Confidence-Aware Imitation Learning from Demonstrations with Varying Optimality

Anonymous Author(s)
Affiliation
Address
email

Abstract

Most existing imitation learning approaches assume the demonstrations are drawn from experts who are optimal, but relaxing this assumption enables us to tackle a much wider range of data. Standard imitation learning fails when learning from demonstrations with varying optimality, and only learns suboptimal policies. Previous works use confidence scores or rankings to capture beneficial information from demonstrations with varying optimality, but they suffer from many limitations, *e.g.*, manually annotated confidence scores or strong assumptions on the environments. In this paper, we propose a general framework for imitation learning from demonstrations with varying optimality that jointly learns the confidence score and a well-performing policy. Our approach, Confidence-Aware Imitation Learning (CAIL) learns a well-performing policy from confidence-reweighted demonstrations, while uses an outer loss to track the performance of our model and learn the confidence. We provide theoretical guarantees on the convergence of CAIL and evaluate its performance in several simulated environments as well as a real robot experiment. Our results demonstrate that CAIL significantly outperforms other imitation learning methods from demonstrations with varying optimality. We also demonstrate that even without access to any optimal demonstrations, our algorithm can still learn a successful policy, and outperforms prior work.

1 Introduction

We consider a novel, yet rich setting in imitation learning— learning a well-performing policy from a mixture of demonstrations with varying optimality. As opposed to standard imitation learning, where the demonstrations come from experts and thus are optimal, this benefits from a larger and more diverse source of data. However, this introduces a new set of challenges. First, one needs to be able to select useful demonstrations that are not only the optimal demonstrations in the mixture to be able to effectively learn a policy. Specifically, we are interested in settings where we do not have sufficient expert demonstrations in the mixture so we have to rely on learning from sub-optimal demonstrations that can still be successful at parts of the task. Second, we need to be able to filter the negative effects of useless or even malicious demonstrations, *e.g.*, demonstrations that implicitly fail the tasks.

To address the above challenges, we propose to use a measure of *confidence* to indicate the likelihood that a demonstration is optimal. A confidence score can provide a fine-grained characterization of each demonstration's optimality. For example, it can differentiate between near-optimal demonstrations or adversarial ones. By reweighting demonstrations with a confidence score, we can simultaneously address the two challenges of learning from useful but sub-optimal demonstrations while avoiding the negative effects of malicious ones. So our problem reduces to learning an accurate confidence measure for demonstrations. Previous work learns the confidence from manually annotated demonstrations [31], which are difficult to obtain and might contain bias—For example, a conservative

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

and careful demonstrator may assign lower confidence compared to an optimistic demonstrator to the same demonstration. In this paper, we remove restrictive assumptions on the confidence, and propose an approach that automatically learns the confidence score for each demonstration based on evaluation of the outcome of imitation learning. This evaluation often requires access to limited *evaluation data*.

We propose a new algorithm, Confidence-Aware Imitation Learning (CAIL), to jointly learn a well-performing policy and the confidence for every state-action pair in the demonstrations. Specifically, our method adopts a standard imitation learning algorithm and evaluates its performance to update the confidence scores with an evaluation loss, which we refer to as the *outer loss*. In our implementation, we use a limited amount of partial rankings as our evaluation data to train the outer loss. We then update the policy parameters using the loss function of the imitation learning algorithm over the distribution of demonstrations reweighted by the confidence, which we refer to as the *inner loss*. We note that our framework is agnostic to the choice of the imitation learning algorithm as long as there exists an evaluation loss to assess the learned policy.

We optimize for the inner and outer loss using a bi-level optimization [5], and prove that our algorithm converges to the optimal confidence assignments under mild assumptions. We further implement the framework using Adversarial Inverse Reinforcement Learning (AIRL) [14] as the underlying imitation learning algorithm along with its corresponding learning loss as our inner loss. We design a ranking loss as the outer loss, which is compatible with the AIRL model and only requires easy-to-access ranking annotations rather than the exact confidence values.

The main contributions of the paper can be summarized as:

- We propose a novel framework, Confidence-Aware Imitation Learning (CAIL), that jointly learns accurate confidence scores and a well-performing policy from demonstrations with varying optimality.
- We formulate our problem as a bi-level optimization. We prove that the confidence learned by CAIL converges to the optimal confidence with $\mathcal{O}(1/\sqrt{T})$ error under some mild assumptions, where T is the number of update steps.
- We conduct experiments on several simulation and robot environments. Our results suggest that the learned confidence can accurately characterize the optimality of demonstrations, and that the learned policy achieves higher expected return compared to other imitation learning approaches.

2 Related Work

Imitation Learning. The most common approaches for imitation learning are Behavioral Cloning (BC) [21, 4, 24, 23, 3], which treats the problem as a supervised learning problem, and Inverse Reinforcement Learning (IRL), which recovers the reward function from expert demonstrations and finds the optimal policy through reinforcement learning over the learned reward [1, 22, 32]. More recently, approaches such as Generative Adversarial Imitation Learning (GAIL) [18] learn the policy by matching the occupancy measure between demonstrations and the policy in an adversarial manner [15]. Adversarial Inverse Reinforcement Learning (AIRL) [14] and some other approaches [13, 17] improve upon GAIL by simultaneously learning the reward function, and the optimal policy. However, these approaches assume that all the demonstrations are expert demonstrations, and cannot learn a well-performing policy when learning from demonstrations with varying optimality.

