Loop-Free Inverse Reinforcement Learning via Sequential Value Recovery with Q-Score Matching

Yang Chen[1*] **Yitan Zhang**[2*] **Michael J. Witbrock**[2] **Shuyue Hu**[1†]
[1] Shanghai Artificial Intelligence Laboratory [2] University of Auckland
chenyang4@pjlab.org.cn, yzhb332@aucklanduni.ac.nz,
m.witbrock@auckland.ac.nz, hushuyue@pjlab.org.cn

Abstract

Inverse Reinforcement Learning (IRL) aims to recover a reward function that explains expert demonstrations. Existing IRL methods typically rely on a bi-level optimization procedure that alternates between reward learning and policy optimization, leading to substantial computational burden and training instability. In this work, we introduce a different route that eliminates policy optimization entirely by leveraging diffusion policies. Our key insight is that a diffusion policy encodes the action-gradient structure of the optimal soft Q function, enabling reward learning to be cast as a sequence of value recovery problems, thereby allowing us to bypass reward-policy loops inherent in prior IRL methods. Specifically, our method proceeds in three stages: (I) recovering the optimal soft Q function via action-gradient matching and estimating the corresponding soft value function (LogSumExp of Q values) in a way inspired by Gumbel regression; (II) calibrating these soft values by inferring a state-dependent offset; (III) extracting the reward by enforcing Bellman consistency. This leads to *Loop-Free Inverse Reinforcement Learning* (LFIRL), a fully offline algorithm that operates in a simple, loop-free, and sequential manner. LFIRL is simple to implement and significantly improves training efficiency while maintaining strong reward recovery performance. Empirically, across Maze, Franka Kitchen, Adroit Hand Pen, and Push-T benchmarks, LFIRL achieves **2-3x** speedup over the fastest baselines, while matching or surpassing state-of-the-art methods in reward recovery quality. [3]

1 Introduction

Inverse reinforcement learning (IRL) [27, 1] studies how to recover a reward function from expert demonstrations, with the goal of explaining expert behavior through an underlying optimization principle. It has become an important paradigm for reward design, behavior understanding, and transferable decision making [21, 34]. In practice, however, a major bottleneck in IRL is time efficiency due to the inherent *loop structure* underlying many classic and modern approaches [1, 50, 9, 10, 47]: the alternating update of the reward function and the policy. This loop arises because most IRL methods generally cannot extract reward-supervision signals directly from demonstrations. Instead, the reward is usually learned through differentiating expert behavior from the learner behavior. This mutual dependence naturally leads to a classic *bi-level* optimization process, where a reward function is updated by contrasting expert demonstrations with learner behavior, the learner policy is then updated under the updated reward, and this process is repeated until convergence. Such a looped training paradigm is computationally expensive, especially in high-dimensional tasks, and also makes

[*] Equal contribution.
[†] Corresponding author.
[3] The implementation is available at https://github.com/NothingThrough/LFIRL.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

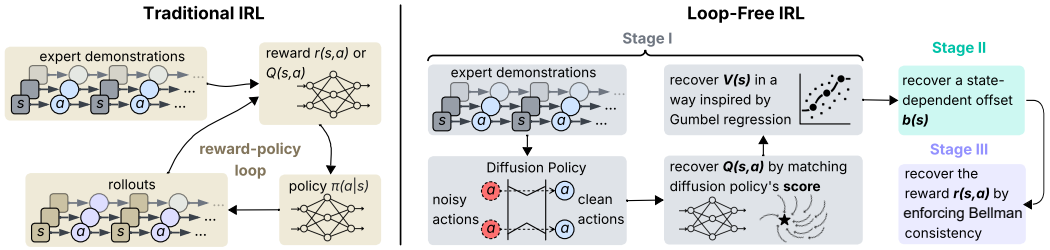

Figure 1: Traditional inverse reinforcement learning alternates between reward learning and policy optimization, while our loop-free version trains each component once without a reward-policy loop.

the overall training process harder to stabilize. This raises our central question: *can we design an IRL method without the reward-policy loop while still recovering a useful reward function?*

If we could extract reward-supervision signals directly from demonstrations, then reward recovery would no longer need to be mediated by a reward-policy loop. Intuitively, demonstrations already contain local information about which nearby actions are more or less preferable around expert behavior. *Diffusion policy* methods [5, 30, 33] provide a natural way to represent this information. Built on the recently burgeoning diffusion models [17, 36], a diffusion policy represents expert action distribution by transforming noise into actions through an iterative refinement process: it learns the gradient of the action distribution's *score function*, $\nabla_{\mathbf{a}} \log \pi^*(\mathbf{a}|\mathbf{s})$, that describes the direction in which actions should move towards expert ones, and iteratively optimizes wrt this gradient field via a series of stochastic steps. This score-based signal is exactly what we need to break the reward-policy loop. Specifically, in the commonly used maximum-entropy reinforcement learning [14] setting, this score function encodes information of the gradient of the expert's optimal soft Q function wrt actions, i.e., $\nabla_{\mathbf{a}} \log \pi^*(\mathbf{a}|\mathbf{s}) = \nabla_{\mathbf{a}} Q^*(\mathbf{a}, \mathbf{s})$. Therefore, by matching its action gradient, we can estimate an optimal soft Q function directly from demonstrations, without relying on the differential information from an iteratively updated learner policy.

Based on this observation, we propose a new IRL algorithm that avoids the conventional reward-policy loops and instead recovers the reward in a *fully sequential* manner (see Fig. 1). Concretely, it proceeds in three sequential stages: **(I)** This stage first recovers an optimal soft Q function by matching its action gradients to the score induced by a diffusion policy, where we use a Sobolev-style training mechanism [6] to favor expert actions over nearby perturbations; then estimates the corresponding soft value function (V = LogSumExp Q) in a way inspired by Gumbel regression [13] (Sec. 4.1). **(II)** For deriving calibrated value functions, the second stage infers a state-dependent offset function, appending which to the already estimated Q and V yields their more accurate estimates (Sec. 4.2). **(III)** The final stage recovers the reward function from the calibrated Q and V functions by enforcing Bellman consistency (Sec. 4.3). The entire pipeline is *fully offline*, using only a static set of trajectories. We empirically demonstrate that our resulting algorithm matches or surpasses state-of-the-art IRL baselines in reward recovery while significantly reducing training time (Sec. 5).

Our contributions are summarized as follows:

1. We introduce diffusion policy to IRL and propose the **Loop-Free Inverse Reinforcement Learning (LFIRL)** algorithm. It is a **fully offline** IRL algorithm that **avoids the computationally expensive looped reward-policy updates** in conventional IRL methods by leveraging the diffusion policy to realize a sequential value recovery procedure.
2. We conduct empirical studies on tasks including Maze, Franka Kitchen, Adroit Hand Pen, and Push-T, showing that our method matches or surpasses strong online and offline baselines in reward recovery while **improving training time efficiency by 2-3x** in most settings.

2 Related Work

Inverse reinforcement learning. IRL has been studied since the early formulations of reward recovery and apprenticeship learning [1, 27, 31, 50, 51], and was later extended to deep-learning variants such as Guided Cost Learning [9]. More recent adversarial formulations remain a dominant route for IRL and imitation learning, but they typically rely on a *nested loop* between reward/discriminator

learning and policy optimization, including GAIL [16], AIRL [10], DAC [22], and observation-only adversarial variants such as GAIfO [39]. A broader unifying view is provided by the moment-matching perspective of imitation learning [37], while empirical studies such as [29] further highlight the sensitivity of adversarial pipelines to design choices. Several recent methods reduce or modify this classical reward-policy loop: Maximum-Likelihood IRL [47] proposes a *single-loop* update, ValueDICE [23] avoids a separate RL optimization procedure through off-policy distribution matching, IQ-Learn [12] and LS-IQ [2] learn implicit Q-function formulations, PIRO [4] stabilizes non-adversarial reward-policy learning through trust-region reward optimization. Other approaches modify the imitation or policy-search process in different ways, including Coherent Soft Imitation Learning [41], successor-feature matching [20], FILTER [38], and Hybrid IRL [32]. In contrast to both nested-loop methods and single-loop approaches that tightly entangle reward and policy learning, our method adopts a *fully staged, sequential* IRL pipeline: it uses diffusion-derived local action-score information to recover soft values, and then recovers rewards.

Diffusion-Based Imitation Learning. Diffusion models have rapidly become strong policy classes for imitation learning, especially in robotics and multimodal control. Diffusion Policy [5] models actions through a conditional denoising process and has become a standard baseline for visuomotor behavior cloning, while subsequent work extends diffusion-based imitation learning through goal-conditioned policies [33], diffusion-augmented behavioral cloning [3], improved visual or self-supervised representations [25, 46], and architectural scaling studies [7, 49]. A smaller line of work uses diffusion models more directly for reward learning, for example by extracting reward-like quantities from diffusion models [28], learning rewards from expert videos [19], leveraging score information for policy optimization [30], or integrating diffusion into adversarial imitation pipelines as the *discriminator* such as DiffAIL [40], DRAIL [24], and DIFO [18]. Recent score-matching-based imitation methods such as [42] likewise emphasize the value of diffusion-derived gradient information. Our method differs from these directions in that we use a diffusion policy only as a score estimator for local supervision of soft Q-recovery, and then recover V and r sequentially, rather than using diffusion as the final policy class, extracting relative trajectory-level rewards from multiple diffusion models, or embedding diffusion inside an adversarial reward-policy loop.

3 Preliminaries

A Markov decision process (MDP) is defined by the tuple $(\mathcal{S}, \mathcal{S}_\perp, \mathcal{A}, P, \gamma, r)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $\mathcal{S}_\perp \subseteq \mathcal{S}$ is a set of absorbing states, $P(\cdot|\mathbf{s}, \mathbf{a})$ is the transition kernel, $\gamma \in (0, 1)$ is the discount factor, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. For any absorbing state $\mathbf{s}_\perp \in \mathcal{S}_\perp$, we have $P(\mathbf{s}_\perp|\mathbf{s}_\perp, \mathbf{a}) = 1$ and $r(\mathbf{s}_\perp, \mathbf{a}) = 0$ for all $\mathbf{a} \in \mathcal{A}$, which describes can-never-escape states, e.g., completions or failures of a game. Let $\pi(\mathbf{a}|\mathbf{s})$ be a stochastic policy, and let $\rho^\pi(\mathbf{s}, \mathbf{a}) \coloneqq \frac{1}{1-\gamma} \sum_{t=0}^{\infty} \Pr(\mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}|\pi, P)$ denote its induced state-action density.

3.1 Maximum-Entropy RL and IRL

The goal of classic RL is to find a policy to maximize the expected discounted long-term rewards $\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\rho^\pi}[r(\mathbf{s}, \mathbf{a})]$. We consider a generalized version of Maximum-entropy (MaxEnt) RL that augments the reward objective with the relative entropy $\mathcal{H}_\mu(\pi) \coloneqq \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\rho^\pi}[-\log \frac{\pi(\mathbf{a}|\mathbf{s})}{\mu(\mathbf{a}|\mathbf{s})}]$ between π and a reference policy μ: $\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\rho^\pi}[r(\mathbf{s}, \mathbf{a})] + \varepsilon\, \mathcal{H}_\mu(\pi)$, where $\varepsilon > 0$ is the temperature parameter. It recovers the standard MaxEnt RL [14] objective up to a constant when μ is uniform.

