Autonomous Robots
https://doi.org/10.1007/s10514-021-10006-9

Expert Intervention Learning:

An online framework for robot learning from explicit and implicit human feedback

Jonathan Spencer · Sanjiban Choudhury · Matthew Barnes · Matthew Schmittle · Mung Chiang · Peter Ramadge · Sidd Srinivasa

Received: 31 January 2021 / Accepted: 30 June 2021 / Published online: 19 October 2021

Abstract Scalable robot learning from human-robot interaction is critical if robots are to solve a multitude of tasks in the real world. Current approaches to imitation learning suffer from one of two drawbacks. On the one hand, they rely solely on off-policy human demonstration, which in some cases leads to a mismatch in train-test distribution. On the other, they burden the human to label every state the learner visits, rendering it impractical in many applications. We argue that learning interactively from *expert interventions* enjoys the best of both worlds. Our key insight is that any amount of expert feedback, whether by intervention or non-intervention, provides information about the quality of the current state, the quality of the action, or both. We formalize this as a constraint on the learner's value function, which we can efficiently learn using no regret, online learning techniques. We call our approach Expert Intervention Learning (EIL), and evaluate it on a real and simulated driving task with a human expert, where it learns collision avoidance from scratch with just a few hundred samples (about one minute) of expert control.

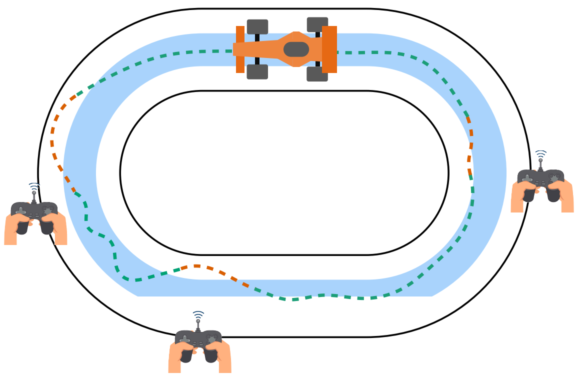


Fig. 1: In intervention learning the expert trains a learner by taking over only when needed, recovering the car and returning control to the learner. A good learner can learn a lot from when the expert does and doesn't intervene.

1 Introduction

Many complex machines we regularly interact with are designed to be operated by humans, and expert humans demonstrate incredible mastery of complex control tasks. However, as self-driving cars [38], robotic factory arms [29], and autonomous helicopters [12] have gradually developed controllers to automate many simple tasks, the human's role has shifted to that of a supervisor that engages or disengages the autopilot, assuming full control only when necessary. This mechanism of supervision and intervention is natural for the expert because it mimics what often occurs in human-human apprenticeship. A goal in robotics is that we similarly endow a robot with all of the human's expertise in a way that is both natural and efficient for the expert human instructor.

Consider the example of training a robot controller for high speed rallycar driving (Fig. 1). While an expert human driver can easily demonstrate this task by driving around a track, we may require long hours of driving to cover all possible input conditions. Even so,

Jonathan Spencer, Mung Chiang, Peter Ramadge
Princeton University
E-mail: {j.spencer,chiangm,ramadge}@princeton.edu

Sanjiban Choudhury, Matthew Barnes, Matthew Schmittle, Sidd Srinivasa
University of Washington
E-mail: {schoudhury,mbarnes,schmttle,sidd}@cs.uw.edu

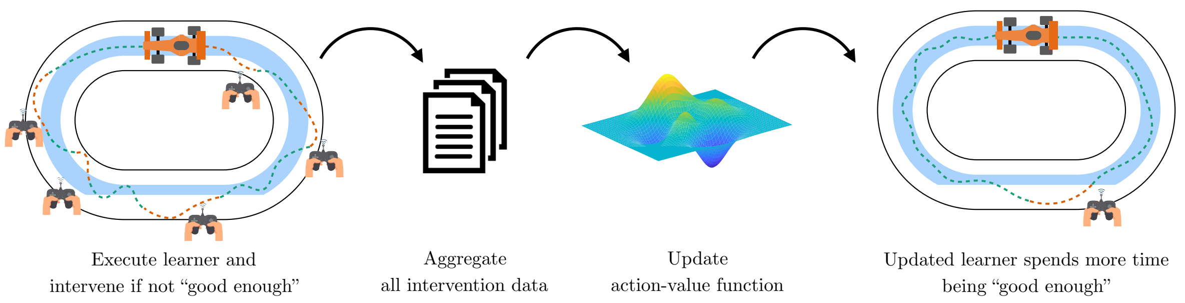


Fig. 2: Overview of EIL. It proceeds iteratively, using the current learner to collect intervention data and map said data to constraints on learner value function. It then aggregates constraints and updates the learner on all of the data

the slightest distribution mismatch between learner and expert can result in compounding errors [35].

Another option is to interactively query the expert on their preferred action [36] while the *learner is in control* of the car. Although interactive learning addresses the distribution mismatch problem, it is impractical due to several human-robot-interaction issues. First, the learner needlessly queries the expert in states that the expert, and ideally a good learner, would never visit [26]. Second, in this approach the expert's suggestions are not used at run-time, so the expert has no actual control of the how the car moves and does not experience the consequences of the feedback from their actions. This results in degraded expert feedback due to delayed response and subsequent over-corrections [37].

On the other hand, if we give the human expert the freedom to intervene and assume control at will, how can the learner correctly interpret such interventions? Consider the scenario in Fig. 1 where the expert nominally monitors the car without any input, similar to the way they might engage with cruise control or an autopilot. As soon as the expert senses that the learner is skidding off the track, they take over, recover the car and toggle control back to the learner. This example illustrates some important truths:

– Expert interventions are natural to provide, and contain useful information.
– In many cases the expert simply wants the learner to perform well enough so the expert doesn't have to intervene.

A good learner should now learn not only how to recover in the future, but also the fact that driving near the middle of the track is much more preferable than being near the edge. Ideally the learner trajectories look something like Fig. 2; as the learner improves, the expert needs to intervene less and less.

Our key insight is that to learn a policy that is optimal everywhere you must query the expert everywhere. If you can settle for good enough, you can use implicit and explicit feedback to quickly learn a level set of the value function rather than learning the value function completely.

We formalize this as Expert Intervention Learning (EIL). When the expert is not in control of the robot, we assume that the state-action pair is good enough. When the expert does occasionally intervene, this provides both implicit feedback about the current state being "bad" and a near-optimal trajectory to return to a "good" state. Fig. 2 shows an overview of the algorithm. At every iteration, we execute the learner, collect intervention data, aggregate it, map to constraints on the learner's action-value function, and update the learner.

Our contributions and the organization of the remainder of the paper are as follows:

1. In Section 3, we formalize the notion of *good enough* performance and frame the mixed control problem in the context of existing work.
2. We introduce an algorithm in Section 4 for solving mixed implicit-explicit feedback problems and show that it has desirable performance guarantees.
3. In Section 5, we empirically demonstrate that our algorithm reduces the number of explicit expert interactions with the system compared to baseline methods both in simulation and using a real robot.

This paper amplifies and expands on [42]. We expand and clarify much of the language and expand our discussion of related work, adding in recent relevant publications. We also include updated proofs and in the appendix provide a counter-example for an alternative method. Finally, in Section 5.3 we perform an additional experiment to analyze the impact that inconsistent, suboptimal experts have on learning.

2 Related Work

The most traditional approach to imitation learning is a technique known as Behavioral Cloning (BC), where the learner gathers a dataset of expert states and actions, and trains a classifier/regressor on that dataset. While straightforward to implement, BC is known to require a large amount (quadratic in trajectory length) of expert demonstration trajectories in order to achieve expert performance [1,13]. This results because learner policies invariably make mistakes and deviate from the expert, inducing a significantly different distribution of states from what the expert originally modeled, an effect known as *covariate shift* [4,33]. Many innovations in imitation learning are attempts to correct for that covariate shift and reduce the required number of expert samples.

DAgger-style Algorithms
DAgger is a foundational algorithm which addresses covariate shift in a provably efficient way by querying the expert online [36]. In DAgger, the learner rolls out their current policy, then queries the expert for action labels corresponding to each state visited by the learner. The learner aggregates this set of learner-state, expert-actions with that of previous iterations, trains a new policy on the combined dataset, and iterates. This approach requires a number of expert labels that is only *linear* in trajectory length. DAgger has been successfully applied to autonomous flight [37] and visual navigation [18,23]. In cases of extreme covariate shift and expert/learner model mismatch, DAgger is known to be optimal [43], though in less extreme cases DAgger introduces redundancy, which can be exploited. DAgger indiscriminately queries the expert for a label at every state the learner visits, which is challenging from a practical usability standpoint. Querying typically happens off-line and can be both cognitively demanding and unsafe [26], inspiring alternative methods to introduce distributional diversity e.g. by injecting noise [27]. Since learner queries indiscriminately, many learner queries are also redundant, leaving room for even greater sample efficiency by intelligently limiting when the expert is queried.

