
HYBRID-AIRL: ENHANCING INVERSE REINFORCEMENT LEARNING WITH SUPERVISED EXPERT GUIDANCE

A PREPRINT

Bram Silue¹, **Santiago Amaya-Corredor¹**, **Patrick Mannion²**, **Lander Willem³**, and **Pieter Libin¹**

¹Artificial Intelligence Lab — Vrije Universiteit Brussel

²College of Science and Engineering — University of Galway

³Family Medicine and Population Health — University of Antwerp

November 27, 2025

ABSTRACT

Adversarial Inverse Reinforcement Learning (AIRL) has shown promise in addressing the sparse reward problem in reinforcement learning (RL) by inferring dense reward functions from expert demonstrations. However, its performance in highly complex, imperfect-information settings remains largely unexplored. To explore this gap, we evaluate AIRL in the context of Heads-Up Limit Hold’em (HULHE) poker, a domain characterized by sparse, delayed rewards and significant uncertainty. In this setting, we find that AIRL struggles to infer a sufficiently informative reward function. To overcome this limitation, we contribute Hybrid-AIRL (H-AIRL), an extension that enhances reward inference and policy learning by incorporating a supervised loss derived from expert data and a stochastic regularization mechanism. We evaluate H-AIRL on a carefully selected set of Gymnasium benchmarks and the HULHE poker setting. Additionally, we analyze the learned reward function through visualization to gain deeper insights into the learning process. Our experimental results show that H-AIRL achieves higher sample efficiency and more stable learning compared to AIRL. This highlights the benefits of incorporating supervised signals into inverse RL and establishes H-AIRL as a promising framework for tackling challenging, real-world settings.

Keywords Inverse Reinforcement Learning · Supervised Learning · Generative Adversarial Networks · Poker

1 Introduction

Deep reinforcement learning (RL) has recently demonstrated remarkable performance in complex tasks, including robotic control, epidemic control, and video games [1, 2, 3, 4]. However, designing effective reward functions remains a critical challenge. In many settings, manually designed reward functions are often characterized by sparse or delayed feedback, which can hinder learning.

A compelling example of these challenges is found in strategic games, as their complex dynamics create a highly demanding learning environment. Poker, as a representative imperfect-information game, poses a particularly demanding scenario. In poker, the reward signal is inherently sparse, as feedback is generally available only at the conclusion of each hand, which restricts the information available for learning effective strategies. Historically, superhuman poker agents have been developed using Counterfactual Regret Minimization (CFR) [5], a method that iteratively converges toward a Nash equilibrium and guarantees optimal play. Despite its theoretical guarantees, CFR is computationally intensive and demands significant resources [6].

Reinforcement learning offers an attractive alternative by enabling agents to learn strategies directly from experience [7]. Yet, conventional RL approaches targeted at poker struggle with the inherent sparsity of the reward signal, which impedes the agent’s ability to capture nuanced expert strategies. In this context, Inverse Reinforcement Learning (IRL)

offers a promising approach by aiming to recover a dense reward function from expert demonstrations. Rather than merely imitating the expert, IRL seeks to uncover the underlying motivations driving expert behavior [8].

Early IRL methods leveraged the maximum-entropy principle to model expert behavior, laying the foundation for probabilistic formulations that capture variability in demonstrations [9]. Building on these ideas, adversarial formulations such as Adversarial Inverse Reinforcement Learning (AIRL) have extended the approach by jointly recovering the reward function and policy [10]. However, our investigation of AIRL in the context of poker reveals that AIRL has difficulty extracting a sufficiently informative reward function. This suggests that AIRL may struggle in domains that are characterized by large state-action spaces, high stochasticity, partial observability, and sparse rewards.

In response to these challenges, we contribute Hybrid Adversarial Inverse Reinforcement Learning (H-AIRL), a novel IRL method that leverages adversarial learning, supervised learning, and stochastic regularization. Inspired by techniques in conditional generative adversarial networks, our approach supplements the adversarial loss with an additional supervised signal, thereby improving sample efficiency and stabilizing training [11]. Through systematic evaluations on carefully selected Gymnasium benchmarks and in Heads-Up Limit Hold’em (HULHE) poker, we demonstrate that H-AIRL exhibits improved sample efficiency and more stable learning compared to AIRL.

2 Related Work

Inverse Reinforcement Learning (IRL) is a framework for recovering the underlying reward function from expert demonstrations, thereby enabling an agent to infer the motivations behind expert behavior. Early work focused on linear reward models [12, 13], and later probabilistic formulations recast the problem under the maximum-entropy principle [9, 14]. This approach establishes a probabilistic model over trajectories that captures the variability in expert behavior.

Building upon these ideas, adversarial methods for imitation learning have emerged. Generative Adversarial Imitation Learning (GAIL) leverages a discriminator to differentiate between expert and generated behavior, implicitly drawing on maximum-entropy formulations [15]. However, GAIL does not yield an explicit reward function. To overcome this limitation, Fu et al. introduced Adversarial Inverse Reinforcement Learning (AIRL), which reframes the discriminator as an odds ratio between the expert and policy behaviors to jointly infer a reward function and a policy [10]. More recent works such as Generative Intrinsic Reward-driven Imitation Learning (GIRIL) [16] and Belief-Module Imitation Learning (BMIL) [17] have further addressed challenges like partial observability, though these approaches focus on imitation rather than explicitly recovering the latent reward function.

In the domain of poker, Counterfactual Regret Minimization (CFR) has long been the gold standard for developing superhuman agents by iteratively approximating Nash equilibria with strong theoretical guarantees [18]. However, CFR-based methods are computationally intensive and require significant domain-specific tuning. Although reinforcement learning has emerged as a promising alternative for learning strategies directly from interactions, conventional RL methods struggle with sparse, stochastic, and delayed rewards in poker [6].

IRL offers the potential to mitigate these challenges by inferring dense reward signals from expert demonstrations, thus providing richer guidance than the sparse, terminal rewards typical of RL. Despite the maturity of RL methods in poker, the application of inverse RL to capture the nuanced reward incentives in this domain has not yet been explored. Our experimental results reveal that AIRL struggles to extract sufficiently informative reward functions in HULHE poker, underscoring its limitations in such complex environments.

To address these gaps, our work introduces a novel hybrid IRL framework, H-AIRL, which incorporates a supervised loss component derived directly from expert data. While supervised signals have been successfully integrated into generative adversarial networks for tasks such as conditional image synthesis [11], no prior work has fused a similar supervised term within an IRL framework. By leveraging this additional guidance, H-AIRL aims to stabilize training and enhance the quality of the recovered reward function, particularly in challenging environments like poker.