Learning from Demonstrations with Varying Optimality: Ranking-based Methods. Ranking-based methods learn a policy from a sequence of demonstrations annotated with rankings [2, 26, 30, 10]. T-REX learns a reward function recovering the ranking of the demonstrations and learns a policy using reinforcement learning [8]. In our work, we assume access to rankings of a small subset of the demonstrations. Relying on such a small number of rankings to learn a reward function using T-REX may lead to learning a reward function with low generalization ability to out of distribution states. D-REX improves T-REX by automatically generating the rankings of demonstrations [9], and SSRR further finds the structure of the reward function [12]. These techniques automatically generate rankings under the assumption that a perturbed demonstration will have a lower reward than the original demonstration, which is not necessarily true for random or malicious demonstrations that can be present in our mixture. DPS utilizes partial orders and pairwise comparisons over trajectories

to learn and generate new policies [20]. However, it requires interactively collecting feedback, which is not feasible in our offline learning setting.

Learning from Demonstrations with Varying Optimality: Confidence-based Methods. Confidence-based methods assume each demonstration or demonstrator holds a confidence value indicating their optimality and then reweight the demonstrations based on this value for imitation learning. To learn the confidence, 2IWIL requires access to ground-truth confidence values for the demonstrations to accurately learn a confidence predictor [31]. Tangkaratt *et al.* require that all the actions for a demonstration are drawn from the same noisy distribution with sufficiently small variance [28]. IC-GAIL implicitly learns the confidence score by aligning the occupancy measure of the learned policy with the expert policy, but requires a set of ground-truth labels to estimate the average confidence [31]. Following works relax the assumption of access to the ground-truth confidence, but still require more optimal demonstrations than non-optimal ones in the dataset [27]. Other works require access to the reward of each demonstration [11]. All of these methods either rely on a specific imitation learning algorithm or require strong assumptions on the confidence. To move forward, we propose a general framework to jointly learn the confidence and the policy. Our framework is flexible as it can use any imitation learning algorithm as long as there exists a compatible outer loss, *i.e.,* the outer loss can evaluate the quality of the imitation learning model.

3 Problem Setting

We formulate the problem of learning from demonstrations with varying optimality as a Markov decision process (MDP): $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rho_0, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability, ρ_0 is the distribution of initial states, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and γ is the discount factor. A policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ defines a probability distribution over the action space in a given state. The expected return, which evaluates the quality of a policy, can be defined as $\eta_\pi = \mathbb{E}_{s_0 \sim \rho_0, \pi} [\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)]$, where t indicates the time step.

We aim to learn a policy that imitates the behavior of a demonstrator d following policy π^d who provides a set of demonstrations $\Xi = \{\xi_1, \ldots, \xi_D\}$ and $\xi_i \sim \pi^d$. Each trajectory is a sequence of state-action pairs $\xi = \{s_0, a_0, \ldots, s_N\}$, and the expected return of a trajectory is $\eta_\xi = \sum_{t=0}^{N-1} \gamma^t \mathcal{R}(s_t, a_t)$. We focus on the offline imitation setting, where a set of demonstrations are provided ahead of time instead of gradually incrementing the dataset as in online imitation learning [19].

A common assumption in classical imitation learning work is that the demonstrations are drawn from the expert policy $\pi^d = \pi^*$, *i.e.,* , the policy that optimizes the expected return of the MDP $\mathcal{M}$ [18, 14]. Here, we relax this assumption so that the demonstrations may contain non-expert demonstrations or even failures—drawn from policies other than π^*. Given the demonstration set $\mathcal{D}$, we need to assess our *confidence* in each demonstration. To achieve learning confidence over this mixture of demonstrations, we rely on the ability to evaluate the performance of imitation learning. This can be achieved by using an evaluation loss trained on *evaluation data*, $\mathcal{D}_E$ (as shown in Fig. 1). In our implementation, we rely on a small amount of rankings between trajectories as our evaluation data: $\mathcal{D}_E = \eta_{\xi_1} \geq \cdots \geq \eta_{\xi_m}$. To summarize, our framework takes a set of demonstrations with varying optimality $\mathcal{D}$ as well as a limited amount of evaluation data $\mathcal{D}_E$ along with an evaluation loss to find a well-performing policy. Note that unlike prior work [31], we do not assume that optimal demonstrations always exist in the demonstration set, and CAIL can still extract useful information from $\mathcal{D}$ while avoiding negative effects of non-optimal demonstrations.

4 Confidence-Aware Imitation Learning

In this section, we first introduce the Confidence-Aware Imitation Learning framework (CAIL). We will then formulate the problem as an optimization in Sec. 4.1, and analyze the convergence of our algorithm in Sec. 4.2. We finally provide an example implementation of CAIL in Sec. 4.3. We provide the pseudocode of CAIL in the appendix.

In our framework, we adopt an imitation learning algorithm with a model F_θ parameterized by θ and a corresponding imitation learning loss $\mathcal{L}_{\text{in}}$, which we refer to as inner loss (as shown in Figure 1). We assign each state-action pair a confidence value indicating the likelihood of the state-action pair appearing in the well-performing policy. The confidence can be defined as a function mapping from a

