}\sum_t^T J_{M^\star}(\pi_E, f_t) - J_{M^\star}(\pi_t, f_t). \tag{14}$$

Applying Corollary A.3 to the tth term in the summation, we get

$$\begin{aligned} J_{M^\star}(\pi_E, f_t) - J_{M^\star}(\pi_i, f_t) \leq \mathbb{E}_{\xi\sim\pi_E,M^\star}\left[\sum_h^H \mathbb{E}_{a\sim\pi_E(s_h)}[\hat{Q}_{f_t}(s_h,a)] - \mathbb{E}_{a\sim\pi(s_h)}[\hat{Q}_{f_t}(s_h,a)]\right] \\ + 2H^2\mathbb{E}_{s,a\sim d_{\tilde{\pi}_{t-1}}}\left[D_{\text{TV}}(M^\star(s,a), M_t(s,a))\right]. \end{aligned}$$

Recall that running NRPI for K iterations guarantees that $\ell_t(\pi_E) - \ell_t(\pi_t) \leq \bar{\epsilon}_\pi^K$, where π_t is the best of the K policy iterates generated (or their average). This directly implies that

$$J_{M^\star}(\pi_E, f_t) - J_{M^\star}(\pi_t, f_t) \leq \bar{\epsilon}_\pi^K H^2 + 2H^2\mathbb{E}_{s,a\sim d_{\tilde{\pi}_{t-1}}}\left[D_{\text{TV}}(M^\star(s,a), M_t(s,a))\right].$$

Via the definition of the regret of our M strategy, we have that

$$\min_{M\in\mathcal{F}_M}\frac{1}{T}\sum_t^T \ell_t(M_t) - \ell_t(M) \leq \bar{\epsilon}_M \Rightarrow \frac{1}{T}\sum_t^T \ell_t(M_t) \leq \bar{\epsilon}_M + \min_{M\in\mathcal{F}_M}\frac{1}{T}\sum_t^T \ell_t(M). \tag{15}$$

Clearly, $\ell_i(M^\star) = 0$ and recall that $M^\star \in \mathcal{M}$. Combining these facts with the non-negativity of the KL Divergence, we have that $\min_{M\in\mathcal{M}}\frac{1}{N}\sum_i^N \ell_i(M) = 0$. We can now apply Pinkser's and Jensen's inequalities to simplify the remaining terms:

$$\begin{aligned} &\frac{1}{T}\sum_{t=1}^T \mathbb{E}_{s,a\sim d_{\tilde{\pi}_{t-1}}}\left[D_{\text{KL}}(M^\star(s,a), M_t(s,a))\right] \leq \bar{\epsilon}_M \\ \Rightarrow &\frac{1}{T}\sum_{t=1}^T \mathbb{E}_{s,a\sim d_{\tilde{\pi}_{t-1}}}\left[D_{\text{TV}}(M^\star(s,a), M_t(s,a))^2\right] \leq \bar{\epsilon}_M \\ \Rightarrow &\frac{2H^2}{T}\sum_{t=1}^T \mathbb{E}_{s,a\sim d_{\tilde{\pi}_{t-1}}}\left[D_{\text{TV}}(M^\star(s,a), M_t(s,a))\right] \leq 2H^2\sqrt{\bar{\epsilon}_M}. \end{aligned}$$

Collecting terms gives us the desired bound. □

B. Implementation Details

We use Optimistic Adam (Daskalakis et al., 2017) for all policy and discriminator optimization, and gradient penalties (Gulrajani et al., 2017) to stabilize our discriminator training for all algorithms. Our policies, value functions, and discriminators are all 2-layer ReLu networks with a hidden size of 256. We sample 4 trajectories to use in the discriminator update at the end of each outer-loop iteration, and a batch size of 4096. In all 4 IRL variants (`HyPE`, `HyPER`, FILTER, MM), we re-label the data with the current reward function during policy improvement, rather than keeping the labels that were set when the data was added to the replay buffer, which we empirically observe to increase performance. This is different from standard IRL implementations and might be of independent interest.

B.1. MuJoCo Tasks

We detail below the specific implementations used for all MuJoCo experiments (Ant, Hopper, Humanoid, Walker). To clearly highlight the differences between our algorithms and the baselines, we enumerate separate sections for each.

Discriminator. For our discriminator, we start with a learning rate of $8e-4$ and decay it linearly over outer-loop iterations. For all model-free MuJoCo experiments, we update the discriminator every 10,000 actor steps. For model-based MuJoCo experiments, we update the discriminator every 2,000 actor steps for Hopper, and every 10,000 actor steps for Ant, Humanoid, and Walker.

