Hybrid Inverse Reinforcement Learning

Juntao Ren *
Cornell University

Gokul Swamy *
Carnegie Mellon University

Zhiwei Steven Wu
Carnegie Mellon University

J. Andrew Bagnell
Aurora Innovation and Carnegie Mellon University

Sanjiban Choudhury
Cornell University

Abstract

The inverse reinforcement learning approach to imitation learning is a double-edged sword. On one hand, it allows the learner to find policies that are robust to compounding errors. On the other hand, it requires that the learner repeatedly solve a computationally expensive reinforcement learning (RL) problem. Often, much of this computation is spent searching over policies dissimilar to the expert's and is therefore wasted. In this work, we propose using *hybrid* reinforcement learning to curtail this unnecessary exploration by leveraging the fact that rather than computing the optimal policy for each adversarially chosen reward function, we merely need to compete with the expert. More formally, we derive a reduction from inverse RL to hybrid RL that allows us to dramatically reduce interaction during the inner policy search loop while still maintaining a degree of robustness to compounding errors. Empirically, we find that our approaches are far more sample efficient than standard inverse RL and several other baselines that require stronger assumptions (e.g. generative model access) on a suite of continuous control tasks.

1 Introduction

Broadly speaking, we can break down the approaches to imitation learning into *offline* algorithms (e.g. behavioral cloning, [1]) and *interactive* algorithms (e.g. inverse reinforcement learning [2], DAgger [3]). The key benefit of interaction in sequential tasks is that it allows the learner to observe the consequences of their actions and therefore learn to recover from their own mistakes. More formally, *all* offline approaches to imitation aren't robust to the covariate shift between the expert's state distribution and the learner's *induced* state distribution and therefore can suffer from compounding errors and poor test-time performance [3, 4]. More colloquially, *what you see is what you get*. This is the fundamental reason that for safety-critical tasks like autonomous driving [5, 6, 7] and robust real-world services like Google Maps [8], interactive approaches like inverse reinforcement learning continue to provide state-of-the-art performance

Of course, any interactive approach necessarily requires that the learner is able to try out actions in the environment they will be tested on (or at least in an accurate simulator of it). This interaction can be unsafe when performed in the real world and of great computational expense when performed in simulation. In fact, because approaches like inverse reinforcement learning reduce the problem of imitation to repeatedly solving a reinforcement learning problem, they can end up having to pay for the exponential interaction complexity of reinforcement learning [9] over and over again. This motivates our key question: *what is the minimal amount of interaction we can perform in imitation learning while still maintaining robustness to compounding errors?*

*Equal contribution. Correspondence to gswamy@cmu.edu.

NeurIPS 2023 Workshop on Robot Learning: Pretraining, Fine-Tuning, and Generalization with Large Scale Models, New Orleans, United States of America

If one was to profile a standard inverse reinforcement learning procedure, they would notice that the majority of computation isn't spent in the *outer loop* picking a reward function (which is essentially a classification problem) and is instead spent in the *inner loop*, optimizing the chosen reward function (which is solving a reinforcement learning problem). In essence, the reason reinforcement learning is computationally expensive is because of *exploration*: the learner needs to try out all possible actions in all possible states to figure out what decisions are optimal over the horizon. This means that in inverse reinforcement learning, the learner often spends the majority of their computational budget trying out policies that are quite dissimilar to the expert's in the hope of finding a bit of reward. If we take a step back, this is rather odd: given our goal is to imitate the expert, we clearly don't need to be thinking about policies that are nothing like it. Put differently, when optimizing a potential reward function, we should only be competing against policies with similar visitation distributions to the expert. We can therefore narrow our overarching question to the following: *how do we focus our policy search on policies that are similar to the expert's?*

Recent work by [10] has proposed one answer to this question: by resetting the learner to states from the expert demonstrations during policy search, one can scope down the amount of exploration the learner does. While this approach comes with strong theoretical guarantees (e.g. polynomial interaction complexity), it requires the ability to reset the learner to arbitrary states. However, for a variety of tasks in the real world, we might not be able to reset our learner to an arbitrary state. For example, for agile robotics, resetting a quadruped to the apex of a back-flip is rather challenging. We therefore focus on how we can curtail unnecessary exploration *without* assuming generative model access to the environment.