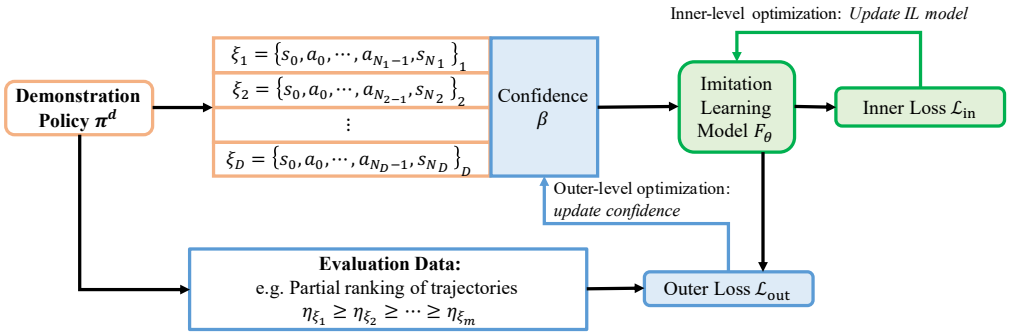


Figure 1: Confidence-Aware Imitation Learning. The demonstrations are shown in the orange box drawn from demonstration policy: $\xi_1, \ldots, \xi_D \sim \pi^d$. The confidence learning component and the outer loss are shown in blue. The confidence β reweights the distribution of state-action pairs in the demonstration set, and then the imitation learning model F_θ learns a well-performing policy and new parameters θ with the confidence-reweighted distribution using the inner loss (imitation loss) shown in green. Next iteration, the updated F_θ generates new trajectories that are then evaluated by the outer loss and potentially other evaluation data (*e.g.* partial ranking of trajectories) to update confidence.

state-action pair to a scalar value $\beta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. We aim to find the optimal confidence assignments β^* to reweight state-action pairs within the demonstrations. We then conduct imitation learning from the reweighted demonstrations using the inner imitation loss $\mathcal{L}_{\text{in}}$ to learn a well-performing policy. Here, we first define the optimal confidence β^* and describe how to learn it automatically.

Defining the Optimal Confidence. We define the distribution of state-action pairs visited by a policy π based on the occupancy measure $\rho_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$: $\rho_\pi(s, a) = \pi(a|s) \sum_{t=0}^{\infty} \gamma^t P(s_t = s|\pi)$, which can be explained as the un-normalized distribution of state transitions that an agent encounters when navigating the environment with the policy π. We can normalize the occupancy measure to form the state-action distribution: $p_\pi(s, a) = \frac{\rho_\pi(s,a)}{\sum_{s,a} \rho_\pi(s,a)}$. Recall that π^d is the policy that the demonstrations are derived from, which can potentially be a mixture of different expert, suboptimal, or even malicious policies. We reweight the state-action distribution of the expert demonstrations to derive a new state-action distribution, which corresponds to another policy π_{new}: $p_{\pi_{\text{new}}}(s, a) = \beta(s, a) p_{\pi^d}(s, a)$. Our goal is to find the optimal confidence β^* that ensures the derived policy π_{new} maximizes the expected return, which can be defined as:

$$\beta^*(s, a) = \arg\max_{\beta} \eta_{\pi_{\text{new}}}. \tag{1}$$

With such $\beta^*(s, a)$, we can conduct imitation learning from the reweighted demonstrations to maximize the expected return with the provided demonstrations.

Learning the Confidence. We will learn an estimate of the confidence score β without access to any annotations of the ground-truth values based on optimizing two loss functions: The inner loss and the outer loss. The inner loss $\mathcal{L}_{\text{in}}$ is accompanied with the imitation learning algorithm encouraging imitation, while the outer loss $\mathcal{L}_{\text{out}}$ captures the quality of imitation learning, and thus optimizing it finds the confidence value that maximizes the performance of the imitation learning algorithm.

Specifically, we first learn the imitation learning model parameters θ^* that minimize the inner loss:

$$\theta^*(\beta) = \arg\min_{\theta} \mathbb{E}_{(s,a)\sim\beta(s,a)p_{\pi^d}(s,a)} \mathcal{L}_{\text{in}}(s, a; \theta, \beta) \tag{2}$$

We note that the inner loss $\mathcal{L}_{\text{in}}(s, a; \theta, \beta)$ refers to settings where (s, a) is sampled from the distribution $\beta(s, a)p_{\pi^d}(s,a)$, and hence implicitly depends on β. Thus we need to find the optimal β^*. This can be derived by minimizing an outer loss:

$$\beta^* = \arg\min_{\beta} \mathcal{L}_{\text{out}}(\theta^*(\beta)), \tag{3}$$

which acts as a form of an evaluation often requiring access to limited evaluation data $\mathcal{D}_E$ (*e.g.* partial rankings if we select a ranking loss as our choice of $\mathcal{L}_{\text{out}}$; we will discuss in more detail in Sec. 4.3).

4.1 Optimization of Outer and Inner Loss

We design a bi-level optimization process consisting of an inner-level optimization and an outer-level optimization to simultaneously update the confidence β and the model parameters θ. Within

the outer-level optimization, we first pseudo-update the imitation learning parameters to build a connection between β and the optimized parameters θ' with the current β. We then update β to make the induced θ' minimize the outer loss $\mathcal{L}_{\text{out}}$ in Eqn. (3). The inner-level optimization is to find the imitation learning model parameters that minimize inner loss $\mathcal{L}_{\text{in}}$ with respect to the confidence β. We introduce the details of the optimization below. We use τ to denote the number of iterations.