Active Learning for Control
Several DAgger-style algorithms employ active learning, where the robot learner decides after observing each new sample whether or not to query the expert for a label. This decision can be based on a threshold of action-classifier confidence [11,32,17], distributional distance or discrepancy [24,34], query by committee [21], or a combination of state novelty and historical error [28]. Many of these active learning frameworks can query the expert in a batch setting off-line, however most (including DAgger) also query the expert at execution time. This results in a mixed control setting, where the executed trajectory switches back and forth between the human and robot on a per-sample basis and the assignment of control (gating) is at the behest of the robot. Robot-gated mixed control and online active learning in robotics is problematic because the *type* of samples required are burdensome, especially in continuous control settings. Humans are sensitive to latency and timing in mixed control, and demanding sporadic samples in real-time is not only more burdensome than uninterrupted trajectories, but can also result in undesirable and unstable system dynamics [22, 26].

Learning from Interventions
To this end, we instead consider learning from interventions; mixed control where the *human* initiates the hand-off. This approach reduces (though does not eliminate) the alertness burden on the expert, and allows them to determine the exact timing of hand-off in a way that is more convenient and stable for them. However, because the human initiates the handoff, it can be challenging to make use of the entire trajectory since the human criteria for intervening isn't explicitly known. The HG-DAgger algorithm [22] (and similarly [16]), collects intervention data, but uses only the human labeled portions of the trajectory (the orange portions of Fig. 2) as the online batch update for DAgger. In addition to reducing expert burden by requiring less demanding samples, this reduces the overall number of samples required to achieve baseline performance in driving tasks. A similar approach [8,7] applies the DAgger update using the human labelled portions, but uses a hierarchical learner policy based on sub-goals to account for delays in human reaction time. In reinforcement learning settings where a reward signal is available, interventions feedback can safely guide state visitation and manually shape reward [40].

While mixed control is relatively straightforward in driving tasks where the human may simply take the wheel, robotic manipulator and other domains are more complex, often requiring force-feedback. In that context, intervention feedback is less straightforward to interpret since the corrective trajectory cannot be assumed optimal. Successful approaches have interpreted interventions as noisily optimal via inverse reinforcement learning [25], as Bayesian estimates of human parameters [6], as point comparisons for iteratively updating a score function [20], and as sparse indicators of important features [5]. Like the driving-focused intervention learning, these approaches update the learner policy based on the explicit control data provided by the human. A primary way in which we build on current approaches is by considering how to learn from the implicit signals in the times when the expert is *not* in control (the green portions of Fig. 2). This addition, first introduced in [42] is unique among the *Learning from Intervention* literature.

Implicit Feedback
Voluntary expert intervention permits the inference of deeper meaning from the timing and nature of expert feedback. Binary feedback is a form of interactive learning[3] where a supervisor provides real-time approval or disapproval of the robot, and readily lends itself to the inference of implicit signals. These binary signals can be used to learn a supervisor's bias towards positive/negative and exploit that to infer implicit signaling from inaction [30], or that feedback can form an advantage function for the current policy [31]. Additionally, [9] the learner can interpret the feedback adaptively based on previous feedback. We build on these techniques by using demonstrations rather than explicit positive/negative labels, and infer the approval or disapproval based on the timing of interventions. We limit the scope of this work to a coarse model of human intervention that provides regret guarantees, leaving for future work the intriguing questions of potential differences between multiple experts, internal human state [39], trust [10], and cooperation [19,15] in this setting.

This work builds upon the growing field of learning from interventions by considering what we can learn from the times when the human decides *not* to intervene. We clarify and expand on [42], adding proofs and counter-examples and an additional experiment to better make our case. We differ from existing corrective feedback algorithms and active learning DAgger-style algorithms because we aim to learn from the *timing* of the expert correction in addition to copying the actions themselves. Rather than directly learning the policy, we instead learn a score function to coarsely separate bad and good state-actions. The different modes of feedback (explicit/implicit) encode to different constraints on our score function. Since we care only about learning a controller good enough to avoid intervention, we quickly learn a score function with the proper level set dividing good and bad state-actions. This approach to learning from both implicit and explicit feedback in continuous real-time mixed-control setting is novel, efficient, and easy to implement. It is also natural, in the sense that it mimics the way that humans often teach one another.

3 Problem Formulation

We introduce a modified formulation of imitation learning, with the the object that the robot merely perform *good enough* such that the expert doesn't have to intervene. This is inspired by practicalities of domains such as self-driving or manipulation where there are several ways to accomplish the task, and the user doesn't really care to distinguish.

We model the problem as Markov Decision Process with unknown/unspecified rewards (MDP\R). Let $\mathcal{M}(\mathcal{S}, \mathcal{A}, P, T, d_0)$ be a tuple consisting of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, an environment transition function $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ where $\Delta(\mathcal{S})$ is a $|\mathcal{S}|$-dimensional probability simplex, a fixed time horizon T, and an initial state distribution $d_0 \in \Delta(\mathcal{S})$. We restrict the learner to a class of policies $\pi : \mathcal{S} \to \mathcal{A}$, $\pi \in \Pi$. Let d_π^t be the state distribution induced by initializing at d_0 and following policy π for t steps, then $d_\pi = \frac{1}{T}\sum_{t=1}^{T} d_\pi^t$ is the average distribution of states induced by policy π.

We assume the following interaction model between a human expert and robot learner:

1. The expert deems a region of the state-action space $(s, a) \in \mathcal{G}$ to be *good enough.*
2. When the robot is in $\mathcal{G}$, the human does not intervene. The robot remains in control even though it may select actions different from what the expert would have chosen.
3. As soon as the robot departs $\mathcal{G}$, the expert takes over and controls the system back to $\mathcal{G}$.

Although this is a natural human-robot interaction model, it inextricably mixes state distributions induced by both expert and learner. To circumvent this problem, we modify the MDP $\mathcal{M}$. If the expert intervenes at state s_t before the end of the episode $t < T$, we mark s_t as an absorbing terminal state. We wish to minimize the average time spent out of good states,

$$\underset{\pi \in \Pi}{\text{minimize}} \quad \mathbb{E}_{s \sim d_\pi}[1_{\{(s,\pi(s)) \notin \mathcal{G}\}}].$$

However, the objective above disregards the intervention actions demonstrated by the expert. Although the expert's actions are off-policy w.r.t. π, they can help regularize learning and speed up convergence. Let π_E be the expert policy. Let d_π^I be the average distribution of intervention states induced by policy π, i.e. states that the expert visits after π leaves $\mathcal{G}$. We also wish to minimize the average misclassification of intervention actions

$$\underset{\pi \in \Pi}{\text{minimize}} \quad \mathbb{E}_{s^I \sim d_\pi^I}[1_{\{\pi(s^I) \neq \pi^E(s^I)\}}].$$

We combine these objectives to define a modified imitation learning problem.

Problem 1 Find a policy that minimizes both the average time spent outside of good enough region and the

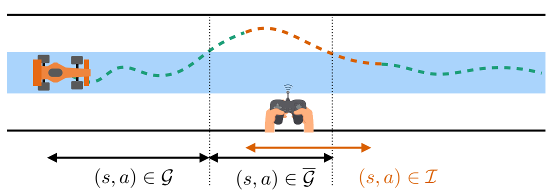


Fig. 3: Given a learner and human intervention trajectory, we can flag the data with three potential categories based on the timing of the correction - good enough state-actions $\mathcal{G}$, bad state-actions $\overline{\mathcal{G}}$ and intervention state-actions $\mathcal{I}$

average misclassification of intervention actions on its own induced distribution:

$$\min_{\pi \in \Pi} \underbrace{\mathbb{E}_{s \sim d_\pi}[1_{\{(s,\pi(s)) \notin \mathcal{G}\}}]}_{\text{stay in good enough region}} + \lambda \underbrace{\mathbb{E}_{s^I \sim d^I_\pi}[1_{\{\pi(s^I) \neq \pi^E(s^I)\}}]}_{\text{learn intervention actions}} \tag{1}$$

where λ is a tuning constant.