We propose to use *hybrid* reinforcement learning [11, 12, 13] to speed up the policy search component of inverse reinforcement learning. In online RL, one trains a policy that does well on the distribution of data it induces. In offline RL, one trains a policy that does well on an offline dataset. In hybrid RL, one trains a policy to do well on *both* (e.g. by using data from both buffers when fitting a Q-function). The upshot of this balanced training procedure is that rather than competing against an arbitrary policy (as we do in online RL and therefore need to pay for extensive exploration), we only ask the learner to compete against policies *covered* by the offline dataset. With hybrid RL, we get the same theoretical guarantees as offline RL [12] but do not require implementing explicit pessimism (which can be finicky and intractable theoretically) while retaining a higher degree of robustness to covariate shift as our learner gets to observe their own trajectories. Given our goal is to compete with the expert, we propose to simply use the expert demonstrations as the offline dataset for hybrid RL. In short, our key insight is that ***we can use hybrid RL as the policy search procedure in IRL to curtail exploration while maintaining some degree of robustness to compounding errors.*** More explicitly, the contributions of our work are two-fold:

1. We provide a hybrid inverse RL (`HyPE`) algorithm and discuss its performance guarantees. We show that while in the worst case, `HyPE` can suffer from compounding errors, it is more robust to them than purely offline approaches like behavioral cloning.

2. We demonstrate that on continuous control tasks, `HyPE` is significantly more sample efficient than standard IRL. In addition to out-performing standard IRL and behavioral cloning, we also find that on some tasks, we are able to sometimes out-perform and never significantly under-perform the `FILTER` algorithm of [10], which requires the ability to reset the learner to arbitrary states.

We begin by formalizing our method before moving onto experimental results.

2 Designing a Hybrid Inverse Reinforcement Learning Algorithm

We consider a finite-horizon Markov Decision Process (MDP) [14] parameterized by $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, T \rangle$ 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 T 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 [4]. More formally, we optimize over policies $\pi : \mathcal{S} \to \Delta(\mathcal{A}) \in \Pi$ and reward functions $f : \mathcal{S} \times \mathcal{A} \to [-1, 1] \in \mathcal{F}_r$. For theoretical simplicity, we assume that our strategy spaces (Π and $\mathcal{F}_r$) are convex and compact, that $\mathcal{F}_r$ is closed

Algorithm 1 Hybrid Policy Emulation (HyPE)

Input: Demos. $\mathcal{D}_E$, Policy class Π, Reward class $\mathcal{F}_r$, Hybrid RL oracle HyRL
Output: Trained policy π
Initialize $f_1 \in \mathcal{F}_r$
for i in $1 \dots N$ **do**
  // Take a no-regret step over rewards (e.g. via OGD)
  $f_{i+1} \leftarrow f_i + \eta \nabla_f (J(\pi_i, f) - J(\pi_E, f))$
  // Perform an expert-competitive response over policies
  $\pi_{i+1} \leftarrow \texttt{HyRL}(\mathcal{D} = \mathcal{D}_E, \hat{r} = -f_j)$
end for
Return π_i with the lowest validation error.

under negation, and that $r \in \mathcal{F}_r, \pi_E \in \Pi$. We assume access to an efficient oracle that fulfills the following contract of returning a policy that competes with those covered by an offline dataset:

Definition 2.1 (Dataset-Competitive Response DCR[ϵ])**.** Given a dataset of trajectories $\mathcal{D}$ and a reward function $\hat{r}$, DCR($\mathcal{D}$, $\hat{r}$) returns a policy π such that

$$J(\pi^*, \hat{r}) - J(\pi, \hat{r}) \leq \epsilon T^2, \tag{1}$$

where ϵ is a constant, T is the horizon of the problem, and π^* is the optimal policy covered by the offline dataset, i.e.

$$\pi^* \in \text{argmax}\, J(\pi, \hat{r}) \text{ s.t. } \left\| \frac{\rho_\pi}{\rho_\mathcal{D}} \right\|_\infty < C, \tag{2}$$

where ρ_π refers to the state-action visitation distribution of π and C is some constant.

For simplicity, we ignore finite sample issues and errors introduced by function approximation and therefore set $C = \infty$. Critically, various hybrid RL algorithms essentially provide the above guarantee. (e.g. HyQ [12] or HAC / HNPG [13]). For our use of this oracle, we will set $\rho_\mathcal{D} = \rho_E$, providing an *expert-competitive response*, which we denote hereafter as ECR[ϵ].