In summary, while significant progress has been made in both imitation learning and RL for complex tasks, our work is the first to integrate a supervised loss component into an IRL framework and to apply IRL to the challenging domain of poker.

3 Background

3.1 Reinforcement Learning

Reinforcement Learning (RL) is one of three main machine learning paradigms alongside supervised and unsupervised learning [19]. In RL, an agent interacts with an environment by taking actions, receiving feedback in the form of rewards, and adjusting its behavior to maximize cumulative expected returns [19, 20].

Formally, RL problems are modeled as Markov Decision Processes (MDPs). An MDP is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, where \mathcal{S} is the set of states, \mathcal{A} is the set of actions, $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ describes the environment's transition dynamics, $r(\mathbf{s}, \mathbf{a})$ is the reward function, and $\gamma \in [0, 1]$ is a discount factor. At each time step, the agent observes a state $\mathbf{s} \in \mathcal{S}$, selects an action $\mathbf{a} \in \mathcal{A}$, after which it transitions to a new state \mathbf{s}' while receiving a scalar reward $r(\mathbf{s}, \mathbf{a})$. The goal is to find a policy $\pi(\mathbf{a}|\mathbf{s})$ that maximizes the expected discounted return.

Q-learning is a foundational RL algorithm that learns an action-value function $Q(\mathbf{s}, \mathbf{a})$, representing the expected return for taking action \mathbf{a} in state \mathbf{s} and following the optimal policy thereafter [21]. In the tabular setting, Q-learning is proven to converge to the optimal policy under mild conditions [20]. In high-dimensional or continuous state spaces, function approximators such as deep neural networks can be employed to estimate $Q(\mathbf{s}, \mathbf{a})$. This approach led to the development of Deep Q-Networks (DQN), an algorithm that achieved human-level performance on Atari games [2]. Despite these successes, a key assumption in RL is that the reward function $r(\mathbf{s}, \mathbf{a})$ is known or specified by the designer, a constraint that motivates the field of Inverse Reinforcement Learning.

3.2 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) inverts the classical RL paradigm by inferring the underlying reward function from expert demonstrations rather than assuming it is given [22, 12]. Formally, consider an MDP defined by the tuple $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$. In standard RL one seeks an optimal policy $\pi^*(\mathbf{a}|\mathbf{s})$ that maximizes the cumulative reward. In IRL, the agent observes expert demonstrations, consisting of trajectories sampled from a policy $\pi^*(\mathbf{a}|\mathbf{s})$ that is assumed to be optimal, and aims to infer a reward function $f(\mathbf{s}, \mathbf{a})$ that explains the expert's behavior.

Early IRL algorithms, which focused on linear reward models, revealed the fundamental *degeneracy* in IRL, namely, that multiple distinct reward functions can yield the same optimal policy [12, 13]. To mitigate this ambiguity, Ziebart et al. introduced the *maximum-entropy* IRL framework, which assigns a probability to each trajectory that both explains the expert's behavior and favors high-entropy policies [9]. In this formulation, the expert's trajectory distribution $\rho_\theta(\tau)$ is given by:

$$\rho_\theta(\tau) \propto \exp\left(\sum_{t=0}^T r_\theta(\mathbf{s}_t, \mathbf{a}_t)\right),$$

where $\tau = (\mathbf{s}_0, \mathbf{a}_0, \dots, \mathbf{s}_T, \mathbf{a}_T)$ denotes a trajectory and $r_\theta(\mathbf{s}, \mathbf{a})$ is a reward function with parameters θ . To transform this expression into a valid probability distribution, a normalizing constant known as the partition function is required:

$$Z_\theta = \sum_\tau \exp\left(\sum_{t=0}^T r_\theta(\mathbf{s}_t, \mathbf{a}_t)\right).$$

In high-dimensional spaces, calculating Z_θ is typically intractable. Methods such as Guided Cost Learning (GCL) address this issue by using function approximators (e.g., neural networks) to estimate the reward function without requiring explicit computation of Z_θ [23]. Nonetheless, scaling maximum-entropy IRL to complex, high-dimensional problems remains challenging [10]. This motivated the development of alternative approaches.

In particular, Generative Adversarial Imitation Learning (GAIL) represents a significant milestone in imitation learning [15]. GAIL reframes imitation as a min-max game between a generator (the policy) and a discriminator. The discriminator $D_\phi(\mathbf{s}, \mathbf{a})$ is trained to distinguish expert state-action pairs from those generated by the policy $\pi(\mathbf{a}|\mathbf{s})$, while the policy is optimized to deceive the discriminator. Although GAIL is not strictly an IRL method, as it does not recover an explicit reward function, it represents a significant milestone in imitation learning and has inspired subsequent work in IRL. Notably, its adversarial formulation paved the way for Adversarial Inverse Reinforcement Learning (AIRL) [10].

3.2.1 The Adversarial IRL Framework.

The AIRL algorithm expands on GAIL by integrating the maximum-entropy formulation of IRL into an adversarial framework. AIRL jointly learns a policy and a reward function by reinterpreting the discriminator as an energy-based model. In AIRL, the discriminator is defined as:

$$D_\theta(\mathbf{s}, \mathbf{a}) = \frac{\exp(f_\theta(\mathbf{s}, \mathbf{a}))}{\exp(f_\theta(\mathbf{s}, \mathbf{a})) + \pi_\phi(\mathbf{a}|\mathbf{s})}, \quad (1)$$

where $\pi_\phi(\mathbf{a}|\mathbf{s})$ denotes the probability of taking action \mathbf{a} in state \mathbf{s} under policy π_ϕ with parameters ϕ , and $f_\theta(\mathbf{s}, \mathbf{a})$ is the learned reward function with parameters θ . Meanwhile, the policy objective is defined as:

$$\max_{\phi} \mathbb{E}_{\tau \sim \pi_\phi} \left[\sum_{t=0}^T \left(f_\theta(\mathbf{s}_t, \mathbf{a}_t) - \log \pi_\phi(\mathbf{a}_t | \mathbf{s}_t) \right) \right]. \quad (2)$$

From an optimization perspective, maximizing the policy objective is equivalent to minimizing a loss function defined as its negative:

$$\mathcal{L}_{\text{AIRL}}^{\text{policy}} = -f_\theta(\mathbf{s}, \mathbf{a}) + \log \pi_\phi(\mathbf{a}|\mathbf{s}). \quad (3)$$

Here, the negated reward term $-f_\theta(\mathbf{s}, \mathbf{a})$ encourages the policy to favor actions that yield higher rewards from the learned reward function. Meanwhile, the entropy regularization term $\log \pi_\phi(\mathbf{a}|\mathbf{s})$ helps maintain exploration by increasing the loss for actions to which the policy assigns a high probability.