Under this formulation of MaxEnt RL, the optimal policy admits the energy-based form

$$\pi^*(\mathbf{a}|\mathbf{s}) = \mu(\mathbf{a}|\mathbf{s}) \exp\left(\tfrac{1}{\varepsilon}\big(Q^*(\mathbf{s}, \mathbf{a}) - V^*(\mathbf{s})\big)\right), \tag{1}$$

where the optimal soft value function $V^*(\mathbf{s})$ and optimal soft action value function $Q^*(\mathbf{s}, \mathbf{a})$ satisfy

$$V^*(\mathbf{s}) = \varepsilon \log \int_{\mathcal{A}} \exp\left(\tfrac{1}{\varepsilon} Q^*(\mathbf{s}, \mathbf{a})\right) d\mu(\mathbf{a}|\mathbf{s}), \quad Q^*(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}'\sim P}\left[V^*(\mathbf{s}')\right]. \tag{2}$$

Note that the MaxEnt optimality condition in Eq. (1) implies the following relation, $\pi^*(\mathbf{a}|\mathbf{s})/\mu(\mathbf{a}|\mathbf{s})$ and $Q^*(\mathbf{a}|\mathbf{s})$ have the same gradient wrt actions, which we will leverage to build our method upon:

$$\nabla_{\mathbf{a}} \log \tfrac{\pi^*(\mathbf{a}|\mathbf{s})}{\mu(\mathbf{a}|\mathbf{s})} = \tfrac{1}{\varepsilon} \nabla_{\mathbf{a}} Q^*(\mathbf{s}, \mathbf{a}). \tag{3}$$

In MaxEnt inverse RL (MaxEnt IRL), the reward function is unknown, while a set of expert demonstrations $\mathcal{D}_E = \{\tau_i\}_{i=1}^N$ is observed, where each trajectory $\tau = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \dots)$ is generated by an expert policy π_E. The goal is to recover a reward function under which π_E is optimal under the above MaxEnt RL framework, which can be interpreted by the following optimization problem:

$$\min_r \max_\pi \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\rho^\pi}[r(\mathbf{s},\mathbf{a})] + \varepsilon\,\mathcal{H}_\mu(\pi) - \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\rho^{\pi_E}}[r(\mathbf{s},\mathbf{a})]. \tag{4}$$

The min-max structure of Eq. (4) reveals an inherent *bi-level* optimization procedure of many classic and modern IRL methods [16, 8, 10, 48]: the upper level updates the reward by contrasting expert behavior with the learner behavior induced by the current reward, while the lower level updates the learner's imitation policy. Classic IRL methods typically implement this bi-level procedure as a computationally expensive *nested reward-policy loop* because the lower level is a full RL procedure for computing an optimal policy. Some recent methods [47, 48] improve time efficiency by introducing a relatively cheap *single reward-policy loop*, where the lower level conducts one or several steps of policy improvement. Although this bi-level optimization structure is well theoretically grounded [16, 47] and has been shown to be effective in applications [43, 11], its induced reward-policy loops introduce inevitable computational burden and instability to the optimization process.

3.2 Diffusion Models and Diffusion Policy

Diffusion models [17, 36] generate samples by modeling a *forward process* that gradually adds noise to data and a *reverse process* that iteratively removes this noise to recover data samples. *Diffusion policy* methods use this idea for imitating expert behavior, where a policy $\pi(\cdot|\mathbf{s}_t)$ is modeled as an action generation process: given an MDP time step t, let $\{\mathbf{a}_t^k\}_{k=0}^K$ denote the latent action sequence indexed by the diffusion step k, where $\mathbf{a}_t^0$ corresponds to the initial action (samples from expert demonstrations) and larger k corresponds to noisier ones. The forward process uses a variance schedule $\{\beta_k\}_{k=1}^K$ with $\alpha_k \coloneqq 1-\beta_k$, $\bar{\alpha}_k \coloneqq \prod_{j=1}^k \alpha_j$, and Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$; the noisy action at step k can be written in closed form as

$$\mathbf{a}_t^k = \sqrt{\bar{\alpha}_k}\,\mathbf{a}_t^0 + \sqrt{1-\bar{\alpha}_k}\,\boldsymbol{\epsilon}. \tag{5}$$

The reverse process trains a noise predictor $\boldsymbol{\epsilon}_{\boldsymbol{\phi}}$ by minimizing the following MSE loss:

$$\mathcal{L}_{\text{diff}}(\boldsymbol{\phi}) \coloneqq \mathbb{E}_{\substack{(\mathbf{s}_t,\mathbf{a}_t^0)\sim\rho^{\pi_E},\\ k\sim\text{Unif}\{1,\dots,K\},\,\boldsymbol{\epsilon}\sim\mathcal{N}(0,I)}} \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\phi}}\left(\mathbf{a}_t^k, \mathbf{s}_t, k\right)\right\|_2^2. \tag{6}$$

Training the noise predictor $\boldsymbol{\epsilon}_\phi$ can be interpreted as learning $\nabla_{\mathbf{a}} \log \pi_E(\mathbf{a}|\mathbf{s}_t)$, the so-called *score* of a diffusion policy, which is a vector field indicating the direction to adjust a noisy action to increase its likelihood under the expert action distribution. In the idealized limit of exact denoising, it can be shown that $\nabla_{\mathbf{a}} \log \pi_E(\mathbf{a}|\mathbf{s}_t)|_{\mathbf{a}=\mathbf{a}_t^k} = -\frac{1}{\sqrt{1-\bar{\alpha}_k}}\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{a}_t^k, \mathbf{s}_t, k) =: g_{\boldsymbol{\phi}}(\mathbf{a}_t^k, \mathbf{s}_t, k)$. Combining this with Eq. (3), under the condition that the expert policy is soft-optimal, i.e., $\pi_E = \pi^*$, we have

$$g_\phi(\mathbf{a}_t^k, \mathbf{s}_t, k) = \nabla_{\mathbf{a}} \log \pi_E(\mathbf{a}|\mathbf{s}_t) = \nabla_{\mathbf{a}} \log \mu(\mathbf{a}|\mathbf{s}_t)|_{\mathbf{a}=\mathbf{a}_t^k} + \tfrac{1}{\varepsilon}\nabla_{\mathbf{a}} Q_r^*(\mathbf{s}_t, \mathbf{a})|_{\mathbf{a}=\mathbf{a}_t^k}. \tag{7}$$

Thus, a pretrained diffusion policy can provide the action-gradient supervision signal to recover the optimal soft Q function. We will utilize this property to design our method. To stay self-contained, detailed derivations of the identity relationship in Eq. (7) are given in Appendix A.1.

4 Methods

In this section, we present **Loop-Free Inverse Reinforcement Learning (LFIRL)**, a fully offline IRL algorithm that avoids reward-policy loops in conventional IRL approaches. Instead, it proceeds in three sequential stages (see Fig. 1): **(I)** first recover a soft Q function from diffusion-policy using Sobolev training and recover the value function V from the learned Q in a way inspired by Gumbel regression; **(II)** then infer the state-dependent offset and calibrate the Q and V; **(III)** finally recover the reward from the learned Q and V by enforcing Bellman consistency.

4.1 Stage I: Recover the Soft Q Function and Value function

In this stage, we recover the optimal soft Q function through *Sobolev Training* [6, 35], which trains neural networks by jointly matching function values and their derivatives. In our formulation, the

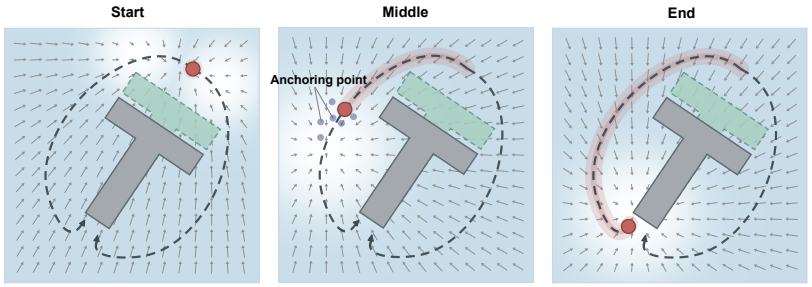

Figure 2: Illustration of matching policy's score on Push-T task, where the goal is to push the T-shaped block to the target pose. The red circle marks the agent's current position (initially in the Start panel), and the dark dashed curves indicate two optimal routes to reach the correct position for pushing the block. The arrows depict the action-gradient field supervised by the diffusion policy's score; the blue dots are perturbed actions whose Q values are constrained below that of the expert action by a margin, thereby providing anchoring information.

score function of the diffusion policy provides direct supervision on action-gradients. Specifically, we train a Q-network $\hat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \mathbf{a})$ such that its action-gradient matches the diffusion policy's score after correcting for the reference policy. Concretely, we align $\nabla_{\mathbf{a}} \hat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \mathbf{a})|_{\mathbf{a}=\mathbf{a}^k}$ with $g_{\boldsymbol{\phi}}(\mathbf{a}^k, \mathbf{s}, k) - \nabla_{\mathbf{a}} \log \mu(\mathbf{a}|\mathbf{s})|_{\mathbf{a}=\mathbf{a}^k}$, where $\mathbf{a}^k$ is obtained by adding noise to an expert action $\mathbf{a}^0$ via Eq. (5).

However, the derivation of value supervision is not straightforward, as the optimal Q values themselves are indeed what we seek for. To address this, we introduce an implicit form of value supervision by enforcing a small margin between the Q values of expert actions and those of their *locally* perturbed counterparts. We do not impose this margin globally, in order to preserve action diversity, as multiple actions may be optimal under the same state. Intuitively, this margin-based constraint effectively regularizes the Q function, using expert actions as "anchor points" to shape a more precise function surface. These settings, illustrated with Push-T task in Fig. 2, lead to the following loss function for training $\hat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \mathbf{a})$:

$$\begin{aligned} \mathcal{L}_{\hat{Q}}(\boldsymbol{\omega}) := \mathbb{E}_{\substack{\mathbf{s}\in\mathcal{S},\mathbf{a}\in\mathcal{A},\\ k\sim\mathrm{Unif}\{1,\dots,K\}}} \underbrace{\left[\left\| g_{\boldsymbol{\phi}}(\mathbf{a}^k, \mathbf{s}, k) - \nabla_{\mathbf{a}} \log \mu(\mathbf{a}|\mathbf{s}_t)|_{\mathbf{a}=\mathbf{a}_t^k} - \tfrac{1}{\varepsilon} \nabla_{\mathbf{a}} \hat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \mathbf{a})|_{\mathbf{a}=\mathbf{a}^k} \right\|_2^2\right]}_{\text{Gradient Matching}} \\ + \lambda\, \mathbb{E}_{\substack{(\mathbf{s},\mathbf{a})\sim\mathcal{D}_E,\\ \tilde{\mathbf{a}}\sim\eta(\cdot|\mathbf{a})}} \underbrace{\left[\mathrm{ReLU}\left(\xi - \left(\hat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \mathbf{a}) - \hat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \tilde{\mathbf{a}})\right)\right)\right]}_{\text{Value Anchoring}}, \end{aligned} \tag{8}$$

where $\xi > 0$ is an intended margin, $\eta(\cdot|\mathbf{a})$ denotes the neighborhood of action $\mathbf{a}$, $\mathrm{ReLU}(x) := \max\{0, x\}$ denotes the Rectified Linear Unit function, and $\lambda \geq 0$ is a weighting coefficient. In practice, we can instantiate the reference policy $\mu(\mathbf{a}|\mathbf{s})$ as a simple distribution (e.g., uniform over bounded actions or a Gaussian prior), whose score $\nabla_{\mathbf{a}} \log \mu(\mathbf{a}|\mathbf{s})$ is tractable.