We note that (1) is non-convex for two reasons. First, the term inside the expectation is non-convex. Secondly, and more importantly, the induced distributions d_π and d^I_π are non-convex (even for simple convex policy classes). In the next section, we will discuss how we efficiently optimize this objective. Finally, we emphasize that Problem 1 combines the best attributes of two extreme paradigms of imitation learning:

1. *Easy to provide labels as in Behavior Cloning [35]*: In fact, we argue it is even less burdensome to provide a sparse set of interventions.
2. *Correctly measures the learner induced loss as in* DAGGER *[36]*: Moreover, we do so without having to require the expert to provide labels in all the states the learner visits.

4 Approach

We present Expert Intervention Learning (EIL), a no-regret online algorithm that learns from interventions alone. EIL builds upon and further generalizes the key insight of DAGGER [36] – *any* imitation learning objective can be reduced to an online, sequential game. Crucially, unlike DAGGER, EIL *does not* require the expert to label every state the learner enters. We show that EIL indeed enjoys the best of many worlds - it is practical, requires minimal user effort and has strong performance guarantees.

4.1 Modelling interventions as action-value constraints

We restrict the learner to a policy class Π of greedy policies with respect to differentiable action-value cost functions $Q_\theta(s,a)$ such that

$$\pi(s) = \arg\min_a Q_\theta(s,a) \tag{2}$$

Because there is no notion of environment reward R, the function Q does not carry the same meaning as it does in a reinforcement learning (RL) and thus does *not* maintain the property of Bellman consistency (i.e. $Q(s,a) = R(s,a) + E_{s' \sim P(s'|s,a)}[\max Q(s',\cdot)]$). As a result, we permit any differentiable function class, including linear feature weights or neural networks. We choose to use the notation Q because, as in RL, our policy class is a greedy minimizer over the state-action function. Rather than the Bellman consistency requirement, our score function Q will have different constraints which we impose to shape it. Our approach can be thought of as a principled way to continuously add constraints and shape Q from multiple sources of feedback.

In our case we minimize cost, so higher $Q_\theta(s,a)$ indicates an undesirable state-action pair. Hence, we can model a good enough state-action pair $(s,a) \in \mathcal{G}$ as a threshold constraint on the action-value

$$Q_\theta(s,a) \leq B \quad \forall (s,a) \in \mathcal{G} \tag{3}$$

where B is a scalar threshold marking when a state-action falls in the *Bad* region. The precise value of B is irrelevant as it can be thought of as a way of offsetting the action-value estimator (we use $B = 0$ for our experiments).

Let an episode be represented by the learner's trajectory $\xi^L = (s_0, a_0, \cdots, s_f, a_f)$ and the subsequent expert intervention trajectory $\xi^E = (s^E_0, a^E_0, \cdots, s^E_f, a^E_f)$. Fig. 3 illustrates such an episode. Let $[\alpha, \beta) \circ \xi$ represent a *snippet* of trajectory ξ from fraction α up to β where $\alpha, \beta \in [0,1], \alpha \leq \beta$. We map snippets of ξ^L and ξ^E to three non-exclusive categories

1. *Good enough state-actions:* By not intervening during the beginning portion of ξ^L, the expert has implicitly labeled those actions as good enough, thus we label the beginning $1 - \alpha^L$ fraction of ξ^L as such.

$$(s,a) \in \mathcal{G} \quad \forall (s,a) \in [0, 1-\alpha^L) \circ \xi^L \tag{4}$$

2. *Bad state-actions:* Upon intervention, the learner policy has clearly failed, bringing the robot into a bad state. As such, the last α_L fraction of ξ^L we label as bad states. Although the human expert chooses good *actions* we will learn to emulate, the *state* at which they take over may be undesirable.

In such cases, we can choose to label the first α^E fraction of ξ^E as undesirable state-actions.

$$(s,a) \notin \overline{\mathcal{G}} \ \ \forall (s,a) \in [1-\alpha^L, 1] \circ \xi^L \cup [0, \alpha^E) \circ \xi^E \quad (5)$$

3. *Intervention state-actions:* All (s,a) pairs in ξ^E are labeled as intervention pairs.

$$(s,a) \in \mathcal{I} \ \ \forall (s,a) \in \xi^E \quad (6)$$

We discuss how to choose α_L and α_E in Section 5.

We then map each of these categories to constraints on the action-value function

1. *Good enough state-actions* map to values below a threshold

$$Q_\theta(s,a) \leq B \ \ \forall (s,a) \in \mathcal{G}. \quad (7)$$

2. *Bad state-actions* map to values that are above a threshold

$$Q_\theta(s,a) > B \ \ \forall (s,a) \in \overline{\mathcal{G}}. \quad (8)$$

3. *Intervention state-actions* map to a relative action-value constraint requiring we emulate the expert in that state

$$Q_\theta(s,a) < Q_\theta(s,a') \ \ \forall (s,a) \in \mathcal{I}, a' \neq a. \quad (9)$$

Combining these constraints, we can express Problem 1 differently, as an optimization over action-value estimates

$$\begin{aligned} \min_\theta \sum_{(s,a)\in\mathcal{G}} 1_{\{Q_\theta(s,a)>B\}} + \sum_{(s,a)\in\overline{\mathcal{G}}} 1_{\{Q_\theta(s,a)\leq B\}} \\ + \lambda \sum_{(s,a)\in\mathcal{I}} \sum_{a'\neq a} 1_{\{Q_\theta(s,a)\geq Q_\theta(s,a')\}}. \end{aligned} \quad (10)$$

4.2 Reduction to online, convex optimization

The objective in (10) is non-convex in Q_θ. To prove performance guarantees, we apply convex relaxations to each of the terms[1] by using a convex hinge penalty.

1. *Good enough state-actions* corresponding to the upper bound constraint (7) relax to

$$\ell^1_B(s,a,\theta) = \max(0, Q_\theta(s,a) - B) \ \ \forall (s,a) \in \mathcal{G}. \quad (11)$$

2. *Bad state-actions* corresponding to the lower bound constraint (8) relax to

$$\ell^2_B(s,a,\theta) = \max(0, B - Q_\theta(s,a)) \ \ \forall (s,a) \in \overline{\mathcal{G}}. \quad (12)$$

[1] While we assume $Q_\theta(\cdot)$ is convex to prove regret guarantees, the update can be applied to non-convex function classes like neural networks as done in similar works [45]

3. *Intervention state-actions* correspond to a *relative* action-value constraint (9) relax to

$$\ell_C(s,a,\theta) = \sum_{a'} \max(0, Q_\theta(s,a) - Q_\theta(s,a')) \ \forall (s,a) \in \mathcal{I} \quad (13)$$

We combine the first two loss functions as a bounds loss $\ell_B(\cdot) = \ell^1_B(\cdot) + \ell^2_B(\cdot)$. We can think of this as an *implicit loss* inferred from when the expert chooses to intervene. The loss $\ell_C(\cdot)$ is a classification loss. We can think of this an an *explicit loss* which uses the actual actions executed by the expert. The total loss is a weighted sum of losses $\ell(\cdot) = \ell_B(\cdot) + \lambda \ell_C(\cdot)$.

We now formally state the relaxed convex optimization problem using $d_{\pi_\theta}(s,a)$ and $d^I_{\pi_\theta}(s,a)$, the distributions induced by π_θ of nominal learner and expert intervention state, respectively. The objective is to minimize the expected loss over these induced distributions

$$\min_\theta \mathbb{E}_{(s,a)\sim d_{\pi_\theta}(s,a)} \ell_B(s,a,\theta) + \lambda \mathbb{E}_{(s,a)\sim d^I_{\pi_\theta}(s,a)} \ell_C(s,a,\theta) \quad (14)$$

Even though the losses themselves are convex in θ, the optimization is still non-convex because θ impacts the distributions $d_{\pi_\theta}(s,a)$ and $d^I_{\pi_\theta}(s,a)$ over which the expectations are evaluated. We leverage a key insight from DAGGER [36] – reduce the *non-convex* imitation learning objective (14) to a *sequence of convex games.*

The game occurs between an adversary that creates loss functions and a learner that selects parameters. We define the game as follows: At round i, let θ_i be the parameters of the current learner. Let $d_i = d_{\pi_{\theta_i}}$ and $d^I_i = d^I_{\pi_{\theta_i}}$ be the induced distributions. The adversary chooses a convex loss $\ell_i(\theta) = \mathbb{E}_{(s,a)\sim d_i} \ell_B(s,a,\theta) + \lambda \mathbb{E}_{(s,a)\sim d^I_i} \ell_C(s,a,\theta)$. The learner proposes a parameter θ_{i+1}. The average regret is defined as

$$\gamma_N = \frac{1}{N}\sum_{i=1}^N \ell_i(\theta_i) - \min_\theta \frac{1}{N}\sum_{i=1}^N \ell_i(\theta). \quad (15)$$