So far, we have described AIRL in its state-action form. The *state-only* variant constrains the learned reward to:

$$f_{\theta, \omega}(\mathbf{s}, \mathbf{a}, \mathbf{s}') = g_\theta(\mathbf{s}, \mathbf{a}) + \gamma h_\omega(\mathbf{s}') - h_\omega(\mathbf{s}),$$

where $g_\theta(\mathbf{s}, \mathbf{a})$ captures the true reward (up to an additive constant) and $h_\omega(\mathbf{s})$ is a potential-based shaping term. In this decomposition, $g_\theta(\mathbf{s}, \mathbf{a})$ alone is a disentangled reward function that remains valid when transferred to new MDPs with different transition dynamics. Crucially, however, learning the state-only form requires access to the actual next state \mathbf{s}' , which is infeasible in many offline or partially observable domains where a faithful simulation model is not available. In such cases, one must revert to the unrestricted state-action form. Accordingly, in the remainder of this paper, we adopt AIRL's state-action formulation as our baseline.

4 The Hybrid-AIRL Framework

The Hybrid-AIRL algorithm introduces a novel integration of supervised learning into both the policy and the discriminator within the adversarial maximum-entropy framework.

4.1 The Policy Objective

Under the maximum-entropy principle, the probability of a trajectory τ can be modeled as:

$$\rho_\theta(\tau) = p(\mathbf{s}_0) \prod_{t=0}^{T-1} p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t) \exp \left(\sum_{t=0}^T f_\theta(\mathbf{s}_t, \mathbf{a}_t) \right) \frac{1}{Z_\theta},$$

where $p(\mathbf{s}_0)$ is the probability of the initial state, $p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$ represents the environment dynamics, and Z_θ is the partition function ensuring normalization. As such, $\rho_\theta(\tau)$ represents the probability distribution over trajectories given reward function $f_\theta(\mathbf{s}, \mathbf{a})$, reflecting which trajectories are desirable based on the model's current parametrization θ of the reward function. Taking the logarithm yields:

$$\begin{aligned} \log \rho_\theta(\tau) &= \sum_{t=0}^T f_\theta(\mathbf{s}_t, \mathbf{a}_t) - \log Z_\theta \\ &\quad + \log p(\mathbf{s}_0) + \sum_{t=0}^{T-1} \log p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t). \end{aligned}$$

We then define the IRL objective $J(\theta)$ as the expected log-likelihood:

$$J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta} [\log \rho_\theta(\tau)].$$

In maximum-entropy IRL frameworks such as state-action AIRL, the goal is to align the policy's trajectory distribution π_ϕ with the target distribution ρ_θ , which represents the probability of trajectories based on the current parametrization θ of the reward function. This can be enforced by minimizing the KL divergence:

$$\mathcal{D}_{\text{KL}}(\pi_\phi \| \rho_\theta) = \mathbb{E}_{\tau \sim \pi_\phi} \left[\log \frac{\pi_\phi(\tau)}{\rho_\theta(\tau)} \right]. \quad (4)$$

Maximizing the negative of this divergence leads to the objective:

$$\max_{\phi} \mathbb{E}_{\tau \sim \pi_{\phi}} [\log \rho_{\theta}(\tau) - \log \pi_{\phi}(\tau)]. \quad (5)$$

Because the environment is assumed to satisfy the Markov property, the probability of a trajectory τ under the policy π_{ϕ} factorizes as:

$$\pi_{\phi}(\tau) = p(\mathbf{s}_0) \prod_{t=0}^{T-1} p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t) \pi_{\phi}(\mathbf{a}_t | \mathbf{s}_t).$$

Since both $\rho_{\theta}(\tau)$ and $\pi_{\phi}(\tau)$ include the initial state probability $p(\mathbf{s}_0)$ and the environment dynamics $p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$ in their definitions, these terms cancel out when computing the difference:

$$\log \rho_{\theta}(\tau) - \log \pi_{\phi}(\tau) \propto \sum_{t=0}^T (f_{\theta}(\mathbf{s}_t, \mathbf{a}_t) - \log \pi_{\phi}(\mathbf{a}_t | \mathbf{s}_t)). \quad (6)$$

Finally, using Equations (5) and (6), we find that the policy objective is equivalent to maximizing the expected cumulative entropy-regularized reward:

$$\max_{\phi} \mathbb{E}_{\tau \sim \pi_{\phi}} \left[\sum_{t=0}^T (f_{\theta}(\mathbf{s}_t, \mathbf{a}_t) - \log \pi_{\phi}(\mathbf{a}_t | \mathbf{s}_t)) \right].$$

This expression precisely matches the policy objective of AIRL, described by Equation (2). In essence, this objective is derived from the standard maximum-entropy IRL framework, which is solely concerned with aligning the policy's trajectory distribution with that of the expert.

In our H-AIRL framework, we extend this objective by aligning the policy's *action* distribution with the expert's. That is, in addition to matching trajectories, we incorporate a supervised learning objective that minimizes the discrepancy between the policy's actions and the expert's actions. We define this hybrid policy objective as:

$$\min_{\phi} \left[\underbrace{(1-\alpha) \mathcal{D}_{\text{KL}}(\pi_{\phi} \| \rho_{\theta})}_{(1) \text{ Max-Entropy IRL Objective}} + \underbrace{\alpha \mathbb{E}_{\tau \sim \rho_E} \left[-\sum_{t=0}^T \log \pi_{\phi}(\mathbf{a}_t | \mathbf{s}_t) \right]}_{(2) \text{ Supervised Learning Objective}} \right], \quad (7)$$

where ρ_E is the distribution of expert trajectories and $\alpha \in [0, 1]$ is a weighting factor. The first term in Equation (7) coincides with the maximum-entropy IRL objective as defined in Equation (4), which leads to AIRL's policy loss formulation in Equation (3). The second term constitutes the supervised learning objective. Consequently, the policy's loss function becomes:

$$\begin{aligned} \mathcal{L}_{\text{H-AIRL}}^{\text{policy}} &= (1-\alpha) \mathcal{L}_{\text{AIRL}}^{\text{policy}} + \alpha \mathcal{L}_{\text{S}}^{\text{policy}} \\ &= (1-\alpha) [-f_{\theta}(\mathbf{s}, \mathbf{a}) + \log \pi_{\phi}(\mathbf{a} | \mathbf{s})] + \alpha \mathcal{L}_{\text{S}}^{\text{policy}}, \end{aligned}$$

where \mathcal{L}_{S} denotes the supervised loss component. This hybrid objective benefits from both adversarial IRL and direct supervised imitation. For discrete action spaces, the supervised loss can be computed as the Cross-Entropy loss or KL divergence between the policy's action distribution $\pi_{\phi}(\cdot | \mathbf{s})$ and the expert's action distribution $\rho_E(\cdot | \mathbf{s})$. For continuous action spaces, this can be approximated using an appropriate loss function (e.g., the mean-squared-error loss).