After obtaining $\hat{Q}_{\boldsymbol{\omega}}$, we estimate the corresponding soft value function. However, direct computation of $\hat{V}(\mathbf{s}) = \varepsilon \log \int_{\mathcal{A}} \exp(\hat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \mathbf{a})/\varepsilon)\, d\mu(\mathbf{a}|\mathbf{s})$ is generally intractable in continuous action spaces. To address this, we reformulate the problem via an equivalent moment condition, drawing inspirations from *Gumbel regression* in RL [13], which models action selection by perturbing Q values with Gumbel distribution noise [15] and leverages *Gumbel-Max Trick* [15] to infer the value function as the expected stochastic maximum, yielding a LogSumExp form. Specifically, the soft value $\hat{V}(\mathbf{s})$ is the unique solution to $\mathbb{E}_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})}[\exp(\frac{\hat{Q}_{\boldsymbol{\omega}}(\mathbf{s},\mathbf{a}) - V(\mathbf{s})}{\varepsilon})] = 1$. This characterization allows us to estimate $\hat{V}(\mathbf{s})$ without explicitly computing the LogSumExp. We then construct a convex surrogate whose first-order condition recovers the above moment constraint:

$$\ell_V(v; \mathbf{s}) := \mathbb{E}_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})} \left[\exp\left(\frac{\hat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \mathbf{a}) - v}{\varepsilon} \right) - \frac{\hat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \mathbf{a}) - v}{\varepsilon} - 1 \right]. \tag{9}$$

This objective is strictly convex in v and admits a unique minimizer, which coincides with the soft value $\hat{V}(\mathbf{s})$. The derivation is provided in Appendix A.2. In practice, we freeze $\hat{Q}_{\boldsymbol{\omega}}$ and train a value

network $\hat{V}_{\boldsymbol{\chi}}$ on a set of trajectories $\mathcal{D}_S$ sampled from μ through the following empirical objective

$$\mathcal{L}_{\hat{V}}(\boldsymbol{\chi}) \coloneqq \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim D_S}\left[\exp(z) - z - 1\right], \quad z = \frac{\hat{Q}_{\boldsymbol{\omega}}(\mathbf{s},\mathbf{a}) - \hat{V}_{\boldsymbol{\chi}}(\mathbf{s})}{\varepsilon}. \tag{10}$$

Notably, although this objective coincides with the Gumbel regression loss [13], our derivation does not rely on a stochastic model for Q values, as $\hat{Q}_{\boldsymbol{\omega}}$ is fixed during training $\hat{V}_{\boldsymbol{\chi}}$; instead, it follows directly from the deterministic characterization of the LogSumExp under the reference policy μ.

4.2 Stage II: Infer the State-Dependent Offset and Calibrate Soft Values

The soft values recovered in Stage I are *uncalibrated*, as the action-gradient matching determines only how soft values change wrt actions, but does not reflect their state-dependent factors. This fact can be formally explained by the following proposition, whose justification is given in Appendix A.3.

Proposition 1. *Assume $\pi_E = \pi^*$ and that the learned diffusion policy's score is exact in the sense of Eq. (7). For each state $\mathbf{s}$, if the action-gradient matching term in Eq. (8) is minimized exactly, then there exists a state-dependent function $b^* : \mathcal{S} \to \mathbb{R}$ such that $Q^*(\mathbf{s},\mathbf{a}) = \hat{Q}_{\boldsymbol{\omega}}(\mathbf{s},\mathbf{a}) + b^*(\mathbf{s})$.*

This reflects the ambiguity caused by *shift invariance*: adding any *state-dependent offset* $b(\mathbf{s})$ leaves action-gradient unchanged, i.e., $\nabla_{\mathbf{a}}\hat{Q}_{\boldsymbol{\omega}}(\mathbf{s},\mathbf{a}) = \nabla_{\mathbf{a}}\big(\hat{Q}_{\boldsymbol{\omega}}(\mathbf{s},\mathbf{a}) + b(\mathbf{s})\big)$, making $\hat{Q}_{\boldsymbol{\omega}}$ identifiable only up to $b(\mathbf{s})$. We therefore introduce an offset network $b_{\boldsymbol{\psi}} : \mathcal{S} \to \mathbb{R}$ and define the calibrated soft Q function as $Q_{\boldsymbol{\omega},\boldsymbol{\psi}}(\mathbf{s},\mathbf{a}) \coloneqq \hat{Q}_{\boldsymbol{\omega}}(\mathbf{s},\mathbf{a}) + b_{\boldsymbol{\psi}}(\mathbf{s})$ and write the calibrated soft value function from Stage I as $\hat{V}_{\boldsymbol{\chi}}(\mathbf{s}) + b_{\boldsymbol{\psi}}(\mathbf{s})$. A key challenge is that the offset $b_{\boldsymbol{\psi}}$ is not identifiable from action-gradient information. While penalizing $b_{\boldsymbol{\psi}}^2$ can fix the shift ambiguity by selecting the minimum-norm solution among all equivalent offsets, it ignores the transition structure and may yield solutions that violate Bellman consistency. Therefore, we regularize the *Bellman-implied reward*,

$$\delta_{\boldsymbol{\omega},\boldsymbol{\chi},\boldsymbol{\psi}}(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1}) \coloneqq \hat{Q}_{\boldsymbol{\omega}}(\mathbf{s}_t,\mathbf{a}_t) + b_{\boldsymbol{\psi}}(\mathbf{s}_t) - \gamma\big(\hat{V}_{\boldsymbol{\chi}}(\mathbf{s}_{t+1}) + b_{\boldsymbol{\psi}}(\mathbf{s}_{t+1})\big), \tag{11}$$

which couples $b_{\boldsymbol{\psi}}$ across consecutive states and enforces consistency with the underlying dynamics. Intuitively, minimizing the squared norm of $\delta_{\boldsymbol{\omega},\boldsymbol{\chi},\boldsymbol{\psi}}$ penalizes inconsistent offset value shifts across transitions, encouraging the offset to vary coherently along trajectories.

In addition, absorbing states provide direct supervision: for any $\mathbf{s}_\perp \in \mathcal{S}_\perp$, we have $Q^*(\mathbf{s}_\perp,\mathbf{a}) = 0$ for all $\mathbf{a}$. [4] Thus, the resulting objective is

$$\mathcal{L}_b(\boldsymbol{\psi}) \coloneqq \mathbb{E}_{\substack{(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1})\\ \sim\mathcal{D}_E\cup\mathcal{D}_S}}[\underbrace{\delta^2_{\boldsymbol{\omega},\boldsymbol{\chi},\boldsymbol{\psi}} + \lambda_b\, b^2_{\boldsymbol{\psi}}(\mathbf{s}_t)}_{\text{Offset regularization}}] + \mathbb{E}_{\mathbf{s}_\perp\in\mathcal{S}_\perp,\mathbf{a}}[\underbrace{\big(\hat{Q}_{\boldsymbol{\omega}}(\mathbf{s}_\perp,\mathbf{a}) + b_{\boldsymbol{\psi}}(\mathbf{s}_\perp)\big)^2}_{\text{Absorbing states supervision}}], \tag{12}$$

where $\lambda_b \geq 0$ is a regularization coefficient, $\hat{Q}_{\boldsymbol{\omega}}$ and $\hat{V}_{\boldsymbol{\chi}}$ are fixed, and only $b_{\boldsymbol{\psi}}$ is updated.

After calibrating the soft Q function, we re-estimate the value function using the calibrated $Q_{\boldsymbol{\omega},\boldsymbol{\psi}}$. Specifically, we freeze $Q_{\boldsymbol{\omega},\boldsymbol{\psi}}$ and optimize the Gumbel regression objective again:

$$\mathcal{L}_V(\boldsymbol{\chi}') \coloneqq \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mu(\cdot|\mathbf{s})}\left[\exp(\bar{z}) - \bar{z} - 1\right], \qquad \bar{z} \coloneqq \frac{Q_{\boldsymbol{\omega},\boldsymbol{\psi}}(\mathbf{s}_t,\mathbf{a}_t) - V_{\boldsymbol{\chi}'}(\mathbf{s}_t)}{\varepsilon}. \tag{13}$$

This second value-fitting step is needed because the soft value function must be consistent with the offset-calibrated soft Q function.

4.3 Stage III: Recover the Reward Function

Given the calibrated soft Q function $Q_{\boldsymbol{\omega},\boldsymbol{\psi}}$ and value function $V_{\boldsymbol{\chi}'}$, reward recovery reduces to enforcing Bellman consistency. For a transition $(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1})$, we write the Bellman-implied reward target $r = Q_{\boldsymbol{\omega},\boldsymbol{\psi}}(\mathbf{s}_t,\mathbf{a}_t) - \gamma V_{\boldsymbol{\chi}'}(\mathbf{s}_{t+1})$. We parameterize the reward by a network $r_{\boldsymbol{\theta}}$ and learn it by minimizing Bellman residual errors. To stabilize training, we apply clipping and normalization:

$$\tilde{r} = \frac{\mathrm{clip}(r,-c_r,c_r) - m_r}{\sigma_r + \zeta_r}, \qquad \tilde{r}_{\boldsymbol{\theta}} = \frac{r_{\boldsymbol{\theta}}(\mathbf{s}_t,\mathbf{a}_t) - m_r}{\sigma_r + \zeta_r}, \tag{14}$$

where $c_r > 0$ is the clipping threshold, (m_r,σ_r) are the running mean and standard deviation of the clipped reward targets, and $\zeta_r > 0$ is a small numerical constant. The reward network is trained via

$$\mathcal{L}_r(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1})\sim\mathcal{D}_E\cup\mathcal{D}_S}\left[(\tilde{r}_{\boldsymbol{\theta}} - \tilde{r})^2\right]. \tag{15}$$

[4] This zero value is caused by the absorbing state itself, not by a particular action, so it is used to calibrate the state-dependent offset rather than to provide the action-level value supervision used in Stage I.