As long as the learner chooses an update that drives regret $\gamma_N \to 0$ as $N \to \infty$ (no regret), we show in Section 4.5 that the learner finds a near-optimal solution to (14). We choose Follow-the-Leader (FTL) [41]:

$$\begin{aligned} \theta_{i+1} &= \arg\min_\theta \sum_{j=1}^i \ \ell_j(\theta) \\ &= \arg\min_\theta \sum_{j=1}^i \ \mathbb{E}_{(s,a)\sim d_j} \ell_B(s,a,\theta) \\ &\qquad + \lambda \mathbb{E}_{(s,a)\sim d^I_j} \ell_C(s,a,\theta). \end{aligned} \quad (16)$$

Algorithm 1 Expert Intervention Learning (EIL)

```
Initialize data sets $\mathcal{G}$, $\overline{\mathcal{G}}$ and $\mathcal{I}$ as {}
Initialize $\pi_1$ to any policy in $\Pi$
for n = 1, ..., N do
    Execute learner policy $\pi_{\theta_i}$.
    Get learner trajectory $\xi_L$ and subsequent
    intervention trajectory $\xi_E$ (if any).
    Aggregate $(s, a)$ pairs to datasets $\mathcal{G}$, $\overline{\mathcal{G}}$ and $\mathcal{I}$.
    Minimize $\ell_B(s, a, \theta) + \lambda \ell_C(s, a, \theta)$ on total dataset
    to compute new parameter $\theta_{i+1}$.
return best parameter from $\theta_1, \dots, \theta_N$ on validation.
```

FTL updates guarantee $\gamma_N = \tilde{O}(\frac{1}{N})$ for strongly convex ℓ_i, and can be realized by simply aggregating data as it is collected. Alternately, one can apply online gradient descent [46] to avoid storing data.

4.3 Algorithm

Algorithm 1 describes EIL. At each iteration i, the learner is executed to collect trajectories ξ_L and ξ_E. These trajectories are then mapped to the 3 dataset buckets described in Section 4.1. These are then aggregated with previous datasets and the learner is trained to solve (16). The intution is that over iterations, we are building up the set of inputs the learner is likely to experience during its execution. Doing well on this dataset amounts to doing well on (14) – a concept we explore further in Section 4.5.

4.4 Comparison to other imitation learning frameworks

Table 1 places EIL with other comparable imitation learning algorithms. BC [35] never lets the learner be in control, leading to issues such as covariate shift. DAgger [36] lets the learner be in control, but requires the expert to label the learner states, which the authors report to be challenging [37]. HG-DAgger [22,16,8] approaches are closest to EIL, and use interventions, but only optimize $\ell_C(\cdot)$. As we discuss in Section 4.5, this results in the learner only learning recovery behaviors rather than learning to stay in $\mathcal{G}$. EIL gets the best of all worlds - the minimal user burden of HG-DAgger, with DAgger like performance guarantees.

4.5 Analysis

We briefly state the main results deferring proofs and counter examples to the appendix.

Let $i = 1, \cdots, N$ denote the rounds of the online game. Let $\theta_1, \cdots, \theta_N$ be the learner parameters in each round. Let $\ell_i(\theta)$ be the loss function for round i. We build on [36] to show that any no-regret algorithm can achieve near-optimal performance.

Table 1: Different imitation learning algorithms

Algorithm	Intervention Rule	Loss Function
EIL (**ours**)	Intervene if $(s, a) \notin \mathcal{G}$	$\ell_B(\cdot) + \lambda \ell_C(\cdot)$
BC	Expert in control	$\ell_C(\cdot)$
DAgger	Learner in control	$\ell_C(\cdot)^\dagger$
HG-DAgger	Intervene if $(s, a) \notin \mathcal{G}$	$\ell_C(\cdot)$

Theorem 1 *Let $\ell_i(\theta) = \mathbb{E}_{(s,a)\sim d_{\pi_{\theta_i}}} \ell(s, a, \theta)$. Let $\epsilon_N = \min_\theta \frac{1}{N} \sum_{i=1}^{N} \ell_i(\theta)$ be the loss of the best parameter in hindsight after N iterations. Let γ_N be the average regret of $\theta_{1:N}$. There exists a $\theta \in \theta_{1:N}$ s.t.*

$$\mathbb{E}_{(s,a)\sim d_{\pi_\theta}}[\ell(s, a, \theta)] \leq \epsilon_N + \gamma_N \tag{17}$$

Proof For Theorem 1 and Corollaries, see Appendix A.1

Theorem 1 is a simple, but powerful generalization because it extends for any loss function $\ell(s, a, \theta)$ and induced distribution $d_{\pi_\theta}(s, a)$. Because our objective is separable, we can also use this to prove a set of corollaries for variants of the EIL algorithm.

Consider the case where we use only the implicit bounds loss, i.e. only a flag to indicate whether (s, a) is good enough.

Corollary 1 *Let $\ell_i(\theta) = \mathbb{E}_{(s,a)\sim d_{\pi_{\theta_i}}} \ell_B(s, a, \theta)$. Let ϵ_N^B and γ_N^B be the best loss in hindsight and average regret respectively.*
$\exists \theta \in \theta_{1:N}$ s.t. $\mathbb{E}_{(s,a)\sim d_{\pi_\theta}}[\ell_B(s, a, \theta)] \leq \epsilon_N^B + \gamma_N^B$,

i.e. we can use EIL to learn a near-optimal policy with *as little as Boolean feedback*, e.g. from only e-stop disengagements as long as we employ FTL.

Now consider the case where we use only the intervention loss, i.e. the HG-DAgger [22,16,8] algorithm.

Corollary 2 *Let $\ell_i(\theta) = \mathbb{E}_{(s,a)\sim d^I_{\pi_{\theta_i}}} \ell_C(s, a, \theta)$. Let ϵ_N^I and γ_N^I be the best loss in hindsight and average regret respectively.*
$\exists \theta \in \theta_{1:N}$ s.t. $\mathbb{E}_{(s,a)\sim d^I_{\pi_\theta}}[\ell_C(s, a, \theta)] \leq \epsilon_N^I + \gamma_N^I$,

i.e. we can learn to be near-optimal w.r.t mimicking intervention recovery. However, *such a policy may perform arbitrarily poorly when it comes to avoiding intervention in the first place.* Although the learner policy may perfectly mimic the expert recovery, HG-DAgger provides no explicit incentive for the learner to remain inside $(s, a) \in \mathcal{G}$. We bolster this by providing a counter example in Appendix B and with empirical observations in the experiments.

Finally, EIL combines both bound and intervention losses

Corollary 3 *Let $\ell_i(\theta) = \mathbb{E}_{(s,a)\sim d_{\pi_{\theta_i}}} \ell_B(s,a,\theta)$ $+\lambda \mathbb{E}_{(s,a)\sim d^I_{\pi_{\theta_i}}} \ell_C(s,a,\theta)$. Let ϵ_N and γ_N be the best hindsight loss and average regret respectively. $\exists \theta \in \theta_{1:N}$ s.t. $\mathbb{E}_{(s,a)\sim d_{\pi_\theta}}[\ell_B(s,a,\theta)] + \lambda \mathbb{E}_{(s,a)\sim d^I_{\pi_\theta}}[\ell_C(s,a,\theta)] \leq \epsilon_N + \gamma_N$,*

i.e. it performs near-optimally on the combination of the induced bounds and intervention loss.

5 Experiments

We test EIL in a robot driving task where the goal is to track a coarse reference path in a way that is good enough (i.e. collision free, smooth) so the expert need not intervene. The reference path (Fig. 4b,c) may have sharp turns or pass through obstacles - hence we wish to learn a controller than can track it appropriately. We focus on repeated training episodes at specific locations (right turn, straight hallway) to benchmark how quickly the robot learns that specific skill from scratch.