4.2 The Discriminator Objective

Given the definition of the discriminator in Equation (1) and a fixed policy π_{ϕ} , AIRL optimizes the discriminator parameters θ via the standard cross-entropy loss:

$$\begin{aligned} \mathcal{L}_{\text{AIRL}}^{\text{disc}} &= -\mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \rho_E} [\log D_{\theta}(\mathbf{s}, \mathbf{a})] \\ &\quad - \mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \pi_{\phi}} [\log (1 - D_{\theta}(\mathbf{s}, \mathbf{a}))]. \end{aligned} \quad (8)$$

This adversarial objective encourages f_θ to assign higher scores to expert state-action pairs, thereby inferring a reward function that explains the demonstrations.

In AIRL, the discriminator learns to distinguish expert from policy-generated trajectories. However, adversarial training alone provides no guarantee that the learned reward f_θ aligns with the true environment reward r_{env} . Optimizing a policy under f_θ thus defines a shaped MDP, which has the same $(\mathcal{S}, \mathcal{A}, P, \gamma)$ but with reward f_θ in place of r_{env} . A policy that performs well in this shaped MDP may perform poorly when evaluated under the original reward [24, 25]. To prevent this deviation, we regularize f_θ with an additional supervised mean-squared-error loss when ground-truth environment rewards $r_{\text{env}}(\mathbf{s}, \mathbf{a})$ are available for the expert demonstrations:

$$\mathcal{L}_S^{\text{disc}} = \mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \rho_E} [(f_\theta(\mathbf{s}, \mathbf{a}) - r_{\text{env}}(\mathbf{s}, \mathbf{a}))^2]. \quad (9)$$

By combining the adversarial and supervised loss components, we obtain a new discriminator objective:

$$\mathcal{L}_{\text{AIRL+S}}^{\text{disc}} = (1 - \beta) \mathcal{L}_{\text{AIRL}}^{\text{disc}} + \beta \mathcal{L}_S^{\text{disc}},$$

where $\beta \in [0, 1]$ is a weighting factor that balances the adversarial loss against supervised loss. This approach aims to prevent pathological reward shaping and to ensure that a policy trained using the learned f_θ continues to perform well when evaluated in the original MDP. If ground-truth rewards are unavailable, we set $\beta = 0$.

4.3 Stochastic Regularization

In the hybrid policy objective (Equation 7), the supervised term can rapidly drive the policy π_ϕ toward expert-like outputs, as demonstrated by our experimental results in Section 6. Consequently, the discriminator D_θ is exposed primarily to high-quality actions and receives limited feedback for distinguishing realistic from unrealistic behavior, which may lead to overfitting.

To mitigate this effect and restore a meaningful adversarial signal, we introduce stochastic regularization by injecting Gaussian noise that decays along the mini-batch axis. Let $\mathcal{B} = \{(\mathbf{s}_i, \mathbf{a}_i)\}_{i=0}^{B-1}$ represent a mini-batch of size B that contains state-action pairs $(\mathbf{s}_i, \mathbf{a}_i)$ associated with the policy π_ϕ . For every index i in the batch, we form a perturbed action vector $\tilde{\mathbf{a}}_i$ using Gaussian noise:

$$\tilde{\mathbf{a}}_i = \mathbf{a}_i + \boldsymbol{\eta}_i, \quad \boldsymbol{\eta}_i \sim \mathcal{N}(\mathbf{0}, \sigma_i^2 \mathbf{I}),$$

where σ_i is the standard deviation and $\tilde{\mathbf{a}}_i$ is the perturbed policy action. The standard deviation per sample σ_i decays monotonically from the first element of the mini-batch to the last. As a result, each batch that reaches the discriminator D_θ contains a spectrum of action qualities, ranging from strongly perturbed (large σ_i) to nearly noise-free (small σ_i).

By combining these regularized adversarial and supervised losses, we obtain the Hybrid-AIRL discriminator objective:

$$\mathcal{L}_{\text{H-AIRL}}^{\text{disc}} = (1 - \beta) \mathcal{L}_{\text{AIRL+SR}}^{\text{disc}} + \beta \mathcal{L}_{\text{S+SR}}^{\text{disc}}. \quad (10)$$

5 Experimental Setup

The IRL training procedure of H-AIRL¹ alternates between:

1. Training the discriminator to classify expert trajectories from those generated by the policy.
2. Updating the policy to maximize the inferred reward.

Once this IRL training process completes, the discriminator effectively becomes a surrogate for the expert's reward function, encapsulating the underlying reward incentives observed in the expert demonstrations. This learned reward function serves to both capture the underlying motivations of expert behavior and provide a dense signal for reinforcement learning. Thus, we subsequently consider an RL training phase, where we integrate this learned reward function into an RL agent, namely, a Proximal Policy Optimization (PPO) agent from Stable-Baselines3 for Gymnasium benchmarks and a DQN agent from the RLCard framework for HULHE poker [26, 27].

¹<https://github.com/silue-dev/hairl>

5.1 Benchmarks

We first benchmark AIRL and H-AIRL on a selection of Gymnasium tasks: Pendulum, Ant, HalfCheetah, Acrobot, LunarLander, and MountainCar [28]. The first three tasks (Pendulum, Ant, and Half-Cheetah) are the same continuous-control benchmarks that were used in the original AIRL paper, making them well-suited for comparison with our approach [10]. On the other hand, Acrobot, LunarLander, and MountainCar are discrete environments, which complement the continuous set and broaden the scope of our evaluation. To obtain expert data for these tasks, we train a PPO agent [29], following the approach used in AIRL.²

Next, we explore a more challenging task: HULHE poker. This imperfect-information, zero-sum game features partial observability, stochastic dynamics, and a vast state-action space. Rewards are only observed at the end of each hand, which makes strategic learning particularly difficult. For this task, we use the IRC Poker dataset³, a high-quality collection of online poker games featuring both expert amateur players and professional players, including a world champion. This extensive dataset comprises millions of real-world poker game state observations, including over 1 million state-action pairs for HULHE poker.