Outer-Level Optimization: Updating β. Let β_τ be the current value of confidence at time τ. Using β_τ, we first pseudo-update the imitation learning parameters θ using gradient descent. Let $\theta'_0 = \theta_\tau$ be the current imitation learning model parameters, and we update θ' as:

$$\theta'_{t+1} = \theta'_t - \mu \nabla_{\theta'} \mathcal{L}_{\text{in}}(s, a; \theta'_t, \beta_\tau), \tag{4}$$

where μ is the learning rate, t is the pseudo-updating time step for θ'. We will update θ' with respect to the fixed β_τ after convergence of Eqn. (4). After updating θ', we now update β using gradient descent with the outer loss $\mathcal{L}_{\text{out}}$ from Eqn. (3):

$$\beta_{\tau+1} = \beta_\tau - \alpha \nabla_\beta \mathcal{L}_{\text{out}}(\theta'), \tag{5}$$

where α is the learning rate for updating β. Intuitively, updating β as in Eqn. (5) aims to find the fastest update direction of θ' for decreasing the outer loss $\mathcal{L}_{\text{out}}$.

Inner-Level Optimization: Updating θ. With the updated $\beta_{\tau+1}$, we now will update θ using gradient descent, where we denote the initialization as $\theta_0 = \theta_\tau$.

$$\theta_{t+1} = \theta_t - \mu \nabla_\theta \mathcal{L}_{\text{in}}(s, a; \theta_t, \beta_{\tau+1}). \tag{6}$$

After convergence, we set $\theta_{\tau+1} = \theta$. With the two updates introduced above (outer and inner optimization), we finish one update iteration with setting β_τ to $\beta_{\tau+1}$ using the converged value from Eqn. (5) and θ_τ to $\theta_{\tau+1}$ using the converged value from Eqn. (6).

In each iteration of the above optimization—in the steps of pseudo-updating and the steps of updating the imitation learning model—multiple gradient steps are required for convergence, meaning that there is a nested loop of gradient descent algorithms. The nested loop costs quadratic time and is inefficient especially for deep networks. To further accelerate the optimization, we propose an approximation, which only updates θ once in the pseudo-updating and the updating steps. Therefore, the new updating rule can be formalized as follows:

$$\begin{aligned} \theta'_{\tau+1} &= \theta_\tau - \mu \nabla_\theta \mathcal{L}_{\text{in}}(s, a; \theta_\tau, \beta_\tau), \\ \beta_{\tau+1} &= \beta_\tau - \alpha \nabla_\beta \mathcal{L}_{\text{out}}(\theta'_{\tau+1}), \\ \theta_{\tau+1} &= \theta_\tau - \mu \nabla_\theta \mathcal{L}_{\text{in}}(s, a; \theta_\tau, \beta_{\tau+1}). \end{aligned} \tag{7}$$

4.2 Theoretical Results

We analyze the convergence of the proposed bi-level optimization algorithm for the CAIL framework and derive the following theorems. We provide the detailed proofs of these theorems in the supplementary materials.

Theorem 1. *(Convergence) Suppose the outer loss $\mathcal{L}_{out}$ is Lipschitz-smooth with constant L, the inequality*

$$\nabla_\theta \mathcal{L}_{out}(\theta_{\tau+1})^\top \nabla_\theta \mathcal{L}_{in}(\theta_\tau, \beta_{\tau+1}) \geq C ||\nabla_\theta \mathcal{L}_{in}(\theta_\tau, \beta_{\tau+1})||^2 \tag{8}$$

holds for a constant $C \geq 0$ in every step τ[1], and the learning rate satisfies $\mu \leq \frac{2C}{L}$, then the outer loss decreases along with each iteration: $\mathcal{L}_{out}(\theta_{\tau+1}) \leq \mathcal{L}_{out}(\theta_\tau)$, and the equality holds if $\nabla_\beta \mathcal{L}_{out}(\theta_\tau) = 0$ or $\theta_{\tau+1} = \theta_\tau$.

Remark 1. *The inequality in the assumption of Theorem 1 (Eqn. 8) indicates that the directions of the gradients of $\mathcal{L}_{out}$ and $\mathcal{L}_{in}$ with respect to θ should be close. Intuitively only when the two gradient directions align, we can decrease the evaluation loss $\mathcal{L}_{out}$ by updating θ with $\mathcal{L}_{in}$.*

Theorem 1 ensures that the confidence and the imitation learning parameters monotonically decrease the outer loss. When the gradient of the outer loss with respect to β is zero, β converges to the optimal confidence that minimizes the outer loss, *i.e.,* , β^* in Eqn. (1). With the optimal confidence, we can learn a well-performing policy from more useful demonstrations by reweighting demonstrations when doing imitation learning. Thus, the learned imitation model induces lower outer loss (has higher-quality) than the imitation learning model learned from the original demonstrations in the dataset without reweighting.

[1]We remove (s, a) in $\mathcal{L}_{\text{in}}$ for notation simplicity.

Theorem 2. *(Convergence Rate) Under the assumptions in Theorem 1, let*

$$g(\theta, \beta) = \theta - \mu \nabla_\theta \mathcal{L}_{in}(s, a; \theta, \beta) \tag{9}$$

We assume that $\mathcal{L}_{out}(g(\theta, \beta))$ is Lipschitz-smooth w.r.t. β with constant L_1, $\mathcal{L}_{in}$ and $\mathcal{L}_{out}$ have σ-bounded gradients, and the norm of $\nabla_\beta \nabla_\theta \mathcal{L}_{in}(\theta; \beta)$ is bounded by σ_1. L is the Lipschitz-smooth constant for $\mathcal{L}_{out}$ w.r.t. $g(\theta, \beta)$ as shown in Theorem 1. Consider the total training steps as T, we set $\alpha = \frac{C_1}{\sqrt{T}}$, for some constant C_1 where $0 < C_1 \leq \frac{2}{L_1}$ and $\mu = \frac{C_2}{T}$ for some constant C_2. CAIL can achieve:

$$\min_{1 \leq \tau \leq T} \mathbb{E}[||\nabla_\beta \mathcal{L}_{out}(\theta_\tau)||^2] \leq O\left(\frac{1}{\sqrt{T}}\right). \tag{10}$$