Algorithm 1 Loop-Free Inverse Reinforcement Learning (LFIRL)

Input: Expert demonstrations $\mathcal{D}_E$; initialized networks $\hat{Q}_{\boldsymbol{\omega}}$, $b_{\boldsymbol{\psi}}$, $V_{\boldsymbol{\chi}}$, and $r_{\boldsymbol{\theta}}$; reference policy μ.
Train a (or load a pre-trained) diffusion policy on $\mathcal{D}_E$. {**Stage I**}
Recover $\hat{Q}_{\boldsymbol{\omega}}(\mathbf{s}, \mathbf{a})$ on $\mathcal{D}_E$ via matching diffusion policy's score. ▷ Eq. (8)
Recover $V_{\boldsymbol{\chi}}(\mathbf{s})$ on $\mathcal{D}_S$ using Gumbel-regression-style value fitting. ▷ Eq. (10)
Recover state-dependent offset $b_{\boldsymbol{\psi}}(\mathbf{s})$ on $\mathcal{D}_E$ and $\mathcal{D}_S$. {**Stage II**} ▷ Eq. (12)
Calibrate $\hat{V}_{\boldsymbol{\chi}'}(\mathbf{s})$ by the calibrated Q values $\hat{Q}_{\boldsymbol{\omega}} + b_{\boldsymbol{\psi}}(\mathbf{s})$. ▷ Eq. (13)
Recover $r_{\boldsymbol{\theta}}$ on $\mathcal{D}_E$ and $\mathcal{D}_S$ by minimizing Bellman residual errors. {**Stage III**} ▷ Eq. (15)
Output: recovered reward function $r_{\boldsymbol{\theta}}$.

4.4 Algorithm Summary

The complete procedure of the proposed Loop-Free IRL framework is summarized in Algorithm 1. In contrast to traditional IRL pipelines, our method eliminates the reward-policy loop entirely: each stage is executed only once in a sequential manner, and every component is trained solely on the frozen outputs produced by preceding stages. Importantly, the algorithm is *fully offline*, as $\mathcal{D}_S$ is an offline dataset collected from the reference policy μ. Consequently, training requires neither environment interactions nor rollouts from a learned dynamics model during training. Overall, this loop-free offline design substantially improves training efficiency, as will be demonstrated experimentally.

5 Experiments

We evaluate LFIRL from two perspectives: **(1)** reward recovery quality and **(2)** training time efficiency.

5.1 Experimental Setup

We consider four different kinds of tasks (see Fig. 3): PointMaze with the UMaze, Medium, and Large settings; Franka Kitchen consists of four subtasks; Adroit Hand Pen; and the Push-T manipulation [5]. We compare against four classes of baselines: (1) *Non-Adversarial IRL* baselines: IQ-Learn [12] and ML-IRL [47]; (2) the *Adversarial IRL* baseline: AIRL [10]; (3) the *Offline IRL* baseline: Offline ML-IRL [48], CLARE [45], and ValueDICE [23]; and (4) *Diffusion-based GAIL* baselines: DRAIL [24] and DIFO [18]. All reported results are averaged over five random runs. All methods are trained with the same environment-step budget, and within each environment we use the same number of expert demonstrations for all methods. Full implementation details, including hyperparameters and network architectures, are in Appendix E.1.

To evaluate reward recovery on PointMaze and Franka Kitchen, we train an SAC [14] policy on the recovered reward, and then evaluate this policy in the original environment. We report success rate on PointMaze and the mean number of completed subtasks on Franka Kitchen. Since our method is designed to recover a reward function rather than directly output a policy, this setting provides a task-level assessment of whether the recovered reward induces the intended behavior. For the higher-dimensional Adroit Hand Pen and Push-T environments, we instead evaluate reward quality through discrimination: we construct balanced sets of high-quality and low-quality trajectories, score them with each learned reward model, and report the resulting classification accuracy. We additionally compare the recovered and ground-truth rewards on the same state-action samples using Pearson's and Spearman's correlation coefficients; the results are presented in Sec. 5.4. To evaluate efficiency, we record the training time required by each method to consume the same number of trajectory throughput, with each method run independently and without parallelized competition for compute resources. In addition to the main comparison, we also report an offset-ablation study and a demonstration-reduction study; the corresponding results are provided in Sec. 5.5 and Sec. 5.6.

5.2 Reward Recovery

The main results are shown in Fig. 4. Overall, LFIRL recovers highly effective rewards and exhibits strong cross-task robustness. Across various tasks, the recovered reward consistently supports competitive downstream control and achieves the best or near-best final performance in most settings.

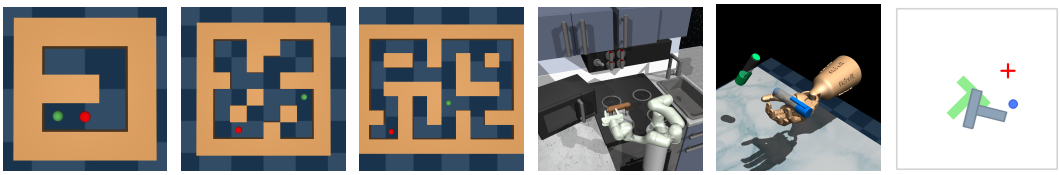

Figure 3: Illustrations of the evaluation tasks used in our experiments.

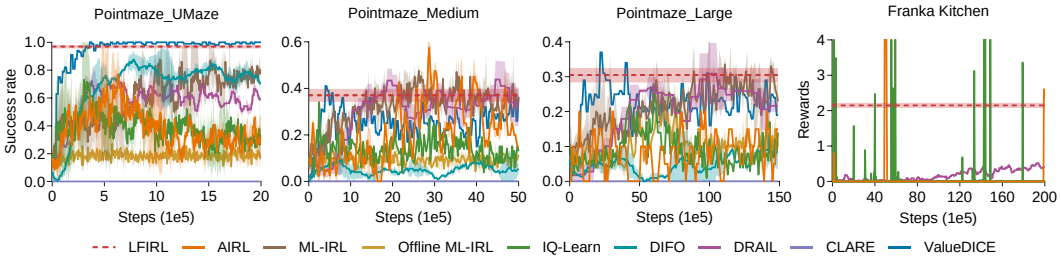

Figure 4: Reward-recovery results on PointMaze and Franka Kitchen. We report success rate on PointMaze and average completed subtasks on Franka Kitchen. Since LFIRL recovers the reward only at the final stage, its result is shown as a fixed final score rather than a policy-learning curve. For online IRL methods, "steps" denotes environment interaction steps. For offline IRL methods, "steps" denotes the number of transitions sampled from the offline dataset during training.

Even in cases where it does not attain the single best score, its performance remains highly competitive, indicating that the proposed sequential recovery pipeline yields stable and transferable reward signals.

For the higher-dimensional Push-T and Adroit Hand Pen, we instead evaluate whether the recovered reward can distinguish high-quality trajectories from low-quality ones. The results are reported in Tab. 1. These results further support a consistent overall picture: the reward recovered by LFIRL captures meaningful behavioral differences and generalizes well across qualitatively different evaluation protocols, including both downstream control and trajectory discrimination.

5.3 Average Running Time

Tab. 2 reports the average running time under the same step budget. Overall, LFIRL substantially reduces the time cost of training. Across tasks, it is consistently the fastest method, typically achieving about a 2–3x reduction in runtime compared with the fastest baseline in each setting, and a roughly 2–5x reduction when loading a pre-trained diffusion policy. These results confirm that removing the reward-policy optimization loop brings not only pipeline simplicity, but also a substantial practical gain in time efficiency.

5.4 Correlation with Ground-Truth Rewards

The preceding evaluations assess whether a recovered reward supports the intended behavior or distinguishes trajectories of different quality. We further evaluate reward recovery directly by comparing each learned reward with the environment's ground-truth reward on the same set of state-action samples. We report Pearson's correlation coefficient (PCC), which measures linear association between the recovered and ground-truth reward values, and Spearman's rank correlation coefficient (SCC), which measures agreement between their rankings. A higher PCC indicates that

Table 1: Trajectory-quality classification accuracy (%) on Push-T and Adroit Hand Pen using reward networks recovered by different IRL algorithms. Results are reported as mean ± standard deviation over five runs.

Task	LFIRL	AIRL	IQ-Learn	DIFO	DRAIL	ML-IRL	Offline ML-IRL	CLARE	ValueDICE
Push-T	**74.17±1.96**	59.67±2.08	39.67±0.58	**79.00±1.00**	49.67±1.15	71.33±2.33	42.67±1.20	71.00±3.67	**78.67±2.52**
Pen	**97.17±2.75**	90.00±2.45	74.33±7.59	91.67±2.05	**98.33±0.94**	71.00±4.32	66.00±12.43	94.67±0.58	95.67±1.15

Table 2: Average running time (in hours). "Load D.P." denotes LFIRL with a pre-trained diffusion policy, while "Train D.P." denotes LFIRL including diffusion-policy training time. "T.R." is short for "time reduction" and reports the relative time saved compared with the fastest baseline (marked with an underline).

Task	Online methods					Offline methods			LFIRL			
	AIRL	IQ-Learn	DIFO	DRAIL	ML-IRL	ML-IRL	CLARE	ValueDICE	Load D.P.	T.R. (SpeedUp)	Train D.P.	T.R. (SpeedUp)
UMaze	1.38	6.88	2.47	1.70	9.76	3.09	5.86	4.66	0.28	79.7% (5x)	0.43	68.8% (3x)
Maze(M)	3.67	10.10	7.05	2.65	13.80	4.99	5.98	4.87	0.77	70.9% (3x)	1.07	59.6% (2x)
Maze(L)	6.88	10.88	7.73	3.05	13.33	5.52	6.67	5.17	1.42	53.4% (2x)	1.80	41.0% (2x)
Kitchen	9.55	13.72	7.40	3.33	18.42	6.83	7.80	5.91	1.52	54.4% (2x)	1.87	43.8% (2x)
Push-T	1.57	8.55	4.05	1.95	9.50	2.10	3.20	5.30	0.35	77.7% (4x)	0.62	60.5% (3x)
Pen	4.83	14.88	10.83	7.50	18.52	6.53	3.79	4.71	1.18	68.9% (3x)	1.71	54.9% (2x)

Table 3: Direct comparison of recovered and ground-truth rewards on the same state-action samples. Each entry reports PCC / SCC; higher values indicate stronger linear association / rank agreement. Bold indicates the highest value for each metric in each task.