We note that a distinct challenge in online poker datasets is that folding actions reveal no information about the player’s hand, as folded cards remain concealed and are therefore absent from the data. This limitation poses a significant obstacle for machine learning algorithms, particularly IRL agents, which rely on observable behavior to infer the underlying reward function. Without the ability to learn from folding actions, the IRL agent must focus on mastering the remaining actions (i.e., calling, raising, and checking) to infer the expert strategy.

5.2 Evaluation

To evaluate the effectiveness of AIRL and H-AIRL, we assess both the alignment of the learned policies with expert behavior and the quality of the inferred reward functions.

First, for Gymnasium benchmarks, we assess the learning curves of the reward obtained by the agent during training. This widely used reinforcement learning evaluation method offers insight into the agent’s sample efficiency and performance under the inferred reward function.

For HULHE poker, where we consider a real-world dataset, we introduce the *state-level action alignment* as a complementary metric to assess IRL training performance. We define this metric as the percentage of visited states in which the learned policy selects the same action as the expert. This metric provides insight into the model’s ability to mimic the expert’s probabilistic decision-making tendencies. Because this metric is applicable to any discrete action space environment, we also compute it for Gymnasium benchmarks with discrete action spaces.

Next, we extend our evaluation beyond the policy and assess the quality of the learned reward function. One key limitation of prior IRL work is the lack of direct validation of the inferred reward signal. To address this, we train separate RL agents (i.e., PPO or DQN) using the learned reward function and analyze their learning curves with respect to the environment reward. If the inferred reward function accurately captures task-relevant features, agents should be able to learn effective policies using only the IRL-derived reward. For poker, we integrate the learned reward function into the DQN implementation provided by RLCard, an open-source library for applying reinforcement learning algorithms to card game environments [27]. Note that for all learning curves, we depict the mean and standard deviation across 10 training runs for each algorithm [31].

For poker, reward learning curves during the RL phase can be less informative, as the agent is trained against a random opponent — an adversary against which even simplistic strategies, such as always raising, can achieve positive expected returns. Consequently, we further investigate the effectiveness of IRL-derived rewards in poker by directly opposing DQN agents trained with the learned dense reward versus those trained with the traditional sparse reward (game payoff). This allows us to assess whether these IRL models can compete with standard RL approaches. To ensure a robust comparison, we use the RLCard framework to simulate 1,000,000 tournaments across 20 random seeds [27]. Performance is measured in milli-big-blinds per hand (mbb/h), a standard poker metric reflecting average gains or losses per hand, normalized across games.

Finally, for MountainCar, an environment with a two-dimensional state space and three discrete actions, we generate directly interpretable 2D visualizations for both AIRL and H-AIRL. These visualizations illustrate the reward function’s preferred action at each state.

²Fu et al. employed TRPO rather than PPO, its successor [10, 30]. For MountainCar, we exceptionally use DQN [2].

³https://poker.cs.ualberta.ca/irc_poker_database.html

6 Experimental Results

6.1 IRL Training

We present the performance of AIRL and H-AIRL during the IRL training phase. Figure 1 shows the reward learning curves on the Gymnasium benchmarks.

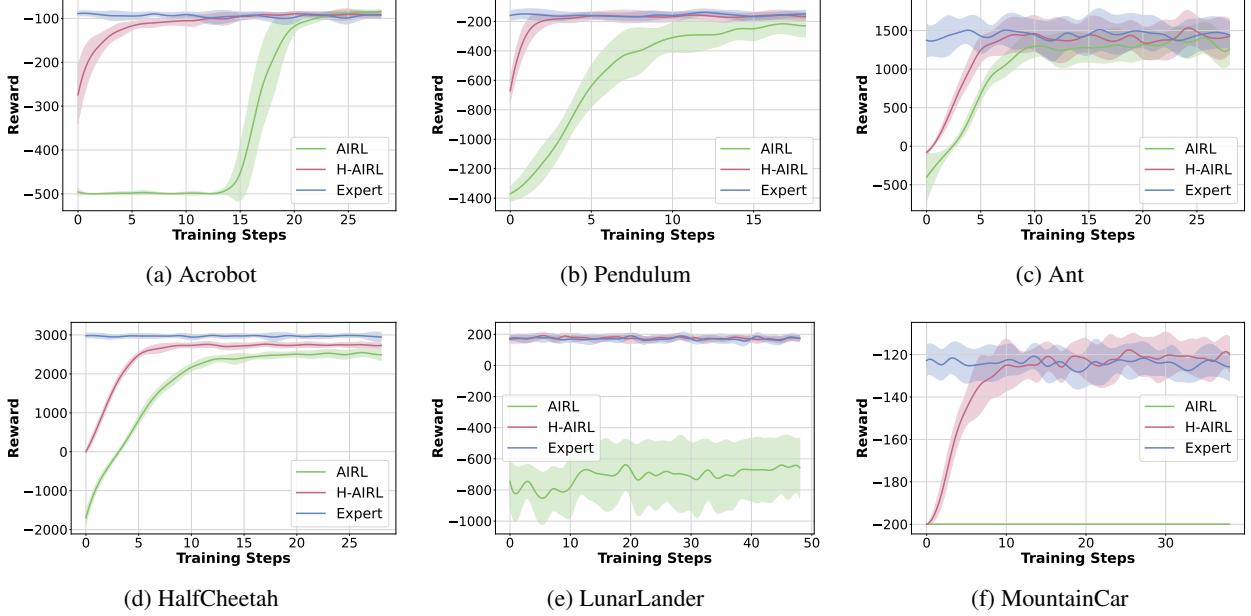


Figure 1: Reward learning curves for AIRL (green) and H-AIRL (red) on Gymnasium benchmarks, alongside an expert PPO baseline (blue).

For tasks with discrete action spaces, including poker, Figure 2 depicts the policy’s state-level action alignment with respect to the expert.