Remark 2. *The assumptions of Theorem 2 are Lipschiz-smoothness and bounded first-order and second-order gradients of $\mathcal{L}_{in}$ and $\mathcal{L}_{out}$, which are satisfied for typical $\mathcal{L}_{in}$ and $\mathcal{L}_{out}$ such as the cross-entropy loss of AIRL and the ranking loss in our implementation of CAIL in Section 4.3.*

With the bound on the convergence rate, the gradient of the outer loss with respect to β is gradually getting close to 0, which means that β gradually converges to the optimal confidence β^* that minimizes the outer loss.

4.3 An Implementation of CAIL

To implement the proposed framework, we need to adopt an imitation learning algorithm with the corresponding imitation loss as the inner loss. We also need to design an outer loss on the imitation learning algorithm to evaluate the quality of imitation given some evaluation data $\mathcal{D}_E$ (*e.g.* partial ranking annotations).

Based on the above considerations, as an instance of the implementation of CAIL, we use Adversarial Inverse Reinforcement Learning (AIRL) [14] as our imitation learning model. We use the imitation loss of AIRL as the inner loss, and a ranking loss (based on a partial ranking of trajectories) as the outer loss. AIRL and the ranking loss are compatible since AIRL can induce the reward function from the discriminator within the model, and the ranking loss can penalize the mismatches of the trajectory rankings computed by the induced reward function and the ground-truth rankings from the evaluation data $\mathcal{D}_E$. Furthermore, the implementation only requires the ranking of a subset of demonstrations $\{\xi_i\}_{i=1}^m \subset \Xi$, *i.e.*, , $\mathcal{D}_E = \eta_{\xi_1} \geq \eta_{\xi_2} \geq \cdots \geq \eta_{\xi_m}$, which is much easier to access than the exact confidence value annotations [6, 20] since confidence not only reflects the rankings of different demonstrations but also how much one demonstration is better than the other.

AIRL consists of a generator G parameterized by θ_G as the policy, and a discriminator parameterized by θ_D. The generator and the discriminator are trained in an adversarial manner as in [15] to match the occupancy measures of the policy and the demonstrations. We write the loss $\mathcal{L}_{\text{in}}$ as:

$$\mathcal{L}_{\text{in}}^D(s, a; \theta^D, \beta) = \mathbb{E}_{(s,a)\sim\beta(s,a)\pi^d}[-\log D(s, a)] + \mathbb{E}_{(s,a)\sim\pi_{\theta^G}}[-\log(1 - D(s, a))], \tag{11}$$

$$\mathcal{L}_{\text{in}}^G(s, a; \theta^G) = \mathbb{E}_{(s,a)\sim\pi_{\theta^G}}[\log D(s, a) - \log(1 - D(s, a))], \tag{12}$$

where $\mathcal{L}_{\text{in}}^D$ is the inner loss for the discriminator, $\mathcal{L}_{\text{in}}^G$ is the inner loss for the generator and π_{θ^G} is the policy derived from the generator. The discriminator D is learned by minimizing the loss $\mathcal{L}_{\text{in}}^D$, which aims to discriminates the state-action pair (s, a) drawn from π_{θ^G} and the state-action pair (s, a) drawn from π^d. The generator parameter θ^G is trained to minimize the loss $\mathcal{L}_{\text{in}}^G$, which enables the generator to generate state-action pairs that are similar to the state transitions in the demonstrations.

For the outer loss, AIRL approximates the reward function by the discriminator parameters, *i.e.,* $\mathcal{R}'_{\theta^D}$. We compute $\eta'_{\xi_i} = \sum_{t=0}^N \gamma^t \mathcal{R}'_{\theta^D}(s_t, a_t)$ as the expected return of a trajectory using the reward $\mathcal{R}'_{\theta^D}$. Then we penalize the mismatches of the rankings derived by η'_{ξ_i} and the ground-truth rankings:

$$\mathcal{L}_{\text{out}}(\theta_D) = \sum_i \sum_{j>i} \text{RK}\left[\eta'_{\xi_i}, \eta'_{\xi_j}; \eta_{\xi_i}, \eta_{\xi_j}\right], \tag{13}$$

where $\mathbb{I}[\eta_{\xi_i} > \eta_{\xi_j}]$ indicates whether the ground-truth expected return of the trajectory ξ_i is larger than ξ_j. RK is defined as a revised version of the widely-used margin ranking loss with margin as 0:

$$\text{RK}\left[\eta'_{\xi_i}; \eta'_{\xi_j}, \eta_{\xi_i}, \eta_{\xi_j}\right] = \begin{cases} \max(0, -\mathbb{I}[\eta_{\xi_i} > \eta_{\xi_j}](\eta'_{\xi_i} - \eta'_{\xi_j})), & |(\eta'_{\xi_i} - \eta'_{\xi_j}| > \epsilon \\ \max(0, \frac{1}{4\epsilon}(\mathbb{I}[\eta_{\xi_i} > \eta_{\xi_j}](\eta'_{\xi_i} - \eta'_{\xi_j}) + \epsilon)^2), & |(\eta'_{\xi_i} - \eta'_{\xi_j})| \leq \epsilon \end{cases} \tag{14}$$

$\mathbb{I}[\eta_{\xi_i} > \eta_{\xi_j}]$ is 1 if $\eta_{\xi_i} > \eta_{\xi_j}$ and otherwise is -1. We revised the original margin ranking loss within a ϵ range around the point of $(\eta'(\xi_i) - \eta'(\xi_j)) = 0$ to make it Lipschitz smooth. If we adopt small enough ϵ, the functionality of the revised marginal ranking loss is close to the original one. In all the experiments, we use $\epsilon = 10^{-5}$.