Baselines. For IQLearn (Garg et al., 2021), we take the exact hyperparameters released in the original repository, but increase the expert memory size to be the same size as all other algorithms. For MuJoCo tasks, this is set to be 64,000 transition tuples. For MM and FILTER baselines, we follow the exact hyperparameters in Swamy et al. (2023) with a small modification of updating the discriminator every 10,000 actor steps instead of the 5,000 in the original repository. Finally, we train all behavioral cloning baselines for 300k steps for Ant, Hopper, and Humanoid, and 500k steps for Walker2d. For BC-Reg, we add the MSELoss to the usual SAC actor update, weighted by λ. We do a sweep over $\lambda = \{0.5, 1.0, 2.5, 5.0, 7.5, 10.0\}$ and take the λ that gives the highest average performance. We report these values in Table 1 for the four MuJoCo environments used.

Environment	λ
Ant-v3	1.0
Humanoid-v3	0.5
Hopper-v3	1.0
Walker2d-v3	0.5

Table 1: Final λ values for BC-Reg baseline.

HyPE. For `HyPE`, we use the Soft Actor Critic (Haarnoja et al., 2018) implementation provided by Raffin et al. (2019) with the hyperparameters in Table 2.

Parameter	Value
buffer size	1e6
batch size	256
γ	0.98
τ	0.02
Training Freq.	64
Gradient Steps	64
Learning Rate	Lin. Sched. 7.3e-4
policy architecture	256 x 2
state-dependent exploration	true
training timesteps	1e6

Table 2: Hyperparameters for `HyPE` using SAC.

HyPER. For `HyPER`, we use the implementation from Pineda et al. (2021) with modifications on the actor update according to Vemula et al. (2023), and turn on the entropy bonus. The model is provided the same demonstration dataset of 64,000

transition tuples as model-free experiments. We use an ensemble of discriminators equal to the model ensemble size, and take the minimum value of the ensemble each forward pass. Further, we find that clipping the discriminator output, adding the same weight decay to the model and actor networks, and using an exponential moving average of the actor weights during inference helps stabilize performance for some environments. We list the specific hyperparameters used for each environment in Tables 3 to 6.

B.2. D4RL Tasks

For the two `antmaze-lage` tasks, we use the data provided by Fu et al. (2020) as out expert demonstrations. We append goal information to the observation for all algorithms following the example in Swamy et al. (2023). For our policy optimizer in every algorithm other than IQLearn, we build upon the TD3+BC implementation of Fujimoto & Gu (2021) with the default hyperparameters.

Discriminator. For our discriminator, we start with a learning rate of $8e-3$ and decay it linearly over outer-loop iterations. For all model-free Antmaze experiments, we update the discriminator every 5,000 actor steps. For all model-based Antmaze experiments, we update the discriminator every 2,000 actor steps.

Baselines. For behavioral cloning, we run the TD3+BC optimizer for 500k steps while zeroing out the component of the actor update that depends on rewards. We use $\alpha = 1.0$ for FILTER. We provide all baselines with the same data provided to `HyPE` and `HyPER` consisting of the entire D4RL dataset for both `antmaze-large` environments. MM and FILTER are pretrained with 10,000 steps of behavioral cloning on the expert dataset.

HyPE. Both `HyPE` and `HyPE+Resets` use the same TD3+BC optimizer and hyperparameters for the actor as MM and FILTER, and is pretrained with 10,000 steps of behavioral cloning. `HyPE+Resets` uses $\alpha = 1.0$ to always reset to expert states.

HyPER. To stabilize performance for `HyPER`, we pretrain the model on the offline dataset. In addition to the hyperparameters from previous algorithms, we use the exponential moving average of the actor weights and add a CosineAnnealing decay on both the actor and critic. Within the learned model, we perform n-step updates backwards in time as inspired by Bagnell et al. (2003) and Hester et al. (2018b) by resetting to a sliding window of expert states within the learned model. Specifically, if T is the total number of training steps and H the horizon of the environment, then for each iteration t we reset to the set of expert states falling within a sliding window of size $\kappa \in [0, 1]$:

$$\left[H \cdot \left(1 - \frac{i}{T}\right), H \cdot \min\left(1, 1 - \frac{i}{T} + \kappa\right)\right].$$

We set $\kappa = 0.05$ in practice. Additional details and visualizations can be found in Appendix C. Finally, we provide the model-based hyperparameters for both `antmaze-large` environments in Table 7.

C. Antmaze Model Pretraining

We visualize below the impact of offline dataset size when pretraining the model for `antmaze-large`. In the leftmost graph of Figure 6, we plot the start and goal distributions for `antmaze-large-play` in orange and red respectively. In the following three plots, we show the state visitation frequency taking various proportions from the offline data. Notably, with fewer samples, there are regions of the maze with extremely low data coverage, such as the bottom left corner. A model trained on 10k or 25k samples may therefore learn inaccurate dynamics in those regions, leading to unreliable transition dynamics and thus unreliable policy in those regions. Thus by taking 80k samples to pretrain the model and decreasing model update frequency, we ensure that the learner is able to receive sufficiently accurate transition tuples in training.