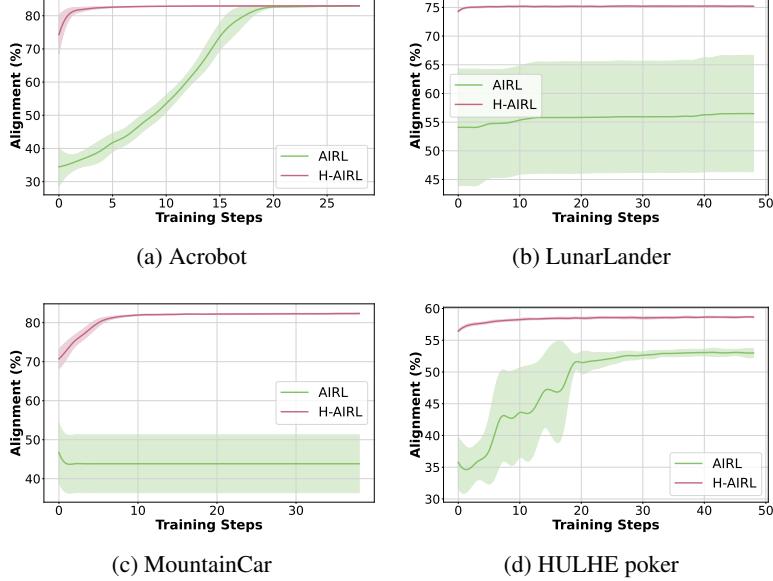


Figure 2: The policy’s state-level action alignment with the expert, for AIRL (green) and H-AIRL (red), across benchmarks with discrete action spaces.

The H-AIRL policy achieves substantially better action alignment throughout learning as it approaches the expert behavior more quickly and accurately than AIRL, with reduced variance throughout the learning process.

Across both evaluation metrics, reward learning and state-level alignment, the H-AIRL model demonstrates a significantly improved ability to approximate expert-like behavior. It converges more rapidly, exhibits lower variance, and better aligns with expert strategies compared to AIRL.

6.2 RL Training

We depict the performance of RL agents that utilize the learned reward functions from AIRL and H-AIRL.

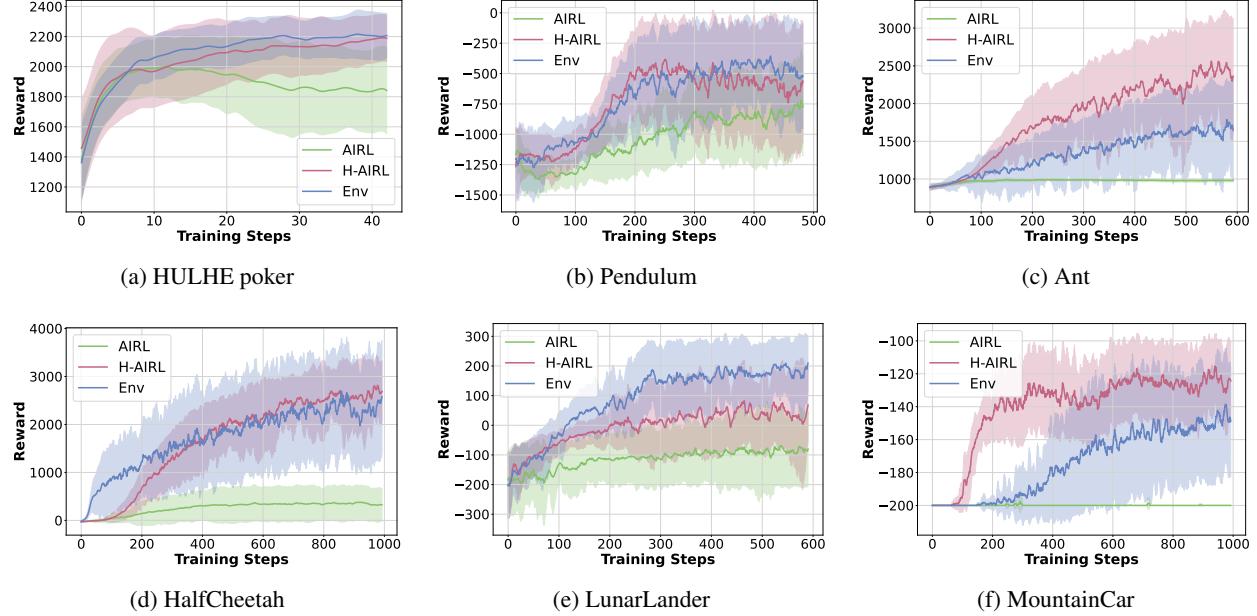


Figure 3: RL training curves of PPO or DQN agents using environment (blue), AIRL-derived (green), and H-AIRL-derived (red) rewards on Gymnasium benchmarks and Heads-Up Limit Hold'em poker.

Figure 3 shows the learning curves of an RL agent (PPO or DQN) on the Gymnasium benchmarks, using the learned reward function from the IRL training phase. Across all tasks, we notice a better or otherwise equal ability of H-AIRL to approach expert learning compared to AIRL.

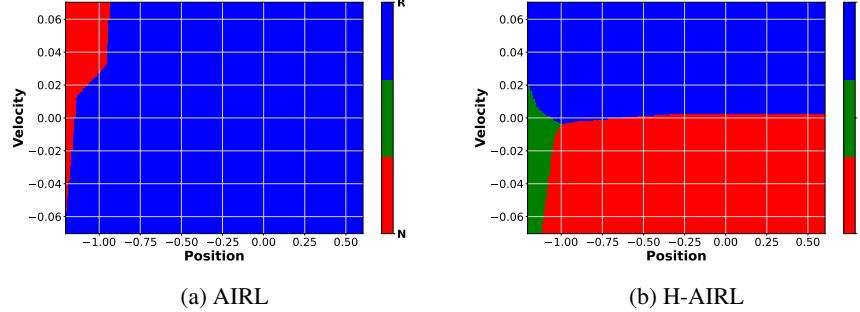


Figure 4: Preferred actions according to the learned reward functions over the MountainCar state space (position vs. velocity), for each discrete action: “thrust right” (R, blue), “no thrust” (N, green), or “thrust left” (L, red).

This difference in the learned reward functions can be illustrated further by visualizing the reward function output in MountainCar, as shown in Figure 4a. In Figure 4a, we see that AIRL’s learned reward has a strong bias towards the “thrust right” action, and there is no area of the state space where “no thrust” is the most valuable action in terms of

reward. As a result, the agent never learns the back-and-forth switching between actions that is needed to build enough momentum to complete the task successfully. By contrast, in Figure 4b, we see that H-AIRL’s reward function is well-balanced, having appropriate areas of the state-action space where each action is most valuable, allowing the RL agent to reproduce the expert’s oscillatory strategy and reach the goal.