Method	UMaze PCC / SCC	Medium PCC / SCC	Large PCC / SCC	Kitchen PCC / SCC	Push-T PCC / SCC	Pen PCC / SCC	Average PCC / SCC
LFIRL (Ours)	**0.83 / 0.91**	**0.76** / 0.82	**0.84 / 0.87**	**0.67 / 0.81**	0.72 / **0.81**	**0.79 / 0.85**	**0.77 / 0.85**
AIRL	0.65 / 0.86	0.63 / 0.74	0.77 / 0.75	0.44 / 0.63	0.69 / 0.74	0.75 / 0.71	0.66 / 0.74
IQ-Learn	0.62 / 0.74	0.59 / 0.62	0.65 / 0.70	0.34 / 0.51	0.48 / 0.65	0.47 / 0.62	0.53 / 0.64
DIFO	0.69 / 0.79	0.53 / 0.67	0.68 / 0.81	0.61 / 0.77	**0.83** / 0.79	0.71 / 0.72	0.68 / 0.76
DRAIL	0.67 / 0.71	0.74 / **0.84**	0.69 / 0.78	0.56 / 0.76	0.51 / 0.56	0.63 / 0.62	0.63 / 0.71
ML-IRL	0.78 / 0.75	0.67 / 0.72	0.64 / 0.71	0.63 / 0.69	0.66 / 0.73	0.78 / 0.69	0.69 / 0.72
Offline ML-IRL	0.49 / 0.52	0.47 / 0.58	0.42 / 0.62	0.45 / 0.54	0.45 / 0.53	0.53 / 0.43	0.47 / 0.54
CLARE	0.72 / 0.80	0.68 / 0.74	0.49 / 0.61	0.24 / 0.46	0.51 / 0.79	0.58 / 0.55	0.54 / 0.66
ValueDICE	0.72 / 0.81	0.61 / 0.79	0.74 / 0.71	0.53 / 0.56	0.65 / 0.76	0.57 / 0.63	0.64 / 0.71

the recovered reward more closely follows the ground-truth reward's linear variation across samples; a higher SCC indicates that it better preserves which samples receive higher or lower reward. These measures assess agreement in structure and ordering, respectively, without requiring the recovered reward to have exactly the same scale or offset as the ground-truth reward.

As shown in Tab. 3, LFIRL obtains the highest PCC on five of the six tasks and the highest SCC on five of the six tasks. Thus, LFIRL's consistently strong results indicate that the sequential recovery procedure preserves both the linear variation and relative ordering of the ground-truth reward across a range of environments.

5.5 Ablation Study

We conduct an ablation study to evaluate the contribution of the state-dependent offset $b(\mathbf{s})$ in LFIRL. The results are shown in Fig. 5. Overall, removing $b(\mathbf{s})$ consistently degrades performance across the evaluated tasks, showing that inferring the offset and calibrating soft values are important for recovering a well-aligned value structure. This result supports our analysis in Sec. 4.2: calibrating this offset is necessary for obtaining a reliable downstream reward.

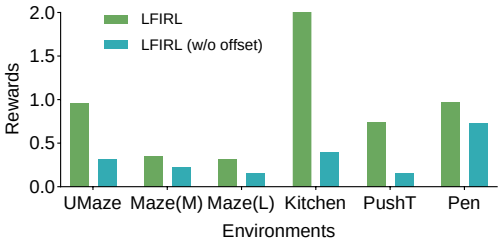

Figure 5: Ablation results on PointMaze, Kitchen, Pen, and Push-T. We compare LFIRL with its variant that removes the state-dependent offset $b(\mathbf{s})$.

5.6 Reduced-Data Experiments

We further study how LFIRL behaves when the number of expert demonstrations is reduced. The results are shown in Fig. 6, where we use the full dataset, one-half of the demonstrations, and one-quarter of the demonstrations. Overall, the performance of LFIRL decreases as the number of demonstrations becomes smaller. This trend is consistent with the design of our method: the first stage recovers the Q function from local supervision around expert actions, and the number of available expert state-action anchors directly affects how accurately this action-value structure can

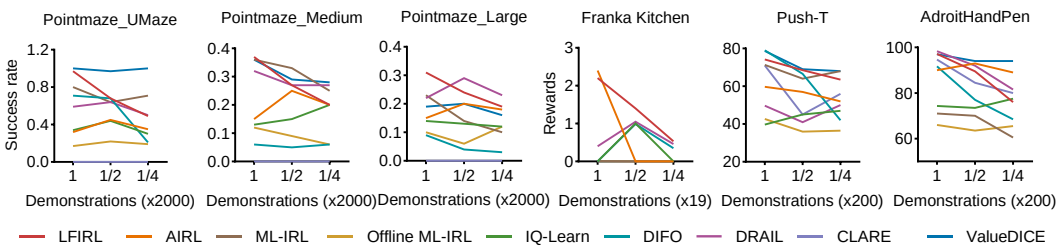

Figure 6: Reduced-data experiments with the full, one-half, and one-quarter of the demonstrations. We report success rate on PointMaze, average completed subtasks on Franka Kitchen, and reward on Push-T and Pen.

be reconstructed. When the demonstration set is reduced, anchor coverage becomes sparser, which weakens the quality of Q recovery and subsequently affects downstream value and reward recovery. Nevertheless, LFIRL remains competitive across the reduced-data settings.

6 Discussion

Source of the time efficiency gain. The efficiency of LFIRL mainly comes from its linear training pipeline, not merely from using offline training framework. Offline IRL methods can still be slow if they keep a reward-policy loop, since reward updates repeatedly depend on policy or value optimization under the current reward, no matter if the policy interacts with the environment. Even methods that reduce this cost, such as IQ-Learn [12], may avoid training an explicit policy in discrete action spaces but still rely on value or policy quantities induced by the current reward-related objective. LFIRL removes this repeated coupling with each stage using the frozen output of previous stages, and this is why LFIRL remains much faster than offline IRL methods in Tab. 2.

Choice of generative models. In fact, LFIRL uses a conditional *Denoising Diffusion Probabilistic Model* [17] (DDPM)-based diffusion policy to provide the derivative signal for Q function recovery, following common practice in diffusion-based imitation learning methods. In principle, any state-conditioned generative model that can provide a reliable score, or an action-gradient signal, can be used as the source of derivative supervision in Stage I. In Appendix B.1, we instantiate a version based on *Flow Matching* [26], another popular generative model paradigm. However, Appendix B.2 shows that, under the same number of training epochs, this variant gives weaker reward recovery. One possible reason is that the two pretraining objectives may have different convergence behavior under the same epoch budget. Conditional DDPM training directly optimizes denoising over multiple noise levels, which provides a strong and stable supervision signal for action refinement around expert demonstrations. Flow Matching instead learns a continuous transport vector field, whose action-generation quality can be more sensitive to integration accuracy, schedule design, and training budget. As a result, the pretrained diffusion policy is less accurate for giving the score signal used for Q function recovery, which then propagates to the final reward.

7 Conclusion

We propose LFIRL, a novel staged IRL framework that removes the conventional reward-policy optimization loop by turning reward recovery into a sequence of value-recovery problems. Concretely, LFIRL first recovers an optimal Q function from diffusion-policy score signals and estimates the soft value function via Gumbel-regression-style value fitting. It then infers a state-dependent offset to calibrate the recovered soft values and re-estimates the value function using the calibrated Q function. Finally, it recovers the reward function by enforcing Bellman consistency. Experiments on PointMaze, Franka Kitchen, Adroit Hand Pen, and Push-T show that LFIRL matches or surpasses strong baselines in reward recovery while substantially reducing training time, achieving 2-3x efficiency gains in most settings. **Limitations.** The quality of the recovered Q function depends on sufficient local anchors around expert actions. When anchor coverage is sparse, the method may require more demonstrations to reconstruct the action-value structure accurately. **Future work** may explore broader classes of generative models beyond DDPM and Flow Matching for providing action-gradient supervision, and develop more sample-efficient ways to recover reliable value and reward information from limited expert demonstrations.

Acknowledgments and Disclosure of Funding

This work was supported by the Natural, Artificial and Organisational Intelligence Institute (NAOInstitute) at the University of Auckland and the Shanghai Artificial Intelligence Laboratory.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004. 1, 2

[2] Firas Al-Hafez, Davide Tateo, Oleg Arenz, Guoping Zhao, and Jan Peters. Ls-iq: implicit reward regularization for inverse reinforcement learning. In *11th International Conference on Learning Representations, ICLR 2023*, 2023. 3

[3] Shang-Fu Chen, Hsiang-Chun Wang, Ming-Hao Hsu, Chun-Mao Lai, and Shao-Hua Sun. Diffusion model-augmented behavioral cloning. In *Forty-first International Conference on Machine Learning*, 2024. 3

[4] Yang Chen, Menglin Zou, Jiaqi Zhang, Yitan Zhang, Junyi Yang, Gael Gendron, Libo Zhang, Jiamou Liu, and Michael J. Witbrock. Trust region reward optimization and proximal inverse reward optimization algorithm. In *Advances in Neural Information Processing Systems*, 2025. 3

[5] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025. 2, 3, 7, 19

[6] Wojciech M Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. *Advances in neural information processing systems*, 30, 2017. 2, 4

[7] Sudeep Dasari, Oier Mees, Sebastian Zhao, Mohan Kumar Srirama, and Sergey Levine. The ingredients for robotic diffusion transformers. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15617–15625. IEEE, 2025. 3

[8] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016. 4

[9] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016. 1, 2

[10] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018. 1, 3, 4, 7

[11] Justin Fu, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. In *International Conference on Learning Representations*, 2023. 4

[12] Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. Iq-learn: Inverse soft-q learning for imitation. *Advances in Neural Information Processing Systems*, 34:4028–4039, 2021. 3, 7, 10

[13] Divyansh Garg, Joey Hejna, Matthieu Geist, and Stefano Ermon. Extreme q-learning: Maxent rl without entropy. In *The Eleventh International Conference on Learning Representations*, 2023. 2, 5, 6

[14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018. 2, 3, 7

[15] Tamir Hazan and Tommi Jaakkola. On the partition function and random maximum a-posteriori perturbations. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1667–1674, 2012. 5

[16] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016. 3, 4

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2, 4, 10

[18] Bo-Ruei Huang, Chun-Kai Yang, Chun-Mao Lai, Dai-Jie Wu, and Shao-Hua Sun. Diffusion imitation from observation. *Advances in Neural Information Processing Systems*, 37:137190–137217, 2024. 3, 7

[19] Tao Huang, Guangqi Jiang, Yanjie Ze, and Huazhe Xu. Diffusion reward: Learning rewards via conditional video diffusion. *European Conference on Computer Vision (ECCV)*, 2024. 3

[20] Arnav Kumar Jain, Harley Wiltzer, Jesse Farebrother, Irina Rish, Glen Berseth, and Sanjiban Choudhury. Non-adversarial inverse reinforcement learning via successor feature matching. In *The Thirteenth International Conference on Learning Representations*, 2025. 3

[21] Mrinal Kalakrishnan, Peter Pastor, Ludovic Righetti, and Stefan Schaal. Learning objective functions for manipulation. In *2013 IEEE International Conference on Robotics and Automation*, pages 1331–1336. IEEE, 2013. 1

[22] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International Conference on Learning Representations*, 2019. 3

[23] Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. *arXiv preprint arXiv:1912.05032*, 2019. 3, 7

[24] Chun-Mao Lai, Hsiang-Chun Wang, Ping-Chun Hsieh, Yu-Chiang F Wang, Min-Hung Chen, and Shao-Hua Sun. Diffusion-reward adversarial imitation learning. *Advances in Neural Information Processing Systems*, 37:95456–95487, 2024. 3, 7

[25] Xiang Li, Varun Belagali, Jinghuan Shang, and Michael S Ryoo. Crossway diffusion: Improving diffusion-based visuomotor policy via self-supervised learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16841–16849. IEEE, 2024. 3