5 Experiments

In this section, we conduct experiments on the implementation of the framework introduced in Sec. 4.3. We verify the efficacy of the CAIL framework in simulated and real-world environments. We report the results on various compositions of demonstrations with varying optimality. **The code is available in the supplementary materials.**

We conduct experiments in four environments including two MuJoCo environments (Reacher and Ant) [29] in OpenAI Gym [7], one Franka Panda Arm[2] simulation environment, and one real robot environment with a UR5e robot arm[3]. For each environment, we collect a mixture of optimal and non-optimal demonstrations with different optimality to show the efficacy of CAIL. We investigate the performance with respect to the optimality of demonstrations ranging from failures to near-optimal or optimal demonstrations. We provide the implementation details and more results on the sensitivity of the parameters, and visualize the learned confidence in the supplementary materials.

Source of Demonstrations. For MuJoCo environments, following the similar demonstration collecting method as [31], we train a reinforcement learning algorithm and select four intermediate policies as policies with varying optimality and the converged policy as the optimal policy, so that the demonstrations range from worse-than-random ones to near-optimal ones. We draw 20% of demonstrations from each policy. For the RL algorithm, we use SAC [16] for the Reacher environment and PPO [25] for the Ant environment. For the Franka Panda Arm simulation and the real robot environment with UR5e, we hand-craft demonstrations with optimality varying continuously from near-optimal ones to unsuccessful ones to approximate the demonstration collecting process from demonstrators with different levels of expertise. We label only 5% of the demonstrated trajectories with rankings since we target realistic settings where only a small number of rankings are available for the demonstrations.

Baselines. We compare CAIL with the most relevant works in our problem setting including: the state-of-the-art standard imitation learning algorithms: GAIL [18], AIRL [14], imitation learning from suboptimal demonstration methods including two confidence-based methods, 2IWIL and IC-GAIL [31], and three ranking-based methods, T-REX [8], D-REX [9], and SSRR [12]. GAIL and AIRL learn directly from the mixture of optimal and non-optimal demonstrations. T-REX needs demonstrations paired with rankings, so we provide the same number of rankings as our approach. For D-REX and SSRR, we further generate rankings by disturbing demonstrations as done in their papers. For 2IWIL and IC-GAIL—that need a subset of demonstrations labeled with confidence—we label the subset of ranked demonstrations with evenly-spaced confidence, *i.e.,* , the highest expected return as confidence 1, and the lowest expected return as 0. This is a reasonable approximation of the confidence score with no prior knowledge available. For a fair comparison, we re-implement 2IWIL with AIRL as its backbone imitation learning method. For the RL algorithm in T-REX, D-REX, and SSRR, we also use PPO. DPS [20] requires interactively collecting demonstrations and the approach in Cao *et al.* [11] requires the ground truth reward of demonstrations, which are both not implementable under the assumptions in our setting, so we do not include them.

5.1 Results

Reacher and Ant. In the Reacher, the end effector of the arm is supposed to reach a final location. Figure 2(a) shows the optimal trajectories of the joint and the end effector in green, which illustrates the policy reaching the location with the minimum energy cost, and the trajectories with lower optimality in red and orange, where the agent just spins around the center and wastes energy without reaching the target. We collect 200 trajectories in total, where each trajectory has 50 interaction steps.

In Ant, the agent has four legs, each with two links and two joints. Its goal is to move in the x-axis direction as fast as possible. Figure 2(b) illustrates the demonstrated trajectories, where green shows

[2]https://www.franka.de

[3]https://www.universal-robots.com/products/ur5-robot

the optimal one, and red shows suboptimal trajectories (darker colors show lower optimality). In optimal demonstrations, the agent moves quickly along the x-axis, while in suboptimal ones, it moves slowly to other directions. We collect trajectories with $200{,}000$ interaction steps in total.

As shown in Figure 2(e) and 2(f), CAIL achieves the highest expected return compared to the other methods and experiences fast convergence. For Reacher, The p-value between CAIL and the closest baseline method, T-REX, is 5.4054×10^{-6} (statistically significant). For Ant, the p-value between CAIL and the closest baseline method, 2IWIL, is 0.1405. CAIL outperforms standard imitation learning methods, GAIL and AIRL, because CAIL selects more useful demonstrations, and avoids the negative influence of harmful demonstrations. We observe that 2IWIL and IC-GAIL do not perform well because neighboring demonstrations in a given ranking are not guaranteed to have the same distance in terms of confidence score and thus the evenly-spaced confidence values derived from rankings are likely not accurate. All the ranking-based methods do not perform well. For T-REX, the potential reason can be that the rankings of a subset of demonstrations are not enough to learn a generalizable reward function covering states. For D-REX and SSRR, the automatically generated rankings can be incorrect since we also have unsuccessful demonstrations— which can at times be worse than random actions—and perturbing such demonstrations is not guaranteed to produce demonstrations that imply rankings.