For poker, Table 1 provides the results of 1,000,000 tournaments of AIRL-DQN and H-AIRL-DQN against RLCard’s default DQN model to see whether agents trained with a learned reward function can beat agents trained with the traditional (sparse) reward of just the game’s payout. To account for potential correlations between seeds, we apply the Bonferroni correction [32], adjusting the significance threshold to $p < 0.0025$.

Model	Payoff (mbb/h)	p-value
H-AIRL-DQN	$+96 \pm 14$	$< 10^{-10}$
AIRL-DQN	-693 ± 34	$< 10^{-10}$

Table 1: The performance (i.e., the average payoff and standard error in mbb/h) of AIRL-DQN and H-AIRL-DQN against DQN in HULHE poker.

The tournament results presented in Table 1 show a stark contrast between AIRL-DQN and H-AIRL-DQN when competing against RLCard’s default DQN agent. AIRL-DQN performs significantly worse than DQN, yielding a negative payoff of -693 ± 34 mbb/h, indicating that it consistently loses to the baseline. In contrast, H-AIRL-DQN achieves a positive payoff of $+96 \pm 14$ mbb/h, demonstrating that it not only outperforms AIRL-DQN, but also performs competitively with the default DQN model. This difference is significant, considering that professional poker players consider 50 mbb/h a sizable margin [33].

7 Ablation Study

Hybrid-AIRL introduces four core hyperparameters:

- $\alpha \in [0, 1]$ is the *policy supervision weight*, which scales the supervised cross-entropy term in the policy loss, as defined in Equation (7).
- $\beta \in [0, 1]$ is the *discriminator supervision weight*, which blends the mean-squared error against ground-truth rewards into the discriminator loss, as defined in Equation (10).
- $\sigma_{\text{start}} \in [0, 1]$ defines the normalized *initial noise standard deviation*, which is the standard deviation of the Gaussian perturbation applied to the first sample in every mini-batch, as described in Section 4.3.
- $\sigma_{\text{end}} \in [0, 1]$ defines the normalized *final noise standard deviation*, which is applied to the last sample in the mini-batch; values $\sigma_{\text{end}} > 0$ leave a residual noise floor.

Collectively, these hyperparameters control the interplay between adversarial learning, supervised learning, and stochastic regularization. Because their effects are interdependent, it is important to understand how each component contributes to the IRL and RL training dynamics. We therefore study these factors individually through a set of controlled ablation experiments.

7.1 Effect of Policy Supervision (α)

Figure 5a plots the learning curve for MountainCar as α varies during the IRL training phase. Introducing a relatively small amount of policy supervision (e.g., $\alpha \approx 0.1$) provides sufficient guidance for the generator to identify the expert-like action distribution, resulting in the substantial performance gains over AIRL observed in Section 6. However, full supervision (i.e., $\alpha = 1$) hurts performance. With $\alpha = 1$ the adversarial game is eliminated, preventing the discriminator from learning a meaningful reward signal.

We note that the value of α that yields the highest policy return may not coincide with the value that results in the best performance when the inferred reward is used to train a separate reinforcement learning agent. This subtle trade-off suggests that designers may prioritize either fast policy convergence or reward fidelity, depending on the application.

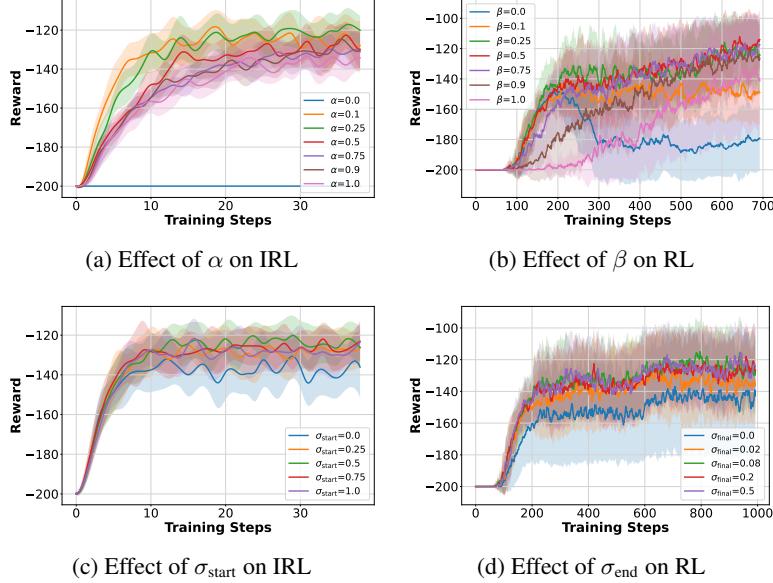


Figure 5: One-factor-at-a-time (OFAT) sweeps on MountainCar for H-AIRL’s core hyperparameters: (a) the policy supervision weight α , (b) the discriminator supervision weight β , (c) the initial noise standard deviation σ_{start} , and (d) the final noise standard deviation σ_{end} . Each curve shows the mean performance and standard deviation over 10 independent runs.

7.2 Effect of Discriminator Supervision (β)

A complementary trend is observed in Figure 5b. Introducing a moderate level of reward supervision (e.g., $\beta \approx 0.25$) improves the performance of the learned reward function. However, assigning excessively high values, such as $\beta = 1.0$, leads to performance degradation once again. These results reinforce the hypothesis that a hybrid approach of combining both supervised and adversarial learning yields the best performance, whereas using either approach alone is often suboptimal.

While reward supervision can be beneficial, it is not essential for learning a high-performing reward function. For example, our results in the HULHE poker setting were achieved without any reward supervision.

7.3 Effect of Stochastic Regularization ($\sigma_{\text{start}}, \sigma_{\text{end}}$)

Noise injection benefits training, especially in more complex environments. Figures 5c and 5d show that by starting with substantial noise levels (e.g., $\sigma_{\text{start}} \approx 0.9$) and decaying to a smaller nonzero σ_{end} , we ensure the discriminator continually sees a diverse spectrum of action qualities, even in late stages of training, thereby avoiding overfitting to homogenized, near-expert behaviors. Moreover, retaining a noise floor (e.g., $\sigma_{\text{end}} \approx 0.08$) is often better than annealing to zero. A nonzero σ_{end} prevents an “expert-vs-expert” collapse; by preserving input diversity through perturbations, H-AIRL enables the discriminator to learn robust distinctions rather than fitting to noise.