A key detail in the above definition is that rather than guaranteeing it will return a policy that is covered by the offline dataset, hybrid RL instead guarantees we will return a policy that *competes* with any policy covered by the offline data. The reason this distinction matters is that we cannot simply treat the above oracle as performing a "best-response" over the covered policy class, which would allow us to plug in ECR into the standard setup for a *dual* inverse RL algorithm [4]. Instead, we generalize the proof strategy of [10] to allow for an *arbitrary* ECR (rather than the specific algorithm, NRPI [15], they use). We then use hybrid RL to provide this ECR, improving the computational efficiency of our procedure without requiring the ability to reset the learner to an arbitrary state.

We outline our full approach in Algorithm 1. We prove the following performance guarantee on the policies returned by HyPE in Appendix A.

Theorem 2.2 (HyPE Performance Guarantee)**.** *Assume we run HyPE with a hybrid RL oracle that satisfies ECR[ϵ]. Let $\pi_1, \dots, \pi_N$ denote the sequence of policies we compute and $\bar{\pi}$ their average. Also, let $f_1, \dots, f_N$ denote the sequence of rewards we compute and $\bar{\epsilon}$ their average regret. Then, we have that*

$$J(\pi_E, r) - J(\bar{\pi}, r) \leq \bar{\epsilon} T + \epsilon T^2 \tag{3}$$

2.1 HyPE vs. Behavioral Cloning.

The above bound indicates that in the worst case, policies produced by HyPE can suffer from compounding errors, in contrast to those produced by standard inverse RL [4]. However, this worst-case analysis hides the fact that on less contrived problems, HyPE will perform better than a purely offline approach like behavioral cloning that do not observe the consequences of their own chosen actions. We illustrate this fact with a simple example.

Example 2.3. Consider the following two-lane MDP. At each timestep, the agent must move right. At each timestep, they can either stay in the same lane or switch lanes. The goal is to stay in the top lane, which provides a constant reward of $+1$. We see data from an optimal expert.

The learner can chose between two deterministic policies (teal and orange) that share the black prefix and then diverge. We visualize a rollout from each of these policies in the below figure. For $s_{3:T}$ in the top row, both policies always take the action to stay in the top row.

Figure 1: A simple two-lane problem where the learner's goal is to stay in the top lane and they are able to choose between the orange and teal policies. HyPE does better than offline methods here.

We can directly calculate that $J(\pi, r) = T - 1$ and $J(\pi, r) = 1$. However, both policies have an equivalent behavioral cloning loss – i.e. for the zero/one loss, $\ell_{BC}(\pi) = \mathbb{E}_{s_E, a_E \sim \rho_E}[1(a_E \neq \pi(s_E)]$, $\ell_{BC}(\pi) = \ell_{BC}(\pi) = \frac{1}{T}$. Thus, for a purely offline method, we wouldn't be able to break ties between these policies. Let's say our reward class is $\mathcal{F}_r = \{r, 1 - r\}$. Then, regardless of the policy we picked, the adversary would always play r. For any $\epsilon < \frac{T-1}{T^2}$, an ECR$[\epsilon]$ must return the orange policy, which means an algorithm like HyPE would always return the best policy in our class.