5.1 Experimental Setup

Robot Agent – All robot experiments (sim and real) use the Multi-agent System for non-Holonomic Racing (MuSHR) driving and simulation platform [44], a 1/10 scale car equipped with: short range lidar, RGBD camera, IMU and NVIDIA Jetson Nano (Fig. 4a). We use lidar to manually create a prior map, which we localize against at run-time. Sim and real use an identical model predictive controller (MPC) where:

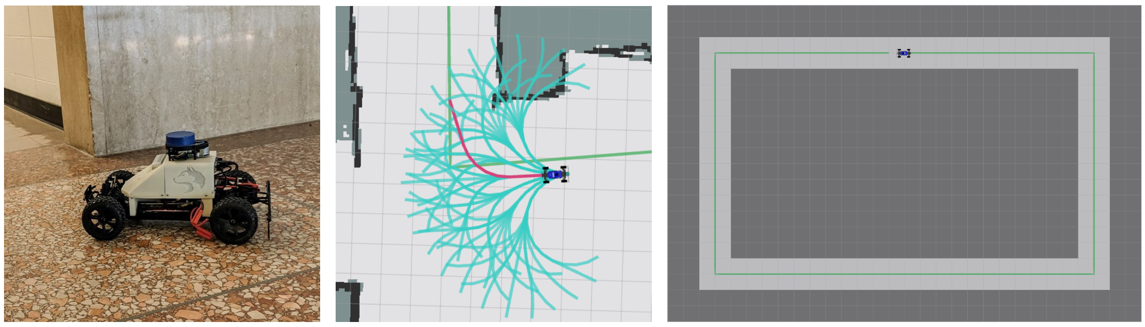
- State $s_t \in \mathcal{S}$ – localized pose and velocity.
- Low-level Control u_t – steering angle ϕ_t and acceleration.
- Action space $\mathcal{A}$ – a fixed library of 64 motion primitives (see Fig. 4b). Each action $a^{(i)}$ is a sequence of pre-defined control and resulting predicted states $a^{(i)}_t = (u^{(i)}_{1:H}, s^{(i)}_{t:t+H})$.
- Feature function $\boldsymbol{f} : s, a \to \mathbb{R}^d$ – for action primitive a we compute the average over states in the primitive horizon $s_{t:t+H}$ for each of: 0-1 boundary violation, absolute curvature to next step, distance to nearest obstacle, and distance to goal path. (We also include unity bias feature.)
- Score function $Q_\theta(s,a) = \theta^T \boldsymbol{f}(s,a)$ – linear in features. (Any differentiable function works for this)
- Policy π_θ – greedy cost minimizer (red line in Fig. 4b) $\pi_\theta(s) = \arg\min_{a\in\mathcal{A}} \ Q_\theta(s,a)$.
- Trajectory $\xi = \{(s_t, a_t, u_t)\}_{t=1}^T$ – a sequence of at most T state-action-control samples of either/both robot or human.

At timestep t, the controller evaluates the policy $a_t = \pi_\theta(s_t)$. Since a_t is a motion primitive, consisting of a sequence of H controls $u_t, \ldots, u_{t+H}$, we execute the first control u_t, observe s_{t+1}, and then repeat the process.

Expert and Hyperparameters – In simulation, the expert uses a set of manually tuned optimal feature weights θ_E, over the same score and policy classes as the learner, $\pi_E(s) = \arg\min_{a\in\mathcal{A}} \ Q_{\theta_E}(s,a)$. We also set an intervention threshold B_E and decide whether to intervene or cede control by continuously scoring the current action w.r.t. Q_E. If the learner action exceeds B_E, the expert supplies subsequent actions until the score drops back below the threshold. In choosing B_E, the expert effectively defines the size of the good enough region, and we discuss how its choice affects performance in Section 5.2. Although a human expert has no such precise internal Q_E, θ_E, or B_E, they provide feedback in a similar way, by intervening and supplying a sequence of nominal or intervention controls $(u_t, \ldots, u_{t+N})$ based on when they perceive the robot as acting poorly or to avoid collision. We project the sequence of expert controls back to the discrete primitive action space $\mathcal{A}$ by minimizing the Fréchet distance[2] [2] between the sequence of states during expert control and the projected states visited for each action in the library $\arg\min_{a\in\mathcal{A}} F((s_{t:t+N}|u_{t:t+N}), (s_{t:t+N}|a))$.

Hyperparameter α_L determines what fraction of the learner trajectory prior to intervention is flagged as good or bad (Eqns 4,5), and α_E represents what fraction of the expert recovery trajectory initially remained in the bad region (Eqn 5). The optimal choice of α_L and α_E depends on properties of both the robot and the human expert, such as MPC horizon and reaction time, though poorly chosen values smoothly degrade performance to be similar to HG-DAgger. For both sim and real, the MPC horizon is $H = 15$ steps ($\sim$ 3m), so we choose α_L so that the last $H = 15$ learner steps prior to the moment of intervention are flagged as bad. In our context, MPC horizon length (rather than expert reaction time) is the dominant factor for choosing α_L since the robot typically lands in a bad state after contemplating it for H steps, and H is usually much longer than the expert reaction time. α_E depends on how quickly the expert recovers and returns to $\mathcal{G}$, but in this context $\alpha_E = 0$ works fine for a skilled expert. Choosing λ very large or very small utilizes only half the available feedback, which slows learning. For our environment, a fixed $\lambda = 1$ worked well, though it is possi-

[2] Fréchet distance is a distance metric commonly used to compare trajectories of potentially uneven length. Informally, given a person walking along one trajectory and a dog following the other without either backtracking, the Fréchet distance is the length of the shortest possible leash for both to make it from start to finish.



(a) MuSHR driving platform (b) MPC discrete action set (blue) and chosen action (red) (c) Generic rectangular track mission

Fig. 4: The Multi-agent System for non-Holonomic Racing (MuSHR) robot is a driving platform that uses model predictive control over a discrete library of possible trajectories (b), but permits takeover from human supervisor. Our mission is to complete collision free laps around a rectangular track (c).

ble to adaptively tune λ as the dataset grows. Likewise, using multiple or sub-optimal experts certainly requires more care, and the Bayesian approach of [30] inspires a way to infer appropriate values of α_L. We set $B = 0$ as it is just a level set threshold for coarsely ranking states. Because Q may develop some bias, the absolute value of B is unimportant, though its relative value holds important lessons, which we discuss in Section 5.2.

Experiment Setup, Baselines, and Evaluation Each policy improvement iteration n as denoted in Algorithm 1 can either be done by gathering a batch of samples under the current policy, or in a fully online manner, updating the policy after each time-step. For the sake of repeatability, we opt for the batch setting, further breaking the mission (Fig. 4c) into two portions trained independently: a straight hallway segment and a sharp corner. The hallway is interesting because it is repetitive, i.e. it should learn with few samples. The corner is interesting because driving through the tip of the corner has only a modest effect on the feature function but is catastrophic for safety. All algorithms are initialized with the same set of (very bad) parameters and given one full example trajectory. For every improvement iteration n, we initialize the car at roughly the same location, generate a short trajectory (50-70 steps, 10-15m), aggregate, improve, and iterate.

We choose as baselines the algorithms listed in Table 1, and evaluate based on 1) the total number of samples in the environment as well as 2) the number of samples supplied by the expert. For Behavioral Cloning (BC) and DAgger, those two numbers will be identical, since the expert labels every state, though DAgger has slightly fewer total samples over the same number of iterations due to learner policy crashing and terminating early. We hold the number of iterations constant, so EIL and HG-DAgger have substantially fewer expert samples since the expert intervenes only when necessary.

In simulation, we judge success according to action suboptimality on a fixed validation set $\mathcal{D}_T$, where suboptimality is $E_{a\sim\pi_L, s\sim\mathcal{D}_T}[Q_E(s, a_L) - Q_E(s, a_E)]$ as a consistent, low-variance benchmark. With a human, the standard of “good enough” is very flexible. Our expert is instructed to intervene *as consistently as possible* based on a) collision avoidance and b) “jerkiness” $\sum_t |\dot{\phi}_t|$. We measure ultimate success by the number of samples required before the policy consistently executes collision-free trajectories and the jerkiness of the converged policy.

5.2 Experimental Results

MuSHR Simulation Results – From our simulation experiments, we make the following observations:

Observation 1: EIL outperforms all algorithms on all datasets both in number of expert samples and total number of environment samples.