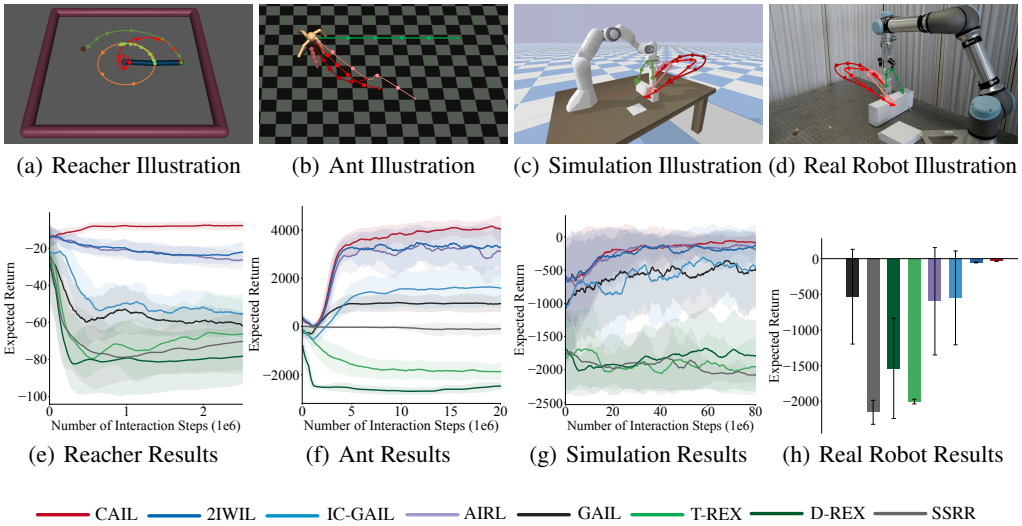
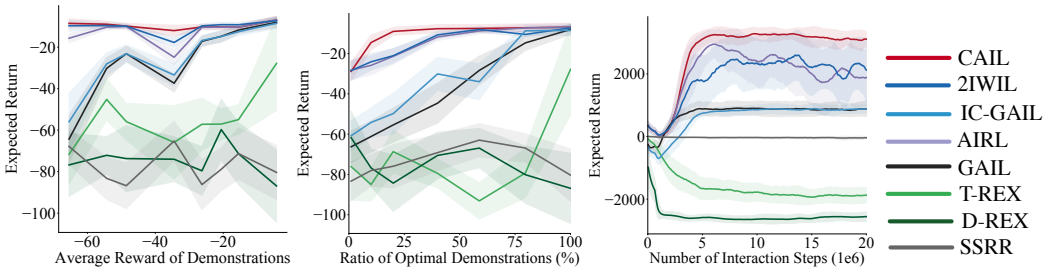


Figure 2: (a) Reacher, (b) Ant, (c) Simulated Panda Robot Arm, (d) Real UR5e Robot Arm. In (a-d), the green trajectories indicate the optimal demonstrations, while the red and orange trajectories indicate demonstrations with varying optimality. (e-g) The expected return with respect to the number of interaction steps. (h) The expected return of the converged policies of CAIL and other methods in UR5e Robot Arm environment.

Robot Arm. We further conduct experiments in more realistic environments. We choose a simulated Franka Panda Arm and a real UR5e robot arm. In both environments, as shown in Figure 2(c) and 2(d), we design a task to let the robot arm pick up a bottle, avoid the obstacle, and put the bottle on a target platform. In the optimal demonstrations in green, the arm takes the shortest path to avoid the obstacle, and puts the bottle on the target, while in suboptimal ones in red (where similar to before the brightness of the trajectories indicates their optimality), the arm detours, does not reach the target, and even at times collides with the obstacle. The suboptimal demonstrations represent a wide range of optimality from near-optimal ones (small detour) to adversarial ones (colliding). We vary the initial position of the robot end-effector and the goal position within an initial area and goal area respectively. For both simulated and real robot environments, we collect trajectories with $200{,}000$ interaction steps in total.

As shown in Figure 2(g) and 2(h), CAIL outperforms other methods in expected return in both the simulated and real robot environments. For the simulated robot arm environment, the p-value between CAIL and the closest baseline, 2IWIL, is 0.0974. For the real robot environment, the p-value between CAIL and the closest baseline, AIRL, is 0.0209 (statistically significant). In particular, in the real robot environment, CAIL achieves a low standard deviation while other methods especially AIRL,



(a) Varying Optimality (b) Varying Ratio (c) Pure Suboptimal

Figure 3: (a-b) The Expected Return with respect to different optimality of demonstrations in the Reacher environment, where the different optimality are created by varying the optimality of non-optimal demonstrations, and varying the ratio of optimal demonstrations. (c) Results for learning from only non-optimal demonstrations in the Ant environment.

IC-GAIL, D-REX and GAIL suffer from an unstable performance. The results demonstrate that CAIL can work stably in the real robot environment. We report the success rate—rate that the robot successfully reaches the target within the time limit without colliding with the obstacle—and videos of sample policy rollouts in the supplementary materials.

Demonstrations with Different Optimality. We show the performance of different methods with demonstrations at different levels of optimality in the Reacher environment. We fix 20% of the total demonstrations to be optimal and make the remaining 80% demonstration drawn from the same suboptimal policy. We vary the optimality of this policy to investigate the performance change with respect to different optimalities. Another way to obtain different optimality is to vary the ratio of optimal demonstrations. We show the results of both varying optimality in Figure 3(a) and 3(b) respectively. We observe that CAIL consistently outperforms or performs comparably to other methods with demonstrations at different optimality. Also, CAIL performs more stably while the baselines suffer from a performance drop at specific optimality levels.