3 Experimental Results

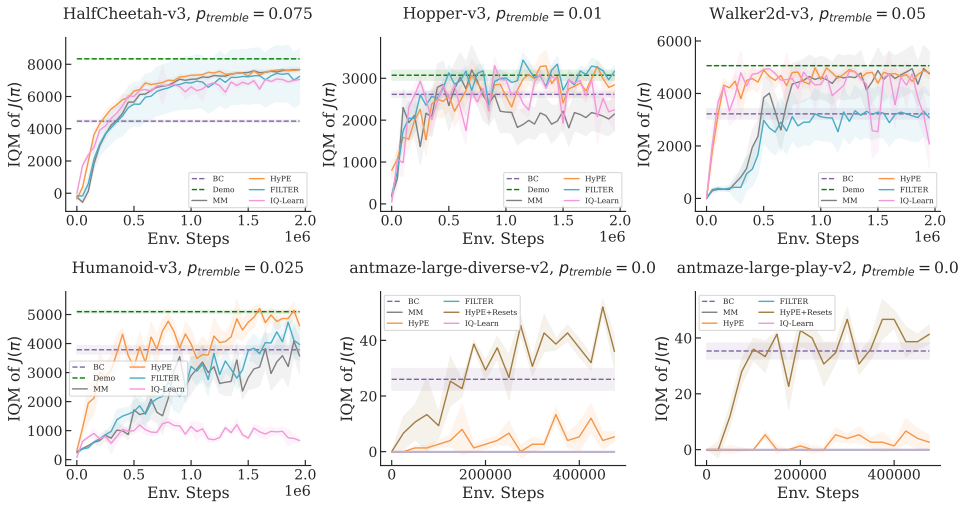

Figure 2: On all tasks considered, HyPE uniformly achieves the highest reward. Further performance gap between HyPE and other algorithms increases as environments increase in difficultly. We plot the interquartile mean as recommended by [16]. Standard errors are computed across 5 seeds.

We conduct experiments with the Mujoco benchmark suite [17]. We train experts using Soft Actor Critic [18] and then present all learners with 64 expert demonstrations. As a simple behavioral cloning baseline matches expert performance under these conditions [4], we harden the problem by introducing randomization: with probability $p_{tremble}$, a random action gets executed in the environment rather than the one the policy chose. Our expert data is free from these corruptions. We also present results on the antmaze-large tasks from [19], but with $p_{tremble} = 0$.

We compare five algorithms: behavioral cloning [1], IQLearn [20] (an interactive approach that eliminates the inner loop of inverse RL at the cost of a secondary regression), standard inverse reinforcement learning (using techniques like gradient penalties and decaying learning rates as suggested by [21] to improve performance over naive implementations), FILTER [10], and HyPE. In all three of the IRL variants, because it leads to significantly better performance, we re-label data with the current reward function during policy improvement rather than keeping the reward

labels that were set when data was added to the replay buffer. We note that this differs from the standard implementation of IRL [4, 21, 10, 22] and might be of independent interest. Essentially, both FILTER and HyPE can be seen as extensions of the standard IRL setup where we perform an expert-competitive response rather than a true best response. In FILTER, we perform an ECR by resetting the learner to states from the expert demonstrations. In HyPE, we perform an ECR by using a batch that is a 50/50 mixture of learner and expert data when performing policy updates.

In Figure 2, we see that across the board, HyPE does no worse than its next closest competitor, significantly out-performing techniques that require stronger assumptions like FILTER. Importantly, we also see that constraining the policy space of the learner never leads to worse performance compared to standard IRL (MM in the figures). We also consistently out-perform a behavioral cloning baseline, highlighting the importance of interaction for mitigating compounding errors. We observe that the performance of IQLearn grows increasingly unstable and deteriorates as the difficulty of the environment increases, while HyPE consistently achieves the best performance across environments. Finally, we find using both resets and a mixed batch is important for performance on the two antmaze environments. We hypothesize that for problems with hard exploration, using both techniques is critical to curtail unnecessary exploration. In summary, our experimental results show that HyPE is able to inherit the benefits of interaction without having to pay for the full price of unstructured exploration, which agrees with our theory.

References

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

[2] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2008.

[3] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 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.

[4] Gokul Swamy, Sanjiban Choudhury, J. Andrew Bagnell, and Zhiwei Steven Wu. Of moments and matching: A game-theoretic framework for closing the imitation gap. *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[5] Eli Bronstein, Mark Palatucci, Dominik Notz, Brandyn White, Alex Kuefler, Yiren Lu, Supratik Paul, Payam Nikdel, Paul Mougin, Hongge Chen, et al. Hierarchical model-based imitation learning for planning in autonomous driving. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8652–8659. IEEE, 2022.

[6] Maximilian Igl, Daewoo Kim, Alex Kuefler, Paul Mougin, Punit Shah, Kyriacos Shiarlis, Dragomir Anguelov, Mark Palatucci, Brandyn White, and Shimon Whiteson. Symphony: Learning realistic and diverse agents for autonomous driving simulation. *arXiv preprint arXiv:2205.03195*, 2022.