In Fig. 5 we see that EIL achieves the highest performance, while HG-DAgger performs comparably to DAgger in terms of total samples, but outperforms DAgger in the number of expert samples since it does not query the expert on the segments of learner control. HG-DAgger amounts to an implementation of EIL with no reliance on implicit feedback, (remove ℓ_B term or make λ very large), which indicates that the improvement in sample efficiency is largely due to the added benefit of implicit feedback. Surprisingly, the hallway proves to be the more challenging task in simulation, because many states are repetitive, and it takes a longer time to discover the key states for shaping optimal performance. On the physical system, however, added lo-

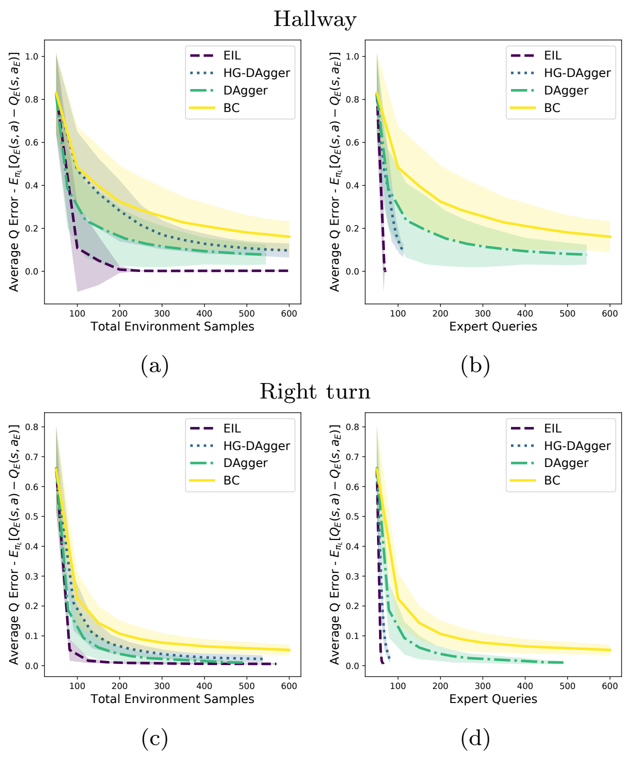


Fig. 5: Simulation performance of EIL compared to baselines (see Table 1) in a straight hallway scenario (a,b) and the right hand turn segment (c,d) pictured in Fig. 4b.

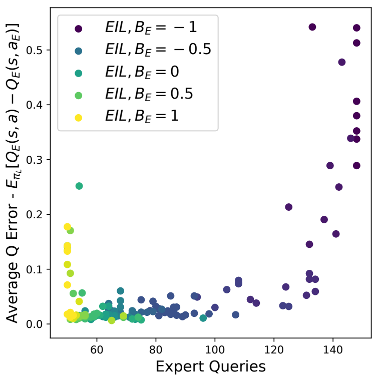


Fig. 6: Performance in driving sim after 200 time-steps for different intervention thresholds B_E. Varying B_E confirms intuitions that an overly aggressive expert is sample inefficient, while an overly passive expert fails to provide useful feedback.

calization error makes precisely clearing the sharp corner the more challenging task.

Observation 2: There exists an optimal level of intervention which minimizes both learner error and expert burden. (Fig. 6)

The hand-crafted π_{θ_E} expert in simulation lets us precisely track learner performance as a function of the expert intervention style, parameterized here by the intervention threshold B_E. In Fig. 6 we explore the range between an aggressively intervening expert ($B_E = -1$) and an overly passive expert ($B_E = 1$). Reducing $B_E \to -\infty$ shrinks the set of good states to nothing and mimics behavioral cloning in requiring that the expert always be in control. The aggressive expert prevents the learner from making mistakes and visiting diverse states that would speed learning, and thus burdens the expert by requiring more samples. On the other hand, by raising the cost threshold $B_E \to \infty$, a passive expert almost never intervenes, giving the agent too much leeway and the agent fails to learn, receiving neither correction nor demonstration. Although a human expert cannot make such precise intervention style adjustments, this gives a useful heuristic: In implementing EIL (and perhaps in life) some intervention is helpful, but too much correction will both exhaust the supervisor and slow the learning process, and too little will rob the learner of meaningful expertise.

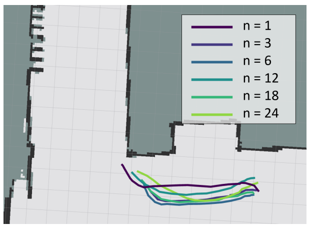
MuSHR Robot with Human Expert – Our experiments with the physical robot and the human expert corroborate our first observation from simulation and add the following observation:

Observation 3: Considering implicit feedback provides clear performance boost over baseline.

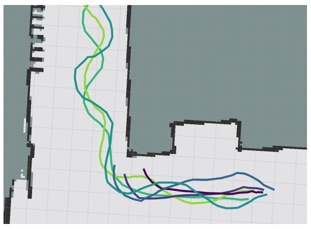
The main difference between EIL and HG-DAGGER is that HG-DAGGER learns only from the recovery trajectories, discarding all other samples, which we notice harms performance. Table 2 shows the number of samples required for the learner to achieve a policy that consistently avoids collision in the right turn scenario. After 24 expert demo trajectories, BC never achieves collision avoidance, demonstrating the difficulty of this task. HG-DAGGER learns collision avoidance with slightly fewer samples than EIL, but the converged policy (after 24 iterations) is still undesirable, making jerky weaving maneuvers seen in Fig. 7b. This is unsurprising, because the HG-DAGGER dataset is imbalanced, largely comprised of jerky explicit "recovery" policy samples supplied by the expert. EIL benefits from the reinforcement of the implicit approval of the smooth parts of its steering and the implicit disapproval of any jerky samples as it swerves towards a wall prior to intervention, and the inclusion of $\alpha_E > 0$ can teach the robot to avoid the initial jerky expert states. In Table 2 and Fig. 7 we see that EIL quickly learns to both avoid collision and smoothly track the reference in a "good enough" way for the expert not to intervene.

Table 2: Samples required to achieve a zero collision policy for right turn, jerkiness ($\sum_t |\dot{\phi}_t|$) and mean obstacle proximity (m) of converged policies after 24 iterations of $T = 70$ steps.

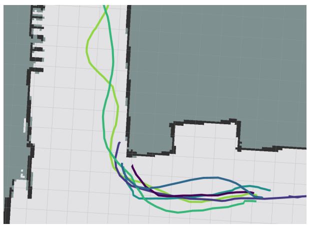
Alg.	**Expert Samps**	**Total Samps**	**Jerkiness**	**Obst. Prox**
BC [35]	> 1680	> 1680	0.0090	0.341
EIL (**ours**)	311	1120	0.0135	0.396
HG-DAGGER	223	560	0.0217	0.437
Human	-	-	0.0114	0.675



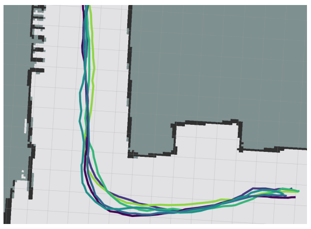
(a) Behavior Cloning



(b) HG-DAGGER



(c) EIL (ours)



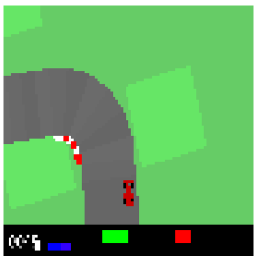
(d) Human demonstrations

Fig. 7: Recorded rollouts of π_{θ_m} show the policy improvement as training progresses for each algorithm.

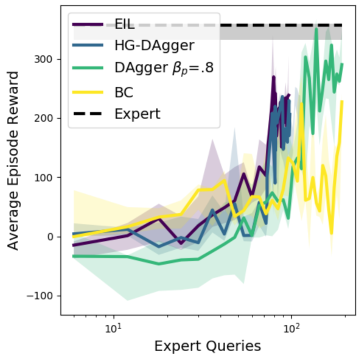
5.3 DQN Expert and Consistency

In our first experiment, the simulated expert Q function is a fine-tuned set of linear feature weights, which means that the expert interventions are consistent w.r.t. the environment features. Similarly, in our second experiment, the human supervisor is trained to be very consistent, intervening at approximately the same estimated badness (time-to-collision, instability), regardless of physical location. In our third experiment we explore the degree to which an *inconsistent* expert hinders performance.

In this experiment we learn to drive from raw pixels using a slightly modified version[3] of the OpenAI Gym CarRacing-v0 environment. We use a Deep Q Network (DQN) trained with the OpenAI Baselines package[14] to produce a Q function which serves as both the expert policy and the scoring function for determining intervention. The learner policy uses the expert's hidden layer output as the feature values to build a linear feature-weighted Q function.



(a)



(b)

Fig. 8: Performance on CarRacing-v0. EIL and HG-DAGGER are more sample efficient than DAGGER , but inconsistent scoring during training by Q_E creates a performance ceiling.

Observation 4: Inconsistency or suboptimality in the expert caps learner performance.

Unlike the MuSHR environment where the handcrafted feature function gave consistent scores, in CarRacing the DQN expert scores very inconsistently, often giving very different scores for states that appear to be visually similar. This ends up misclassifying some bad states as good and vice versa, and is a good indication of the effect that a novice human might have on the system. In this situation, scoring inconsistently ensures that there is no unique minimizer of the objective, and the converged solution is non-deterministic, depending heavily on θ initialization and other optimization properties. Fig. 8 shows the effect this has on performance. Although EIL and HG-DAGGER are initially the most sample efficient, they both reach a performance cap because of the inconsistent expert policy.