[26] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. 10, 16

[27] Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000. 1, 2

[28] Felipe Nuti, Tim Franzmeyer, and João F Henriques. Extracting reward functions from diffusion models. *Advances in Neural Information Processing Systems*, 36:50196–50220, 2023. 3

[29] Manu Orsini, Anton Raichuk, Léonard Hussenot, Damien Vincent, Robert Dadashi, Sertan Girgin, Matthieu Geist, Olivier Bachem, Olivier Pietquin, and Marcin Andrychowicz. What matters for adversarial imitation learning? *Advances in Neural Information Processing Systems*, 34:14656–14668, 2021. 3

[30] Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. In *Forty-first International Conference on Machine Learning*, 2024. 2, 3

[31] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736, 2006. 2

[32] Juntao Ren, Gokul Swamy, Zhiwei Steven Wu, J Andrew Bagnell, and Sanjiban Choudhury. Hybrid inverse reinforcement learning. *arXiv preprint arXiv:2402.08848*, 2024. 3

[33] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023. 2, 3

[34] Sahand Sharifzadeh, Ioannis Chiotellis, Rudolph Triebel, and Daniel Cremers. Learning to drive using inverse reinforcement learning and deep q-networks. *arXiv preprint arXiv:1612.03653*, 2016. 1

[35] Hwijae Son, Jin Woo Jang, Woo Jin Han, and Hyung Ju Hwang. Sobolev training for physics informed neural networks. *arXiv preprint arXiv:2101.08932*, 2021. 4

[36] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. 2, 4

[37] Gokul Swamy, Sanjiban Choudhury, J Andrew Bagnell, and Steven Wu. Of moments and matching: A game-theoretic framework for closing the imitation gap. In *International Conference on Machine Learning*, pages 10022–10032. PMLR, 2021. 3

[38] Gokul Swamy, David Wu, Sanjiban Choudhury, Drew Bagnell, and Steven Wu. Inverse reinforcement learning without reinforcement learning. In *International Conference on Machine Learning*, pages 33299–33318. PMLR, 2023. 3

[39] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018. 3

[40] Bingzheng Wang, Guoqiang Wu, Teng Pang, Yan Zhang, and Yilong Yin. Diffail: Diffusion adversarial imitation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 15447–15455, 2024. 3

[41] Joe Watson, Sandy Huang, and Nicolas Heess. Coherent soft imitation learning. *Advances in Neural Information Processing Systems*, 36:14540–14583, 2023. 3

[42] Runzhe Wu, Yiding Chen, Gokul Swamy, Kianté Brantley, and Wen Sun. Diffusing states and matching scores: A new framework for imitation learning. In *The Thirteenth International Conference on Learning Representations*, 2025. 3

[43] Zheng Wu, Liting Sun, Wei Zhan, Chenyu Yang, and Masayoshi Tomizuka. Efficient sampling-based maximum entropy inverse reinforcement learning with application to autonomous driving. *IEEE Robotics and Automation Letters*, 5(4):5355–5362, 2020. 4

[44] Omar G. Younis, Rodrigo Perez-Vicente, John U. Balis, Will Dudley, Alex Davey, and Jordan K Terry. Minari, September 2024. URL https://doi.org/10.5281/zenodo.13767625. 19

[45] Sheng Yue, Guanbo Wang, Wei Shao, Zhaofeng Zhang, Sen Lin, Ju Ren, and Junshan Zhang. Clare: Conservative model-based reward learning for offline inverse reinforcement learning. In *11th International Conference on Learning Representations, ICLR 2023*, 2023. 7

[46] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024. 3

[47] Siliang Zeng, Chenliang Li, Alfredo Garcia, and Mingyi Hong. Maximum-likelihood inverse reinforcement learning with finite-time guarantees. *Advances in Neural Information Processing Systems*, 35:10122–10135, 2022. 1, 3, 4, 7

[48] Siliang Zeng, Chenliang Li, Alfredo Garcia, and Mingyi Hong. When demonstrations meet generative world models: A maximum likelihood framework for offline inverse reinforcement learning. *Advances in Neural Information Processing Systems*, 36:65531–65565, 2023. 4, 7

[49] Minjie Zhu, Yichen Zhu, Jinming Li, Junjie Wen, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, et al. Scaling diffusion policy in transformer to 1 billion parameters for robotic manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10838–10845. IEEE, 2025. 3

[50] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008. 1, 2

[51] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Modeling interaction via the principle of maximum causal entropy. In *Proceedings of the 27th International Conference on Machine Learning*, pages 1255–1262, 2010. 2

Appendices

A Detailed Derivations

A.1 Derivation of Diffusion Scores and Their Relation to Soft Q-Gradients

In this appendix, we derive the identities underlying Section 3. We use t for the MDP time index and k for the diffusion step. For a fixed state $\mathbf{s}_t$, let $\mathbf{a}_t^0 \sim \pi_E(\cdot \mid \mathbf{s}_t)$ denote a clean expert action, and let $\mathbf{a}_t^k$ denote its noisy version at diffusion step k.

The DDPM forward process is defined by

$$q(\mathbf{a}_t^k \mid \mathbf{a}_t^{k-1}) = \mathcal{N}\Big(\mathbf{a}_t^k; \sqrt{1-\beta_k}\,\mathbf{a}_t^{k-1}, \beta_k I\Big), \qquad \alpha_k \coloneqq 1-\beta_k, \qquad \bar{\alpha}_k \coloneqq \prod_{j=1}^{k} \alpha_j.$$

This gives the closed-form marginal

$$q(\mathbf{a}_t^k \mid \mathbf{a}_t^0) = \mathcal{N}\big(\mathbf{a}_t^k; \sqrt{\bar{\alpha}_k}\,\mathbf{a}_t^0, (1-\bar{\alpha}_k) I\big),$$

or equivalently,

$$\mathbf{a}_t^k = \sqrt{\bar{\alpha}_k}\,\mathbf{a}_t^0 + \sqrt{1-\bar{\alpha}_k}\,\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, I). \tag{16}$$

This is the same forward noising relation used in Eq. (5).

For each fixed $(\mathbf{s}_t, k)$, let $p_k(\mathbf{a} \mid \mathbf{s}_t)$ denote the conditional density of noisy actions at diffusion step k, induced by sampling $\mathbf{a}_t^0 \sim \pi_E(\cdot \mid \mathbf{s}_t)$ and then applying Eq. (16). The score of this noisy action density is

$$\nabla_{\mathbf{a}} \log p_k(\mathbf{a} \mid \mathbf{s}_t).$$

The diffusion policy is trained by the noise-prediction objective

$$\mathcal{L}_{\text{diff}}(\boldsymbol{\phi}) = \mathbb{E}_{\substack{(\mathbf{s}_t, \mathbf{a}_t^0) \sim \rho^{\pi_E} \\ k \sim \text{Unif}\{1,\dots,K\},\, \boldsymbol{\epsilon} \sim \mathcal{N}(0,I)}} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\phi}}\left(\mathbf{a}_t^k, \mathbf{s}_t, k\right) \right\|_2^2.$$

For a fixed $(\mathbf{a}_t^k, \mathbf{s}_t, k)$, the pointwise minimizer of this MSE objective is

$$\boldsymbol{\epsilon}^*_{\boldsymbol{\phi}}(\mathbf{a}_t^k, \mathbf{s}_t, k) = \mathbb{E}\big[\boldsymbol{\epsilon} \mid \mathbf{a}_t^k, \mathbf{s}_t, k\big].$$

Using the Gaussian corruption in Eq. (16), the noisy-action score satisfies

$$\nabla_{\mathbf{a}} \log p_k(\mathbf{a} \mid \mathbf{s}_t)\big|_{\mathbf{a}=\mathbf{a}_t^k} = -\frac{1}{\sqrt{1-\bar{\alpha}_k}}\, \mathbb{E}\big[\boldsymbol{\epsilon} \mid \mathbf{a}_t^k, \mathbf{s}_t, k\big].$$

Therefore, the learned noise predictor induces the score estimator

$$g_{\boldsymbol{\phi}}(\mathbf{a}_t^k, \mathbf{s}_t, k) \coloneqq -\frac{1}{\sqrt{1-\bar{\alpha}_k}} \boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{a}_t^k, \mathbf{s}_t, k) \approx \nabla_{\mathbf{a}} \log p_k(\mathbf{a} \mid \mathbf{s}_t)\big|_{\mathbf{a}=\mathbf{a}_t^k}. \tag{17}$$

In the idealized limit of exact denoising, the approximation in Eq. (17) becomes exact for the noisy conditional density $p_k(\mathbf{a} \mid \mathbf{s}_t)$.

For small diffusion noise, $\mathbf{a}_t^k$ is close to the clean action $\mathbf{a}_t^0$, and the noisy-action score approximates the clean expert-policy score in a local neighborhood of the expert action:

$$g_{\boldsymbol{\phi}}(\mathbf{a}_t^k, \mathbf{s}_t, k) \approx \nabla_{\mathbf{a}} \log \pi_E(\mathbf{a} \mid \mathbf{s}_t)\big|_{\mathbf{a}=\mathbf{a}_t^k}.$$

If the expert policy is soft-optimal, $\pi_E = \pi^*$. Under the reference-policy MaxEnt formulation in Eq. (1), we have

$$\pi^*(\mathbf{a} \mid \mathbf{s}) = \mu(\mathbf{a} \mid \mathbf{s}) \exp\big(\tfrac{1}{\varepsilon}\big(Q^*(\mathbf{s}, \mathbf{a}) - V^*(\mathbf{s})\big)\big).$$

Taking the action gradient gives

$$\nabla_{\mathbf{a}} \log \pi^*(\mathbf{a} \mid \mathbf{s}) = \nabla_{\mathbf{a}} \log \mu(\mathbf{a} \mid \mathbf{s}) + \tfrac{1}{\varepsilon} \nabla_{\mathbf{a}} Q^*(\mathbf{s}, \mathbf{a}),$$

because $V^*(\mathbf{s})$ does not depend on $\mathbf{a}$. Equivalently,

$$\nabla_{\mathbf{a}} \log \tfrac{\pi^*(\mathbf{a}|\mathbf{s})}{\mu(\mathbf{a}|\mathbf{s})} = \tfrac{1}{\varepsilon} \nabla_{\mathbf{a}} Q^*(\mathbf{s}, \mathbf{a}),$$

which is Eq. (3).

Combining the low-noise score approximation with Eq. (3), we obtain

$$g_{\boldsymbol{\phi}}(\mathbf{a}_t^k, \mathbf{s}_t, k) - \nabla_{\mathbf{a}} \log \mu(\mathbf{a} \mid \mathbf{s}_t)\big|_{\mathbf{a}=\mathbf{a}_t^k} \approx \tfrac{1}{\varepsilon} \nabla_{\mathbf{a}} Q^*(\mathbf{s}_t, \mathbf{a})\big|_{\mathbf{a}=\mathbf{a}_t^k}. \tag{18}$$

This is the identity used in Eq. (8): the diffusion-policy score, after subtracting the reference-policy score, provides the action-gradient supervision for recovering the soft Q-function.