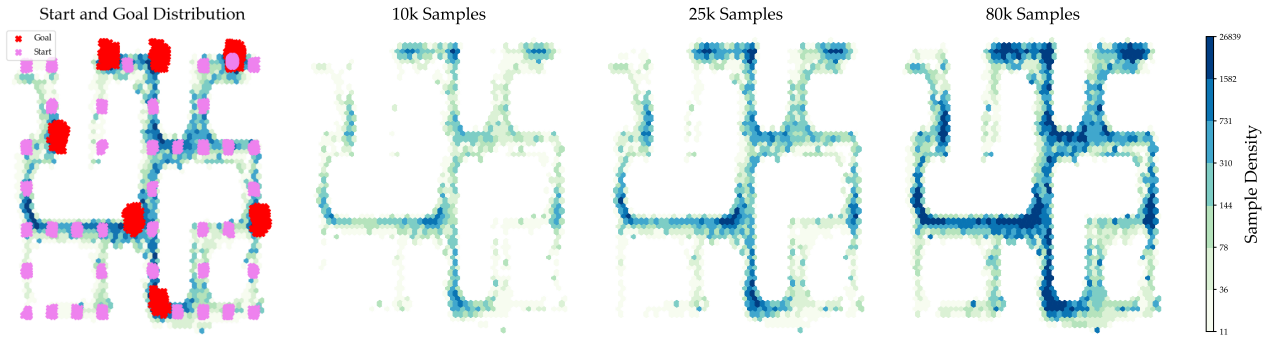

Figure 6: The leftmost plot shows the start and goal distributions of the `antmaze-large-play` environment. In the subsequent three plots, we show the state visitation frequency across the maze with varying number of samples from the offline dataset. We find a sufficiently large enough sample is necessary to ensure accurate transition tuples in training.

D. Additional Ablations

D.1. Classical Adversarial Imitation Learning Methods

Our inverse RL baseline MM can be thought of as a significantly improved variant of the classical methods such as GAIL (Ho & Ermon, 2016). More explicitly, we implement the four changes to GAIL suggested by (Swamy et al., 2022) which, when combined, lead to a significant improvement over off-the-shelf implementations of GAIL. Explicitly, these are 1) using a Wasserstein GAN loss rather than the original GAN loss, 2) using SAC instead of PPO, 3) using gradient penalties in the loss function, and 4) using the Optimistic Adam Optimizer. We would like to point readers to Appendix C of Swamy et al. (2022) for ablations on each of these components. To ablate the joint benefits of these modifications, we compare our MM baseline to GAIL (as implemented in https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, a popular implementation with more than 3.5k Github stars) on the Humanoid environment, and report the average performance over 10 seeds. Figure 7 shows that while GAIL achieves some improvement over 1 million environment interactions, it massively under performs MM, reaffirming the fact that MM is a very strong baseline for standard IRL in and of itself.

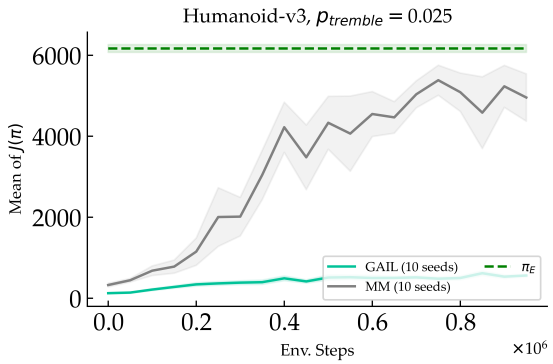

Figure 7: Comparison of our MM baseline against GAIL (Ho & Ermon, 2016), reporting the average and standard error over 10 seeds. While GAIL make gradual improvement over 1 million environment interactions, it is far from the performance of MM.

D.2. Low Data Regime

We present in Figure 8 an ablation of the performance of various algorithms in the low-data regime. Specifically, we take just 5 trajectories to train each algorithm, as opposed to the 64 used in the main experiments. We observe that HyPE still has the quickest convergence over other model-free IRL baselines, even in the low-data regime. HyPER performs rather poorly in the low-data regime, which we hypothesize is due to the difficulty of learning an accurate model from such limited data. However, given a world model can be learned from other data sources which might be more plentiful in practice (e.g. suboptimal demonstrations, multi-task demonstrations, and even robot play data), we believe there are multiple remedies to this issue in practice.