6 Discussion

Learning from demonstration in a way that is both sample efficient and easy to implement is challenging since

[3] We modify the action space to have a low constant acceleration and no braking so that the action space was just a discrete set of possible steering angles $[-1, 0, 1]$ to more closely match that of the original DAGGER experiment. We pre-process the 96x96 rgb pixel observation space to LAB color values, using the A,B channels to form a single channel binary thresholded image with all relevant features. We downscale that image to an 8x8 float image, and reshape that into the final state vector $s \in \mathbb{R}^{64}$. The expert network is a DQN of dims 64, (8), 3 with tanh activation at the hidden layer. We use the 8 hidden layer outputs as our feature vector. The learner function class $\boldsymbol{q}(s,a)$ is the set of 27 weights and biases for the output layer.

many techniques ask human experts to perform burdensome off-line labeling [36, 27, 28]. *Expert Intervention Learning* introduces a novel way to train robots that is natural to implement, provides theoretical guarantees, and demonstrates strong performance in practice. EIL exploits both implicit and explicit feedback from corrective demonstrations, the benefit of which we see clearly in the policies produced by the real robot experiment. Relying solely on recovery actions creates an imbalanced dataset, biased towards actions exhibiting undesirable jerky behaviors, whereas the incorporation of coarse implicit feedback gives a more balanced sample set, producing both safe and desirable policies (Fig. 7). Our simulation results also suggest a trade-off for the expert in deciding how good is "good enough" and how strictly to enforce it (Fig. 6). With regard to interventions, if the expert can afford to be patient, then less is more.

Our approach is successful when the expert is consistent and we are satisfied with simply achieving a performance threshold. We posited that EIL is natural and un-burdensome, but supervision still requires alertness, and we showed that when the expert is inconsistent in the timing/location of intervention, the learner performance is stunted, hitting an upper bound. Our coarse goodness threshold was key for harnessing implicit feedback, but an interesting avenue of future work is to incorporate a multi-task learning framework in order to learn from multiple experts. The expert parameters were hardcoded into the hyperparameters of these experiments, and future work will look towards a more general setting to allow for variation between experts.

Acknowledgment

This work was (partially) funded by the DARPA Dispersed Computing program, NIH R01 (R01EB019335), NSF CPS (#1544797), NSF NRI (#1637748), the Office of Naval Research, RCTA, Amazon, and Honda Research Institute USA.

References

1. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the twenty-first International Conference on Machine learning (ICML) (2004)
2. Alt, H., Godau, M.: Computing the Fréchet distance between two polygonal curves. International Journal of Computational Geometry & Applications **5**, 75–91 (1995)
3. Amershi, S., Cakmak, M., Knox, W.B., Kulesza, T.: Power to the people: The role of humans in interactive machine learning. AI Magazine **35**, 105–120 (2014)
4. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and Autonomous Systems (2009)
5. Bajcsy, A., Losey, D.P., O'Malley, M.K., Dragan, A.D.: Learning from physical human corrections, one feature at a time. In: Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction (HRI) (2018)
6. Bajcsy, A., Losey, D.P., O'Malley, M.K., Dragan, A.D.: Learning robot objectives from physical human interaction. In: Proceedings of the 1st Annual Conference on Robot Learning (CoRL). PMLR (2017)
7. Bi, J., Dhiman, V., Xiao, T., Xu, C.: Learning from interventions using hierarchical policies for safe learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34-06, pp. 10352–10360 (2020)
8. Bi, J., Xiao, T., Sun, Q., Xu, C.: Navigation by imitation in a pedestrian-rich environment. arXiv preprint arXiv:1811.00506 (2018)
9. Celemin, C., Ruiz-del Solar, J.: An interactive framework for learning continuous actions policies based on corrective feedback. Journal of Intelligent & Robotic Systems **95**, 77–97 (2019)
10. Chen, M., Nikolaidis, S., Soh, H., Hsu, D., Srinivasa, S.: Planning with trust for human-robot collaboration. In: Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction (HRI) (2018)
11. Chernova, S., Veloso, M.: Interactive policy learning through confidence-based autonomy. Journal of Artificial Intelligence Research (2009)
12. Choudhury, S., Dugar, V., Maeta, S., MacAllister, B., Arora, S., Althoff, D., Scherer, S.: High performance and safe flight of full-scale helicopters from takeoff to landing with an ensemble of planners. Journal of Field Robotics (JFR) (2019)
13. Daumé III, H., Langford, J., Marcu, D.: Search-based structured prediction. Machine Learning Journal (MLJ) (2009)
14. Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., Zhokhov, P.: Openai baselines. https://github.com/openai/baselines (2017)
15. Fisac, J.F., Gates, M.A., Hamrick, J.B., Liu, C., Hadfield-Menell, D., Palaniappan, M., Malik, D., Sastry, S.S., Griffiths, T.L., Dragan, A.D.: Pragmatic-pedagogic value alignment. Robotics Research p. 49–57 (2019)
16. Goecks, V.G., Gremillion, G.M., Lawhern, V.J., Valasek, J., Waytowich, N.R.: Efficiently combining human demonstrations and interventions for safe training of autonomous systems in real-time. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 2462–2470 (2019)
17. Grollman, D.H., Jenkins, O.C.: Dogged learning for robots. In: Proceedings 2007 IEEE International Conference on Robotics and Automation (ICRA) (2007)
18. Gupta, S., Davidson, J., Levine, S., Sukthankar, R., Malik, J.: Cognitive mapping and planning for visual navigation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
19. Hadfield-Menell, D., Russell, S.J., Abbeel, P., Dragan, A.: Cooperative inverse reinforcement learning. In: Advances in Neural Information Processing Systems (NeurIPS) (2016)
20. Jain, A., Wojcik, B., Joachims, T., Saxena, A.: Learning trajectory preferences for manipulators via iterative improvement. In: Advances in Neural Information Processing Systems (NeurIPS) (2013)

21. Judah, K., Fern, A.P., Dietterich, T.G.: Active imitation learning via reduction to iid active learning. In: 2012 AAAI Fall Symposium Series (2012)
22. Kelly, M., Sidrane, C., Driggs-Campbell, K., Kochenderfer, M.J.: Hg-dagger: Interactive imitation learning with human experts. In: 2019 International Conference on Robotics and Automation (ICRA) (2019)
23. Kim, B., Farahmand, A., Pineau, J., Precup, D.: Learning from limited demonstrations. In: Advances in Neural Information Processing Systems (NeurIPS) (2013)
24. Kim, B., Pineau, J.: Maximum mean discrepancy imitation learning. In: Robotics: Science and Systems (RSS) (2013)
25. Kollmitz, M., Koller, T., Boedecker, J., Burgard, W.: Learning human-aware robot navigation from physical interaction via inverse reinforcement learning. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 11025–11031. IEEE (2020)
26. Laskey, M., Chuck, C., Lee, J., Mahler, J., Krishnan, S., Jamieson, K., Dragan, A., Goldberg, K.: Comparing human-centric and robot-centric sampling for robot deep learning from demonstrations. In: IEEE International Conference on Robotics and Automation (ICRA) (2017)
27. Laskey, M., Lee, J., Hsieh, W., Liaw, R., Mahler, J., Fox, R., Goldberg, K.: Iterative noise injection for scalable imitation learning. arXiv preprint arXiv:1703.09327 (2017)
28. Laskey, M., Staszak, S., Hsieh, W.Y.S., Mahler, J., Pokorny, F.T., Dragan, A.D., Goldberg, K.: SHIV: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces. In: 2016 IEEE International Conference on Robotics and Automation (ICRA) (2016)
29. Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., Quillen, D.: Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. The International Journal of Robotics Research (IJRR) (2018)
30. Loftin, R., Peng, B., MacGlashan, J., Littman, M.L., Taylor, M.E., Huang, J., Roberts, D.L.: Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning. Autonomous Agents and Multi-Agent Systems (2016)
31. MacGlashan, J., Ho, M.K., Loftin, R., Peng, B., Wang, G., Roberts, D.L., Taylor, M.E., Littman, M.L.: Interactive learning from policy-dependent human feedback. In: Proceedings of the 34th International Conference on Machine Learning (ICML) (2017)
32. Menda, K., Driggs-Campbell, K.R., Kochenderfer, M.J.: EnsembleDAgger: A Bayesian Approach to Safe Imitation Learning. arXiv preprint arXiv:1807.08364 (2018)
33. Osa, T., Pajarinen, J., Neumann, G., Bagnell, J.A., Abbeel, P., Peters, J.: An algorithmic perspective on imitation learning. Foundations and Trends in Robotics (2018)
34. Packard, B., Ontañón, S.: Policies for active learning from demonstration. In: 2017 AAAI Spring Symposium Series (2017)
35. Pomerleau, D.A.: Alvinn: An autonomous land vehicle in a neural network. In: Advances in Neural Information Processing Systems (NeurIPS) (1989)
36. Ross, S., Gordon, G., Bagnell, D.: 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 (AIStats) (2011)
37. Ross, S., Melik-Barkhudarov, N., Shankar, K.S., Wendel, A., Dey, D., Bagnell, J.A., Hebert, M.: Learning monocular reactive UAV control in cluttered natural environments. In: IEEE International Conference on Robotics and Automation (ICRA) (2013)
38. Sadat, A., Ren, M., Pokrovsky, A., Lin, Y.C., Yumer, E., Urtasun, R.: Jointly learnable behavior and trajectory planning for self-driving vehicles. arXiv preprint arXiv:1910.04586 (2019)
39. Sadigh, D., Sastry, S.S., Seshia, S.A., Dragan, A.: Information gathering actions over human internal state. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2016)
40. Saunders, W., Sastry, G., Stuhlmueller, A., Evans, O.: Trial without error: Towards safe reinforcement learning via human intervention. arXiv preprint arXiv:1707.05173 (2017)
41. Shalev-Shwartz, S.: Online learning and online convex optimization. Foundations and Trends in Machine Learning **4**(2), 107–194 (2012)
42. Spencer, J., Choudhury, S., Barnes, M., Schmittle, M., Chiang, M., Ramadge, P., Srinivasa, S.: Learning from interventions: Human-robot interaction as both explicit and implicit feedback. In: Robotics: Science and Systems (RSS) (2020)
43. Spencer, J., Choudhury, S., Venkatraman, A., Ziebart, B., Bagnell, J.A.: Feedback in imitation learning: The three regimes of covariate shift. arXiv preprint arXiv:2102.02872 (2021)
44. Srinivasa, S.S., Lancaster, P., Michalove, J., Schmittle, M., Summers, C., Rockett, M., Smith, J.R., Choudhury, S., Mavrogiannis, C., Sadeghi, F.: MuSHR: A Low-Cost, Open-Source Robotic Racecar for Education and Research. arXiv preprint arXiv:1908.08031 (2019)
45. Sun, W., Venkatraman, A., Gordon, G.J., Boots, B., Bagnell, J.A.: Deeply AggreVaTeD: Differentiable Imitation Learning for Sequential Prediction. In: Proceedings of the 34th International Conference on Machine Learning (ICML) (2017)
46. Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent. In: Proceedings of the 20th International Conference on Machine Learning (ICML) (2003)