[7] Eugene Vinitsky, Nathan Lichtlé, Xiaomeng Yang, Brandon Amos, and Jakob Foerster. Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world. *arXiv preprint arXiv:2206.09889*, 2022.

[8] Matt Barnes, Matthew Abueg, Oliver F. Lange, Matt Deeds, Jason Trader, Denali Molitor, Markus Wulfmeier, and Shawn O'Banion. Massively scalable inverse reinforcement learning in google maps, 2023.

[9] Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

[10] Gokul Swamy, Sanjiban Choudhury, J. Andrew Bagnell, and Zhiwei Steven Wu. Inverse reinforcement learning without reinforcement learning. *ArXiv*, abs/2303.14623, 2023.

[11] Stephane Ross and J Andrew Bagnell. Agnostic system identification for model-based reinforcement learning. *arXiv preprint arXiv:1203.1007*, 2012.

[12] Yuda Song, Yifei Zhou, Ayush Sekhari, J Andrew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid rl: Using both offline and online data can make rl efficient. *arXiv preprint arXiv:2210.06718*, 2022.

[13] Yifei Zhou, Ayush Sekhari, Yuda Song, and Wen Sun. Offline data enhanced on-policy policy gradient with provable guarantees. *arXiv preprint arXiv:2311.08384*, 2023.

[14] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[15] Stéphane Ross and J. Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *ArXiv*, abs/1406.5979, 2014.

[16] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

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

[18] Tuomas Haarnoja, Aurick Zhou, P. Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

[19] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

[20] Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. Iq-learn: Inverse soft-q learning for imitation. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[21] Gokul Swamy, Nived Rajaraman, Matt Peng, Sanjiban Choudhury, Drew Bagnell, Steven Wu, Jiantao Jiao, and Kannan Ramchandran. Minimax optimal online imitation learning via replay estimation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[22] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *30th Conference on Neural Information Processing Systems*, 2016.

[23] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2017.

[24] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[25] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3, 2019.

[26] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

A Proofs

We begin by stating our key proof technique for general zero-sum games before specializing to inverse RL.

A.1 Two Player Zero-Sum Games with Relative Best Responses

Theorem A.1. *Consider a two-player zero-sum game $\max_{x\in\mathcal{X}} \min_{y\in\mathcal{Y}} \ell(x,y)$ with payoff ℓ that is concave in x and convex in y. Given access to a no-regret online convex optimization algorithm over $\mathcal{X}$ and a relative best response oracle over $\mathcal{Y}$ that guarantees to output a y s.t. $\ell(y,x) - \ell(y_E,x) \leq \epsilon$, we are able to compute an average iterate $\bar{y}$ in T rounds such that*

$$\max_{x\in\mathcal{X}} \ell(x,\bar{y}) - \ell(x,y_E) \leq \epsilon + \frac{\mathit{Reg}_{\mathcal{X}}(T)}{T}. \tag{4}$$

Proof. By the convexity of ℓ in y and Jensen's inequality, we have that

$$\max_{x\in\mathcal{X}} \ell(x,\bar{y}) - \ell(x,y_E) \leq \frac{1}{T}\sum_{t=1}^{T} \ell(x,y_t) - \ell(x,y_E). \tag{5}$$

Define $\ell_t(x) = \ell(x,y_t) - \ell(x,y_E)$. Thus, to prove our original claim, it is sufficient to upper-bound

$$\max_{x\in\mathcal{X}} \frac{1}{T}\sum_{t=1}^{T} \ell_t(x). \tag{6}$$

From our relative best response, we directly have that

$$\forall t \in [T], \ell(x_t,y_t) - \ell(x_t,y_E) \leq \epsilon \Rightarrow \forall t \in [T], \ell_t(x_t) \leq \epsilon. \tag{7}$$

By the definition of regret of our x selection strategy, we have that

$$\max_{x\in\mathcal{X}} \frac{1}{T}\sum_{t=1}^{T} \ell_t(x) - \ell_t(x_t) \leq \frac{\mathrm{Reg}_{\mathcal{X}}(T)}{T}. \tag{8}$$

Rearranging terms, this tells us that

$$\max_{x\in\mathcal{X}} \frac{1}{T}\sum_{t=1}^{T} \ell_t(x) \leq \frac{\mathrm{Reg}_{\mathcal{X}}(T)}{T} + \frac{1}{T}\sum_{t=1}^{T} \ell_t(x_t) \leq \frac{\mathrm{Reg}_{\mathcal{X}}(T)}{T} + \epsilon. \tag{9}$$

where the last inequality comes from the definition of the relative best response. This completes the proof. We note that via the no-regret property of our x selection strategy, the first term tends to 0 as $N \to \infty$. □