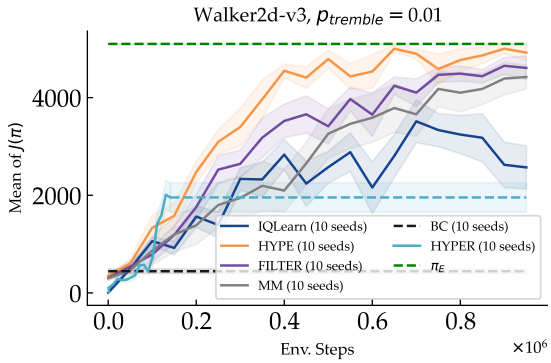

Figure 8: Mean and standard error of various algorithms when provided only 5 expert demonstrations over 10 seeds. `HyPE` still outperforms existing baselines. `HyPER` suffers a performance drop, which is expected as low amounts of expert data may not be sufficient for fitting a good model. We believe there are various ways to improve model fitting in practice in the presence of few expert demonstrations, such as multi-task or robot play data.

Parameter	Value
Expert Dataset Size	64000
Exploration Sample Size	64000
Model Ensemble Size	7
Model Ensemble Elite Number	5
Model Learning Rate	3e-4
Model Weight Decay	5e-5
Model Batch Size	256
Model Train Frequency	125
Model Hidden Dims Size	400
Model Clip Output	True
Discriminator Clip Output	True
Discriminator Weight Decay	True
Discriminator Ensemble Size	7
Schedule Model LR	False
Schedule Policy LR	False
EMA policy weights	False
Number policy updates per step	30
Policy updates every steps	1
Rollout step in learned model	400
Rollout Length	$1 \rightarrow 25$
Policy Type	Stochastic Gaussian Policy

Table 3: Hyperparameters for `Ant-v3`.

Parameter	Value
Expert Dataset Size	64000
Exploration Sample Size	64000
Model Ensemble Size	7
Model Ensemble Elite Number	5
Model Learning Rate	3e-4
Model Weight Decay	5e-5
Model Batch Size	256
Model Train Frequency	125
Model Hidden Dims Size	400
Model Clip Output	False
Discriminator Clip Output	False
Discriminator Weight Decay	False
Discriminator Ensemble Size	7
Schedule Model LR	False
Schedule Policy LR	False
EMA policy weights	False
Number policy updates per step	30
Policy updates every steps	1
Rollout step in learned model	400
Rollout Length	$1 \rightarrow 25$
Policy Type	Stochastic Gaussian Policy

Table 4: Hyperparameters for `Hopper-v3`.

Parameter	Value
Expert Dataset Size	64000
Exploration Sample Size	64000
Model Ensemble Size	7
Model Ensemble Elite Number	5
Model Learning Rate	3e-4
Model Weight Decay	5e-5
Model Batch Size	256
Model Train Frequency	125
Model Hidden Dims Size	400
Model Clip Output	False
Discriminator Clip Output	False
Discriminator Weight Decay	False
Discriminator Ensemble Size	7
Schedule Model LR	False
Schedule Policy LR	True
EMA policy weights	True
Number policy updates per step	20
Policy updates every steps	2
Rollout step in learned model	400
Rollout Length	$1 \rightarrow 25$
Policy Type	Stochastic Gaussian Policy

Table 5: Hyperparameters for `Humanoid-v3`.

Parameter	Value
Expert Dataset Size	64000
Exploration Sample Size	1000
Model Ensemble Size	7
Model Ensemble Elite Number	5
Model Learning Rate	3e-4
Model Weight Decay	5e-5
Model Batch Size	256
Model Train Frequency	125
Model Hidden Dims Size	200
Model Clip Output	True
Discriminator Clip Output	True
Discriminator Weight Decay	True
Discriminator Ensemble Size	7
Schedule Model LR	True
Schedule Policy LR	True
EMA policy weights	True
Number policy updates per step	20
Policy updates every steps	2
Rollout step in learned model	400
Rollout Length	$1 \rightarrow 25$
Policy Type	Stochastic Gaussian Policy

Table 6: Hyperparameters for `Walker-v3`.

Parameter	Value
Expert Dataset Size	999000
Exploration Sample Size	10000
Model Ensemble Size	7
Model Ensemble Elite Number	5
Model Learning Rate	3e-4
Model Weight Decay	5e-5
Model Batch Size	256
Model Train Frequency	1000
Model Hidden Dims Size	200
Model Clip Output	False
Discriminator Clip Output	False
Discriminator Weight Decay	False
Discriminator Ensemble Size	1
Schedule Model LR	True
Schedule Policy LR	True
EMA policy weights	True
Number policy updates per step	20
Policy updates every steps	1
Rollout step in learned model	400
Rollout Length	$1 \rightarrow 25$
Policy Type	Stochastic Gaussian Policy

Table 7: Hyperparameters for `antmaze-large`.