A Appendix - Proofs

A.1 Reduction to no-regret, online learning

The general non i.i.d optimization we wish to solve is

$$\min_{\pi} \mathbb{E}_{(s,a)\sim d^I_{\pi}(s,a)} \ell_C(s,a,\theta) + \lambda \mathbb{E}_{(s,a)\sim d_{\pi}(s,a)} \ell_B(s,a,\theta). \quad (18)$$

We'll directly prove the general setting here rather than proving individually for ℓ_C and ℓ_B.

We reduce this optimization problem to a sequence of convex losses $\ell_i(\theta)$ where the i-th loss is a function of the distribution at that iteration, $\ell_i(\theta) = \mathbb{E}_{(s,a)\sim d^I_i} \ell_C(s,a,\theta) + \lambda \mathbb{E}_{(s,a)\sim d_i} \ell_B(s,a,\theta)$ In our algorithm, the learner at iteration i applies Follow-the-Leader (FTL)

$$\begin{aligned}\theta_{i+1} &= \arg\min_\theta \sum_{t=1}^{i} \quad \ell_t(\theta) \\ &= \arg\min_\theta \sum_{t=1}^{i} \quad \mathbb{E}_{(s,a)\sim d_t^I} \ell_C(s,a,\theta) \\ &\qquad\qquad + \lambda \mathbb{E}_{(s,a)\sim d_t s} \ell_B(s,a,\theta)\end{aligned} \tag{19}$$

Since FTL is a no-regret algorithm, we have the average regret

$$\frac{1}{N}\sum_{i=1}^{N} \ell_i(\theta_i) - \min_\theta \frac{1}{N}\sum_{i=1}^{N} \ell_i(\theta) \leq \gamma_N \tag{20}$$

go to 0 as $N \to \infty$, with $\tilde{O}(\frac{1}{N})$ for strongly convex ℓ_i, (See Theorem 2.4 and Corollary 2.2 in [41])

In this framework, we restate and prove Thm. 1.

Theorem 2 *Let* $\ell_i(\theta) = \mathbb{E}_{(s,a)\sim d_{\pi_{\theta_i}}} \ell(s,a,\theta)$. *Also let* $\epsilon_N = \min_\theta \frac{1}{N}\sum_{i=1}^{N} \ell_i(\theta)$ *be the loss of the best parameter in hindsight after N iterations. Let* γ_N *be the average regret of* $\theta_{1:N}$. *There exists a* $\theta \in \theta_{1:N}$ *s.t.*

$$\mathbb{E}_{(s,a)\sim d_{\pi_\theta}}[\ell(s,a,\theta)] \leq \epsilon_N + \gamma_N \tag{21}$$

Proof The performance of the best learner in the sequence $\theta_1, \cdots, \theta_N$ must be smaller than the average loss of each learner on its own induced distribution (min smaller than average)

$$\min_{\theta\in\theta_{1:N}} \mathbb{E}_{(s,a)\sim d_{\pi_\theta}}[\ell(s,a,\theta)] \leq \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{(s,a)\sim d_{\pi_{\theta_i}}}[\ell(s,a,\theta_i)] \tag{22}$$

Using (20) we have

$$\begin{aligned}&\frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{(s,a)\sim d_{\pi_{\theta_i}}}[\ell(s,a,\theta_i)] \\ &\qquad \leq \gamma_N + \min_\theta \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{(s,a)\sim d_{\pi_{\theta_i}}}[\ell(s,a,\theta)] \\ &\qquad \leq \gamma_N + \epsilon_N\end{aligned} \tag{23}$$

This proof can be extended for finite sample cases following the original DAgger proofs. This theorem applies to each portion of the objective individually, yielding regret terms γ_N^B and γ_N^I which each individually go to zero as $N \to \infty$, thus we are guaranteed that the combined objective as well as each individual objective is zero regret.

B Appendix - HG-DAgger counter example

Several methods on learning from interventions [22,16,8] have proposed a modified form of the DAgger algorithm, first called HG-DAgger [22]. Recall that in HG-DAgger, we only use the intervention loss $\ell_C(.)$, discarding all samples of robot control and only regress to the additional samples of human control at each iteration. Here we construct a counter-example to show when that approach fails.

The MDP is such that the learner can choose between two actions - Left (L) and Right (R) only at states s_0 and s_1. Unknown to the learner, but known to the expert, some of the edges are associated with costs. The expert deems a "good enough" state as having value of -9. Hence whenver the learner enters s_1, the expert takes over to intervene and demonstrates (s_1, L).

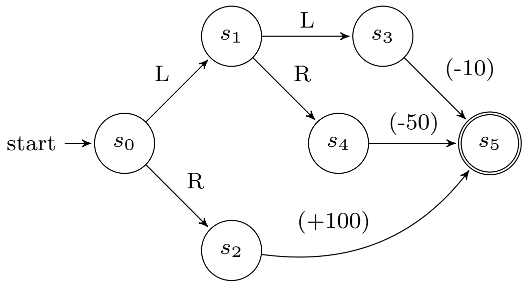


Fig. 9: Counter example for HG-Dagger. Edges without costs are assumed to have $c = 0$, and a single edge leaving a node corresponds to taking any action.

HG-DAgger only keeps this intervention data and uses it as classification loss. Let's say it is using a tabular policy. If it learns the policy (s_0, L) and (s_1, L) - it will indeed achieve $\ell_c(s,a,\theta) = 0$. However, the expert will continue to intervene as this policy always exits the good enough state

Let's look at all policies and their implicit bounds and intervention losses. Assume we get a penalty of 1 for every bad state or misclassified action. We have:

1. Policy $(s_0, L), (s_1, L)$: Loss $\ell_B = 2$, $\ell_C = 0$
2. Policy $(s_0, L), (s_1, R)$: Loss $\ell_B = 2$, $\ell_C = 1$
3. Policy $(s_0, R), (s_1, L)$: Loss $\ell_B = 0$, $\ell_C = 0$
4. Policy $(s_0, R), (s_1, R)$: Loss $\ell_B = 0$, $\ell_C = 0$

The last two policies have the same intervention loss because the induced distribution is such that these policies never result in interventions (even though one learns an incorrect intervention action).

HG-DAgger looks at only the last column and hence my not end up learning (s_0, R). EIL on the other hand will.