A.2 Derivations for Stage I: Q-Recovery and Initial Value Fitting

In this subsection, we justify the first-stage recovery of the uncalibrated soft Q-function and the initial value function. The key point is that matching action derivatives recovers the action-dependent geometry of Q^*, but cannot determine its state-dependent offset.

We next justify the Gumbel-regression objective used to obtain the initial value estimate from the frozen $\hat{Q}_{\boldsymbol{\omega}}$. For a fixed state $\mathbf{s}$ and a fixed Q function $\widetilde{Q}$, define

$$\ell_V(v;\mathbf{s}) := \mathbb{E}_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})}\left[\exp\left(\frac{\widetilde{Q}(\mathbf{s},\mathbf{a})-v}{\varepsilon}\right) - \frac{\widetilde{Q}(\mathbf{s},\mathbf{a})-v}{\varepsilon} - 1\right].$$

Differentiating with respect to v gives

$$\frac{\partial}{\partial v}\ell_V(v;\mathbf{s}) = -\frac{1}{\varepsilon}\mathbb{E}_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})}\left[\exp\left(\frac{\widetilde{Q}(\mathbf{s},\mathbf{a})-v}{\varepsilon}\right)\right] + \frac{1}{\varepsilon}.$$

The first-order condition is therefore

$$\mathbb{E}_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})}\left[\exp\left(\frac{\widetilde{Q}(\mathbf{s},\mathbf{a})-v}{\varepsilon}\right)\right] = 1.$$

Rearranging gives

$$v = \varepsilon\log\mathbb{E}_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})}\left[\exp\left(\frac{1}{\varepsilon}\widetilde{Q}(\mathbf{s},\mathbf{a})\right)\right].$$

Moreover,

$$\frac{\partial^2}{\partial v^2}\ell_V(v;\mathbf{s}) = \frac{1}{\varepsilon^2}\mathbb{E}_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})}\left[\exp\left(\frac{\widetilde{Q}(\mathbf{s},\mathbf{a})-v}{\varepsilon}\right)\right] > 0,$$

so the minimizer is unique.

In Stage I, we instantiate $\widetilde{Q}$ as the frozen action value function $\hat{Q}_{\boldsymbol{\omega}}$. Replacing the expectation over $\mu(\cdot \mid \mathbf{s})$ by actions gives

$$\mathcal{L}_V(\boldsymbol{\chi}) = \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mu(\cdot|\mathbf{s})}\left[\exp(z) - z - 1\right], \qquad z = \frac{\hat{Q}_{\boldsymbol{\omega}}(\mathbf{s}_t,\mathbf{a}_t) - \hat{V}_{\boldsymbol{\chi}}(\mathbf{s}_t)}{\varepsilon}.$$

This is Eq. (10). Since $\hat{Q}_{\boldsymbol{\omega}}$ is fixed in this step, gradients are taken only with respect to $\boldsymbol{\chi}$. The resulting value network $\hat{V}_{\boldsymbol{\chi}}$ is the initial value estimate used in Stage II for offset calibration.

A.3 Derivations for Stage II: Offset Calibration and Value Re-estimation

Proof of Proposition 1. Fix any state $\mathbf{s}$ and a connected action neighborhood on which the low-noise diffusion-score supervision is reliable. If the derivative-matching term in Eq. (8) is minimized exactly, then for every initial action point $\mathbf{a}$ in this neighborhood,

$$\frac{1}{\varepsilon}\nabla_{\mathbf{a}}\hat{Q}_{\boldsymbol{\omega}}(\mathbf{s},\mathbf{a}) = g_\phi(\mathbf{a}^k,\mathbf{s},k),$$

where $\mathbf{a}^k$ denotes the low-noise corrupted version of $\mathbf{a}$. By Eq. (7), under the idealized assumptions that $\pi_E = \pi^*$ and the diffusion score is exact,

$$g_\phi(\mathbf{a}^k,\mathbf{s},k) = \frac{1}{\varepsilon}\nabla_{\mathbf{a}}Q^*(\mathbf{s},\mathbf{a}).$$

Thus,

$$\nabla_{\mathbf{a}}\hat{Q}_{\boldsymbol{\omega}}(\mathbf{s},\mathbf{a}) = \nabla_{\mathbf{a}}Q^*(\mathbf{s},\mathbf{a}).$$

Since the gradients match only with respect to the action variable, the two functions can differ by a term that depends on the state but not on the action. Therefore, there exists a state-dependent function $b^* : \mathcal{S} \to \mathbb{R}$ such that

$$Q^*(\mathbf{s},\mathbf{a}) = \hat{Q}_{\boldsymbol{\omega}}(\mathbf{s},\mathbf{a}) + b^*(\mathbf{s})$$

on the sampled action neighborhood. This proves Proposition 1. □

B Flow-Matching Version of Diffusion Policy

B.1 Flow-Matching Policy and Score Estimation

The main method uses a DDPM-based diffusion policy to provide the score estimator $g_\phi(\mathbf{a}_t^k, \mathbf{s}_t, k)$. To keep the presentation concise, this subsection follows the simplified case that the reference policy μ is uniform. In this case, $\nabla_{\mathbf{a}} \log \mu(\mathbf{a} \mid \mathbf{s}) = 0$ inside the support, so the generative-model score can be directly used as the action-gradient signal for Stage I. This is not the only possible generative pretraining choice. A Flow-Matching-based model can also be used as a conditional action generator, and under a simple Gaussian interpolation path it can provide a score estimator that plays the same role in Stage I. This section explains this replacement.

Flow Matching trains a time-dependent vector field that transports samples from a simple noise distribution to the data distribution [26]. In the policy setting, the data distribution is the expert action distribution conditioned on the current state. Let $u \in [0, 1]$ denote the flow time, where $u = 0$ corresponds to Gaussian noise and $u = 1$ corresponds to expert actions. Given an expert pair $(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}_E$ and Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$, the interpolation is defined as:

$$\mathbf{a}_t^u = (1 - u)\boldsymbol{\epsilon} + u\mathbf{a}_t. \tag{19}$$

Here, $\mathbf{a}_t^u$ is an intermediate noisy action along the path from noise to the expert action. The derivative of this path with respect to u is

$$\frac{d}{du}\mathbf{a}_t^u = \mathbf{a}_t - \boldsymbol{\epsilon}.$$

A Flow-Matching policy therefore learns a state-conditioned velocity field $v_\phi(\mathbf{a}_t^u, \mathbf{s}_t, u)$ by the regression objective

$$\mathcal{L}_{\mathrm{FM}}(\phi) := \mathbb{E}_{\substack{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}_E \\ u \sim \mathrm{Unif}(0,1),\, \boldsymbol{\epsilon} \sim \mathcal{N}(0,I)}} \left[\| v_\phi(\mathbf{a}_t^u, \mathbf{s}_t, u) - (\mathbf{a}_t - \boldsymbol{\epsilon}) \|_2^2 \right]. \tag{20}$$

After training, action generation can be performed by starting from Gaussian noise and integrating

$$\frac{d}{du}\mathbf{a}^u = v_\phi(\mathbf{a}^u, \mathbf{s}_t, u)$$

from $u = 0$ to $u = 1$. The resulting endpoint is used as an action sampled from the learned policy.

For LFIRL, we need not only a sampling procedure, but also a score signal for action-gradient matching. Let $p_u(\mathbf{a} \mid \mathbf{s}_t)$ denote the conditional density of the intermediate action $\mathbf{a}_t^u$ at flow time u. Its score is

$$\nabla_{\mathbf{a}} \log p_u(\mathbf{a} \mid \mathbf{s}_t),$$

which indicates how an intermediate action should be adjusted to move toward higher-probability expert actions under state $\mathbf{s}_t$.

Under the interpolation defined in Eq. (19), this score can be estimated from the learned velocity field. If the learned velocity is exact, then at a fixed $(\mathbf{a}_t^u, \mathbf{s}_t, u)$ it estimates the conditional mean of the path derivative,

$$v_\phi(\mathbf{a}_t^u, \mathbf{s}_t, u) \approx \mathbb{E}[\mathbf{a}_t - \boldsymbol{\epsilon} \mid \mathbf{a}_t^u, \mathbf{s}_t, u].$$

Combining this relation with the interpolation equation gives

$$\mathbb{E}[\boldsymbol{\epsilon} \mid \mathbf{a}_t^u, \mathbf{s}_t, u] \approx \mathbf{a}_t^u - u\, v_\phi(\mathbf{a}_t^u, \mathbf{s}_t, u).$$

For the same Gaussian path, the score of $p_u(\mathbf{a} \mid \mathbf{s}_t)$ can be written in terms of this conditional noise estimate. Thus, for $u < 1$, we obtain the Flow-Matching score estimator

$$g_\phi^{\mathrm{FM}}(\mathbf{a}_t^u, \mathbf{s}_t, u) := -\frac{\mathbf{a}_t^u - u\, v_\phi(\mathbf{a}_t^u, \mathbf{s}_t, u)}{1 - u} \approx \nabla_{\mathbf{a}} \log p_u(\mathbf{a} \mid \mathbf{s}_t)\big|_{\mathbf{a}=\mathbf{a}_t^u}. \tag{21}$$

The expression is used for $u < 1$; at $u = 1$, the path reaches the expert action endpoint and the denominator vanishes. In practice, the analogue of the low-noise DDPM regime is to use flow times close to, but not exactly equal to, 1.

This gives the connection needed by LFIRL. When u is close to 1, the intermediate action $\mathbf{a}_t^u$ is close to the initial expert action. Therefore, $g_\phi^{\mathrm{FM}}(\mathbf{a}_t^u, \mathbf{s}_t, u)$ estimates the local score of the expert action distribution around that action. Thus, in the low-noise regime, Eq. (3) gives

$$g_\phi^{\mathrm{FM}}(\mathbf{a}_t^u, \mathbf{s}_t, u) \approx \tfrac{1}{\varepsilon} \nabla_{\mathbf{a}} Q^*(\mathbf{s}_t, \mathbf{a})\big|_{\mathbf{a}=\mathbf{a}_t^u}. \tag{22}$$

Therefore, a Flow-Matching-based diffusion policy can be used in Stage I by replacing the DDPM score estimator $g_\phi(\mathbf{a}_t^k, \mathbf{s}_t, k)$ in Eq. (8) with $g_\phi^{\mathrm{FM}}(\mathbf{a}_t^u, \mathbf{s}_t, u)$.

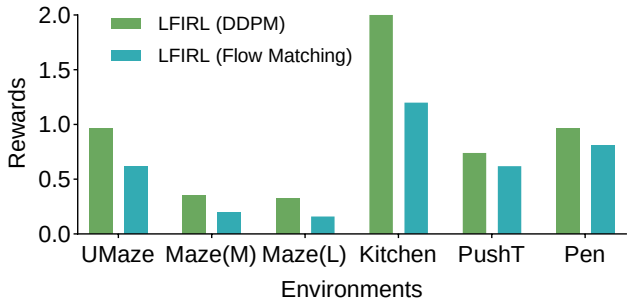

Figure 7: Additional ablation comparing DDPM-based and Flow-Matching-based pretraining in LFIRL. The two variants use the same subsequent sequential reward-recovery pipeline, but differ in how the pretrained generative policy is obtained. Higher is better.