Learning from Only Non-optimal Demonstrations. We verify that CAIL can also learn from solely non-optimal demonstrations without relying on any optimal demonstrations. We remove the optimal demonstrations in the Ant environment and use the remaining demonstrations to conduct imitation learning. As shown in Figure 3(c), CAIL still achieves the best performance among all the methods, which demonstrates that even with all demonstrations being non-optimal, CAIL still can learn useful knowledge from those demonstrations with higher expected return and induce a better policy. The highest p-value between CAIL and the closest baseline (2IWIL) is 0.0067, which indicates the performance gain is statistically significant.

6 Conclusion

Summary. In this paper, we propose a general learning framework, Confidence-Aware Imitation Learning, for imitation learning from demonstrations with varying optimality. We adopt standard imitation learning algorithms with their corresponding imitation loss (inner loss), and leverage an outer loss to evaluate the quality of the imitation learning model. We simultaneously learn a confidence score over the demonstrations using the outer loss and learn the policy through optimizing the inner loss over the confidence-reweighted distribution of demonstrations. Our framework is generally applicable to any imitation learning model with compatible choices of inner and outer losses. We provide theoretical guarantees on the convergence of CAIL and show that the policy learned by CAIL outperforms other imitation learning methods on various simulated and real-world environments under demonstrations with varying optimality.

Limitations and Future Work. Although we propose a flexible framework to address the problem of imitation learning from demonstrations with varying optimality, our work is also limited in a few ways: To learn a well-performing policy from demonstrations with varying optimality, we still require that the dataset consists of demonstrations that encode useful knowledge for policy learning—meaning that our algorithm cannot learn a much better policy from a dataset of fully suboptimal or failure demonstrations. We also require that the demonstrations and the imitation agent have the same dynamics. In the future, we plan to learn from demonstrations with more failures and relax the assumptions of the demonstrations being drawn from the same dynamics as the imitator.

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.

[2] Riad Akrour, Marc Schoenauer, and Michele Sebag. Preference-based policy learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 12–27. Springer, 2011.

[3] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[4] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, 1995.

[5] Jonathan F Bard. *Practical bilevel optimization: algorithms and applications*, volume 30. Springer Science & Business Media, 2013.

[6] Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *arXiv preprint arXiv:2006.14091*, 2020.

[7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[8] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning*, pages 783–792. PMLR, 2019.

[9] Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on Robot Learning*, pages 330–359. PMLR, 2020.

[10] Benjamin Burchfiel, Carlo Tomasi, and Ronald Parr. Distance minimization for reward learning from scored trajectories. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[11] Zhangjie Cao and Dorsa Sadigh. Learning from imperfect demonstrations from agents with varying dynamics. *IEEE Robotics and Automation Letters (RA-L)*, 2021.

[12] Letian Chen, Rohan Paleja, and Matthew Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. In *Conference on Robot Learning*. PMLR, 2020.

[13] 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.

[14] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[16] 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.

[17] Peter Henderson, Wei-Di Chang, Pierre-Luc Bacon, David Meger, Joelle Pineau, and Doina Precup. Optiongan: Learning joint reward-policy options using generative adversarial inverse reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[18] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

[19] Jonathan Lee, Ching-An Cheng, Ken Goldberg, and Byron Boots. Continuous online learning and new insights to online imitation learning. *arXiv preprint arXiv:1912.01261*, 2019.

[20] Ellen R. Novoseller, Yibng Wei, Yanan Sui, Yisong Yue, and J. Burdick. Dueling posterior sampling for preference-based reinforcement learning. In *Uncertainty in Artificial Intelligence (UAI)*, 2020.

[21] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

[22] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591, 2007.

[23] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[24] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

[25] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[26] Hiroaki Sugiyama, Toyomi Meguro, and Yasuhiro Minami. Preference-learning based inverse reinforcement learning for dialog control. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[27] Voot Tangkaratt, Nontawat Charoenphakdee, and Masashi Sugiyama. Robust imitation learning from noisy demonstrations. *arXiv preprint arXiv:2010.10181*, 2020.

[28] Voot Tangkaratt, Bo Han, Mohammad Emtiyaz Khan, and Masashi Sugiyama. Variational imitation learning with diverse-quality demonstrations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9407–9417, Virtual, 13–18 Jul 2020. PMLR.

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

[30] Christian Wirth, Johannes Fürnkranz, and Gerhard Neumann. Model-free preference-based reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[31] Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. Imitation learning from imperfect demonstration. In *International Conference on Machine Learning*, pages 6818–6827, 2019.

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

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We introduce the motivation and basic ideas of the work and summarize the main contribution at the end of the introduction.
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We do not foresee major negative societal impacts of this work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] We provide all assumptions of Theorem 1 and 2.
   (b) Did you include complete proofs of all theoretical results? [Yes] The proofs are in the 'Proofs' section of the supplementary materials.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide the code, data, and instructions in the supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We describe the training details in the 'Implementation Details' subsection of Section 'Experiments' in the supplementary materials.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We provide the error bars in all figures including Figure 2 and 3
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The computation details are included in the 'Implementation Details' subsection of Section 'Experiments' in the supplementary materials.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite the environments we use and the link to the information on robotics hardware used in Sec. 5. In addition, we use the standard Academic Mujoco licence for our experiments in Mojuco environments.
   (b) Did you mention the license of the assets? [Yes] Our assets are open-source under the MIT license provided in the supplementary material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include the implementation of our algorithm and the baselines mentioned in Sec 5.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] The data we are using/curating are all either already public or are synthetically generated. We will open source the generated data along with our code.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] The data contains no personally identifiable information or offensive content
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]