A.2 Proof of Theorem 2.2

Theorem A.2 (HyPE Performance Guarantee)**.** *Assume we run HyPE with a hybrid RL oracle that satisfies ECR[ϵ]. Let $\pi_1,\ldots,\pi_N$ denote the sequence of policies we compute and $\bar{\pi}$ their average. Also, let $f_1,\ldots,f_N$ denote the sequence of rewards we compute and $\bar{\epsilon}$ their average regret. Then, we have that*

$$J(\pi_E,r) - J(\bar{\pi},r) \leq \bar{\epsilon}T + \epsilon T^2 \tag{10}$$

Proof. We essentially follow the same proof structure as our above general template, with $\mathcal{X}$ mapping to $\mathcal{F}_r$ and the relative best-response oracle mapping to ECR[ϵ].

By Jensen's inequality, we have that

$$J(\pi_E,r) - J(\bar{\pi},r) \leq \max_{f\in\mathcal{F}_r} J(\pi_E,f) - J(\bar{\pi},f) \leq \max_{f\in\mathcal{F}_r} \sum_{i=1}^{N} J(\pi_E,f) - J(\pi_i,f). \tag{11}$$

Notice that $\forall i \in [N]$, the ECR[ϵ]-property of our hybrid reinforcement learning oracle guarantees that

$$J(\pi_E, -f_i) - J(\pi_i, -f_i) = J(\pi_i, f_i) - J(\pi_E, f_i) \leq \epsilon T^2 \tag{12}$$

Define the loss we feed to our no-regret reward learning algorithm as $\ell_i(f) = \frac{1}{T}(J(\pi_i, f) - J(\pi_E, f)) \in [-1, 1]$. Then, by the definition of regret of our reward-selection strategy, we have that

$$\max_{f^* \in \mathcal{F}_r} \frac{1}{N} \sum_{i=1}^{N} (J(\pi_i, f^*) - J(\pi_E, f^*)) - (J(\pi_i, f_i) - J(\pi_E, f_i)) \leq \bar{\epsilon} T, \tag{13}$$

Adding the second term to both sides, we get that

$$\max_{f^* \in \mathcal{F}_r} \frac{1}{N} \sum_{i=1}^{N} (J(\pi_i, f^*) - J(\pi_E, f^*)) \leq \frac{1}{N} \sum_{i=1}^{N} (J(\pi_i, f_i) - J(\pi_E, f_i)) + \bar{\epsilon} \tag{14}$$

$$\leq \epsilon T^2 + \bar{\epsilon} T, \tag{15}$$

where the last inequality comes from the ECR[ϵ] property. This completes the proof of the desired claim. We note that via the no-regret property of our reward-selection strategy, the first term tends to 0 as $N \to \infty$. □

B Implementation Details

We use Optimistic Adam [23] for all policy and discriminator optimization, and gradient penalties [24] 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. For our discriminator, we start with a learning rate of $8e-4$ and decay it linearly over outer-loop iterations.

B.1 Mujoco Tasks

For the Mujoco Tasks (HalfCheetah, Hopper, Walker, and Humanoid), we use the Soft Actor Critic [18] implementation provided by [25] for policy optimization for both the expert and the learner. We use the hyperparameters in Table 1 for all experiments. We train behavioral cloning for 300,000 steps.

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 1: Expert and learner hyperparameters for SAC.

We use the best response variant of FILTER from [10] with $\alpha = 0.5$. Each outer loop iteration lasts for 10000 steps of environment interaction.

B.2 D4RL Tasks

For the two antmaze-lage tasks, we use the data provided by [19] as out expert demonstrations. We append goal information to the observation for all algorithms following the example in [10].

For our policy optimizer, we build upon the TD3+BC implementation of [26] with the default hyperparameters. 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. For MM, FILTER, and HyPE, we pretrain the policy with 10,000 steps of behavioral cloning. We use $\alpha = 1.0$ for FILTER.

Each outer loop iteration lasts for 5000 steps of environment interaction.