B.2 Additional Results: DDPM vs. Flow Matching Pretraining

In the main experiments, LFIRL uses a DDPM-based diffusion policy to provide the score signal for recovering the soft Q-function. We also evaluate an alternative implementation in which the diffusion-policy pretraining stage is replaced by a Flow Matching version, while keeping the remaining LFIRL pipeline unchanged. This comparison isolates the effect of the pretrained generative policy used to provide local action-gradient supervision.

As shown in Fig. 7, under the same number of pretraining epochs, the Flow-Matching-based variant generally performs worse than the DDPM-based variant. This suggests that, in our setting, DDPM pretraining provides a more effective score signal for the subsequent Q-recovery stage. Since LFIRL relies on the pretrained generative policy to supply local action-gradient information, a weaker pretraining stage can directly reduce the quality of the recovered Q-function and subsequently degrade the final recovered reward.

C Sensitivity to the Reference Policy

The reference policy μ specifies the behavioral baseline relative to which the expert policy is interpreted. Under the reference-policy MaxEnt formulation,

$$\nabla_{\mathbf{a}} \log \pi_E(\mathbf{a} \mid \mathbf{s}) = \nabla_{\mathbf{a}} \log \mu(\mathbf{a} \mid \mathbf{s}) + \frac{1}{\varepsilon} \nabla_{\mathbf{a}} Q^*(\mathbf{s}, \mathbf{a}). \tag{23}$$

We examine how the choice of μ affects LFIRL by comparing a uniform reference policy with a state-conditioned Gaussian reference policy.

The two variants use the same expert demonstrations and LFIRL training configuration. The uniform variant uses a uniform distribution over the bounded action space, for which $\nabla_{\mathbf{a}} \log \mu(\mathbf{a} \mid \mathbf{s}) = 0$ in the interior of its support. For the Gaussian variant, we initialize an SAC actor before LFIRL training and freeze it throughout the reward-recovery pipeline. The actor outputs the mean and standard deviation of a state-conditioned action distribution and is not fitted to the expert demonstrations. At each noisy action $\mathbf{a}_t^k$, we evaluate the score of this frozen reference policy and use

$$g_{\boldsymbol{\phi}}(\mathbf{a}_t^k, \mathbf{s}_t, k) - \nabla_{\mathbf{a}} \log \mu(\mathbf{a} \mid \mathbf{s}_t)\big|_{\mathbf{a}=\mathbf{a}_t^k}$$

as the target for $\varepsilon^{-1} \nabla_{\mathbf{a}} \hat{Q}_{\boldsymbol{\omega}}(\mathbf{s}_t, \mathbf{a}_t^k)$. Actions used for soft-value estimation are also sampled from the same frozen reference policy.

We evaluate these variants on MuJoCo tasks, where the expert demonstrations can be generated by stochastic sampling from pretrained SAC experts. This allows the reference-policy comparison to be conducted with a specified source of expert actions; the generating policy distributions of the robotic-manipulation datasets cannot be verified in the same way. Table 4 reports the resulting episodic returns.

The choice of reference policy affects performance, and the direction of the effect depends on the task. The uniform variant performs better on Swimmer and Walker2d, whereas the Gaussian

Table 4: Sensitivity to the reference policy on MuJoCo tasks. Results are episodic returns, reported as mean ± standard deviation. The expert return is included for context.

Environment	LFIRL-Uniform	LFIRL-Gaussian	Expert
Swimmer-v5	284.56±27.64	208.42±7.25	315.55±1.38
Walker2d-v5	4933.41±52.79	4722.40±230.42	5861.18±73.99
Hopper-v5	3413.50±210.23	3868.04±79.58	4098.17±247.70

Table 5: Sensitivity of PointMaze success rate to Stage-I hyperparameters. Each group varies one parameter while the remaining parameters are held at their default values. Bold indicates the highest result within each parameter group for each environment.

	Diffusion-noise magnitude			Value-anchoring coefficient			Intended margin		
Environment	Low	Medium	High	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$	$\xi = 0.5$	$\xi = 1$	$\xi = 2$
UMaze	**0.98**	0.87	0.68	0.92	**0.98**	0.97	0.57	**0.98**	0.61
Medium	**0.39**	0.35	0.25	0.38	**0.39**	0.35	0.24	**0.39**	0.26
Large	**0.31**	0.29	0.25	0.26	0.31	**0.34**	0.18	**0.31**	0.24

variant performs better on Hopper. LFIRL therefore is not invariant to μ: changing the behavioral baseline changes the reward interpretation and can alter the recovered reward. Nevertheless, the Gaussian-reference variant remains effective, showing that the recovery procedure is not restricted to the uniform-reference simplification.

This experiment assumes that the reference policy is specified. It does not infer μ from expert demonstrations. In general, demonstrations alone do not uniquely separate an unknown reference policy from the reward without additional assumptions; jointly inferring the two is beyond the scope of this work.

D Sensitivity to Stage-I Hyperparameters

LFIRL has several sequential recovery stages, but their hyperparameters need not have the same effect on performance. The value-recovery stage uses a Gumbel-regression-style objective, while the final reward-recovery stage enforces Bellman consistency. Here, we focus on three choices that directly affect the local Q-recovery signal in Stage I: the magnitude of the diffusion noise, the value-anchoring coefficient λ, and the intended margin ξ in value anchoring.

We vary one parameter at a time while holding the other two at their default settings: the low-noise schedule, $\lambda = 1$, and $\xi = 1$. The noise settings scale the coefficient $\sqrt{1 - \bar{\alpha}_k}$ in the forward noising process by $1\times$, $4\times$, and $8\times$, respectively. Table 5 reports PointMaze success rates under these settings.

Diffusion-noise magnitude. Performance decreases as the noise coefficient increases. Increasing $\sqrt{1 - \bar{\alpha}_k}$ moves noisy actions farther from the clean expert-action distribution, where the local approximation between the noisy-action score and the expert-policy score becomes less reliable. The decline is modest from low to medium noise on Medium and Large, but larger at high noise across all three environments. This supports using the low-noise schedule as the default for local Q-gradient supervision.

Value-anchoring coefficient λ. Performance is comparatively stable between $\lambda = 1$ and $\lambda = 2$. Increasing λ to 2 slightly improves Large while causing a small decrease on UMaze and Medium. Reducing it to 0.5 weakens the anchoring signal and lowers performance, particularly on Large. Thus, the method is less sensitive to this coefficient once anchoring is sufficiently strong, with $\lambda = 1$ providing a stable choice across the three environments.

Intended margin ξ. The intended margin has the strongest effect among the three tested parameters. Both $\xi = 0.5$ and $\xi = 2$ underperform $\xi = 1$ on every environment. A small margin provides

insufficient separation between expert actions and their perturbed neighbors. An excessively large margin demands larger local Q differences and may conflict with gradient matching or suppress near-optimal actions. In these experiments, $\xi = 1$ provides the most reliable balance across all three environments.

Overall, these results identify the diffusion-noise magnitude and the anchoring margin as the more consequential Stage-I settings. The value-anchoring coefficient is less sensitive once it is large enough to provide effective anchoring.

E Additional Experimental Details

E.1 Implementation Details and Hyperparameter Setup

The overall training procedure of LFIRL is given in Alg. 1. Key network architecture and hyperparameter settings for each environment are summarized in Tab. 6 and Tab. 7.

Table 6: Network architecture and hyperparameter setup for the PointMaze environments.

	PointMaze_UMaze-v3	PointMaze_Medium-v3	PointMaze_Large-v3
Expert demonstrations	2000	2000	2000
Policy pretraining epochs	60	40	40
Q/b/V/r learning rate	3e-5	3e-5	3e-5
Stage passes (Q, b, V, r)	8/8/20/20	8/8/20/20	8/8/20/20
Q, b, V, r hidden layers	256, 256, 256, 256	256, 256, 256, 256	256, 256, 256, 256
IRL batch size	256	256	256
Discount factor γ	0.99	0.99	0.99
EMA coefficient τ	0.005	0.005	0.005

Table 7: Network architecture and hyperparameter setup for the robotic manipulation environments.

	FrankaKitchen-v1	gym_pusht/PushT-v0	AdroitHandPen-v1
Expert demonstrations	19	200	200
Policy pretraining epochs	20	200	40
Q/b/V/r learning rate	3e-5	3e-4	3e-5
Stage passes (Q, b, V, r)	8/8/20/20	40/40/100/100	40/40/100/100
Q, b, V, r hidden layers	256, 256, 256, 256	256, 256, 256, 256	256, 256, 256, 256
IRL batch size	256	256	256
Discount factor γ	0.99	0.99	0.99
EMA coefficient τ	0.005	0.005	0.005

E.2 Expert Demonstrations Sources

The sources of offline trajectory datasets for experts are provided in Tab. 8. In Maze and Kitchen tasks, we directly use the expert trajectories from the Minari Offline Reinforcement Learning datasets [44]. For the Push-T task, we use the expert trajectories from the dataset in Diffusion Policy [5].

Table 8: The sources of expert policies or demonstrations.

Task	Source
UMaze	https://minari.farama.org/datasets/D4RL/pointmaze/umaze-v2/
Medium	https://minari.farama.org/datasets/D4RL/pointmaze/medium-v2/
Large	https://minari.farama.org/datasets/D4RL/pointmaze/large-v2/
Franka Kitchen	https://minari.farama.org/datasets/D4RL/kitchen/complete-v2/
Push-T	https://diffusion-policy.cs.columbia.edu/data/training/pusht.zip
Pen	https://minari.farama.org/datasets/D4RL/pen/expert-v2/

E.3 Hardware Information

Hardware specifications are provided in Tab. 9.

Table 9: Hardware configuration used in experiments.

Hardware	Specifications
CPU	AMD EPYC 7713 64-Core Processor @ 2 GHz
GPU	NVIDIA A100-SXM4-80GB @ 1215 MHz
Memory	2 TB

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We end the introduction section (Sec. 1) with a summary of main contributions. These contributions are introduced in a more intuitive and condensed manner in the abstract.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations of our proposal approach in the conclusion section (Sec. 7).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We explain the ideas of proofs in the main text of the paper (see Section 4) and provide assumptions and full proofs in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The contribution of this paper is a new algorithm with empirical evaluation. The code is submitted as supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We submit the code in the supplemental material with a readme file that indicates the detailed instructions for running the code.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We briefly introduce the experimental settings in Sec. 5 in the main paper and provide detailed settings (network architecture, hyperparameters, data collection, and pre-processing and training procedures) in Appendices E. The code is submitted as supplementary material.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We conduct multiple independent runs for each experiment (with different seeds) and report the mean value of standard deviations in figures and tables.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide this information in Appendix E.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper presents a novel inverse reinforcement learning algorithm that advances the research in machine learning, which we feel has no negative societal impacts.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original expert dataset and give URLs in Appendix. E.2.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code is submitted as supplemental material with a readme field for detailed instructions to run the code.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The algorithm proposed in this paper does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.