8 Discussion

We experimentally show that H-AIRL consistently outperforms AIRL, achieving improved policy learning and reward function inference across all evaluation metrics. While AIRL eventually learns an effective policy in Gymnasium benchmarks, H-AIRL exhibits faster convergence, greater training stability, better action alignment, and action distributions that better match those of expert players. The advantages of a hybrid IRL framework are further underscored in the context of HULHE poker, where H-AIRL is able to infer a more informative reward function that more effectively guides the RL agent towards expert behavior. This is evidenced by the improved learning curve in the RL training phase and the higher payoffs observed in our tournament evaluations. Overall, these findings suggest that while state-action AIRL serves as a powerful baseline for learning from expert demonstrations, the modifications introduced in H-AIRL provide a robust enhancement that is effective at scaling the IRL approach to even more complex settings. The improved performance of H-AIRL in both Gymnasium benchmarks and HULHE poker highlights its

potential as a promising approach for inverse reinforcement learning in real-world domains characterized by highly complex dynamics and sparse rewards.

Our aim in this work is to isolate and evaluate the contribution of H-AIRL’s hybrid loss framework relative to the foundational AIRL baseline. A broader empirical evaluation against recent IRL methods is left to future work in order to better situate H-AIRL within the current landscape [34, 35]. While our results are encouraging, the study has several limitations. First, our poker data lacks folding actions, a common limitation in real-world datasets where folded cards are never revealed. Second, H-AIRL does not recover disentangled rewards that lead to theoretical guarantees for transfer, and does not explicitly address partial observability. These points suggest several directions for future work, such as extending the hybrid framework formulation to produce disentangled rewards, and studying recurrent or belief-state extensions of H-AIRL for partially observable domains.

Use of Generative AI

The use of generative AI was limited to tasks such as language refinement, without replacing critical analysis, original research, or authorship contributions.

Acknowledgements

LW and PJKL gratefully acknowledge support from the Research Foundation Flanders (FWO), via ACCELERATE project G059423N. PJKL gratefully acknowledges support from the Research council of the Vrije Universiteit Brussel (OZR-VUB via grant number OZR3863BOF). This research acknowledges funding from the Flemish Government through the AI Research Program. We made use of computational resources and services provided by the Flemish Supercomputer Centre (VSC), funded by the FWO and the Flemish Government.

References

- [1] Eduardo F. Morales, Rafael Murrieta-Cid, Israel Becerra, and Marco A. Esquivel-Basaldua. A survey on deep learning and deep reinforcement learning in robotics with a tutorial on deep reinforcement learning. *Intelligent Service Robotics*, 2021.
- [2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.
- [3] Pieter J. K. Libin, Arno Moonens, Timothy Verstraeten, Fabian Perez-Sanjines, Niel Hens, Philippe Lemey, and Ann Nowé. Deep reinforcement learning for large-scale epidemic control. In Yuxiao Dong, Georgiana Ifrim, Dunja Mladenić, Craig Saunders, and Sofie Van Hoecke, editors, *Proceedings of ECML-PKDD*, 2021.
- [4] Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, and Honglak Lee. Control of memory, active perception, and action in minecraft. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of ICML*, June 2016.
- [5] Todd W. Neller and Marc Lanctot. An introduction to counterfactual regret minimization. In *Proceedings of EAAI*, 2013.
- [6] Enmin Zhao, Renye Yan, Jinqiu Li, Kai Li, and Junliang Xing. Alphaholdem: High-performance artificial intelligence for heads-up no-limit poker via end-to-end reinforcement learning. In *Proceedings of AAAI*, 2022.
- [7] Fredrik A. Dahl. A reinforcement learning algorithm applied to simplified two-player texas hold’em poker. In *Proceedings of LNCS*, 2001.
- [8] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 2021.
- [9] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of AAAI*, 2008.
- [10] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *Proceedings of ICLR*, 2018.
- [11] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In Doina Precup and Yee Whye Teh, editors, *Proceedings of ICML*, Aug 2017.

- [12] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In Pat Langley, editor, *Proceedings of ICML*, 2000.
- [13] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of ICML*, 2004.
- [14] Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of AISTATS*, April 2011.
- [15] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Proceedings of NeurIPS*, 2016.
- [16] Xingrui Yu, Yueming Lyu, and Ivor W. Tsang. Intrinsic reward driven imitation learning via generative model. In Ameet Talwalkar and Kilian Q. Weinberger, editors, *Proceedings of ICML*, 2020.
- [17] Tanmay Gangwani, Joel Lehman, Qiang Liu, and Jian Peng. Learning belief representations for imitation learning in pomdps. In Ryan P. Adams and Vibhav Gogate, editors, *Proceedings of UAI*, 22–25 Jul 2020.
- [18] Huale Li, Xuan Wang, Fengwei Jia, Yifan Li, and Qian Chen. A survey of nash equilibrium strategy solving based on cfr. *Archives of Computational Methods in Engineering*, 2021.
- [19] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Education, 1st edition, 1997.
- [20] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.
- [21] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 1992.
- [22] Stuart J. Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of COLT*, 1998.
- [23] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of ICML*, 20–22 Jun 2016.
- [24] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of ICML*, 1999.
- [25] Eric Wiewiora, Garrison W. Cottrell, and Charles Elkan. Principled methods for advising reinforcement learning agents. In *Proceedings of ICML*, 2003.
- [26] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 2021.
- [27] Daochen Zha, Kwei-Herng Lai, Songyi Huang, Yuanpu Cao, Keerthana Reddy, Juan Vargas, Alex Nguyen, Ruzhe Wei, Junyu Guo, and Xia Hu. Rlcard: A platform for reinforcement learning in card games. In *Proceedings of IJCAI*, 2020.
- [28] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.
- [29] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, 2017.
- [30] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of ICML*, 07–09 Jul 2015.
- [31] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Proceedings of NeurIPS*, 2021.
- [32] Hervé Abdi. Bonferroni and Šidák corrections for multiple comparisons. In Neil J. Salkind, editor, *Encyclopedia of Measurement and Statistics*. Sage, 2007.
- [33] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 2017.
- [34] Weichao Zhou and Wencho Li. Rethinking inverse reinforcement learning: From data alignment to task alignment. In *Proceedings of NeurIPS*, 2024.
- [35] Sangwoong Yoon, Himchan Hwang, Dohyun Kwon, Yung-Kyun Noh, and Frank Park. Maximum entropy inverse reinforcement learning of diffusion models with energy-based models. *Advances in Neural Information Processing Systems*, 2024.