

1 A Theoretical Derivations

2 In section 4.1, we have defined the following unbiased imitation objective:

$$D_{KL}(d^\pi(s, a) \| d^H(s, a)) = \mathbb{E}_{(s, a) \sim d^\pi} \left[\log \frac{d^\pi(s, a)}{d^B(s, a)} + \log \frac{d^B(s, a)}{d^H(s, a)} \right] \quad (1a)$$

$$= \mathbb{E}_{(s, a) \sim d^\pi} \left[\log \frac{d^B(s, a)}{d^H(s, a)} \right] + D_{KL}(d^\pi(s, a) \| d^B(s, a)), \quad (1b)$$

3 and we have employ Distribution Correction Estimation (DICE) [20, 24, 25, 26, 32] to reformulate
4 the objective into a tractable form. Here, we provide the detailed theoretical derivations.

5 First, we denote $\tilde{r}(s, a) = -\log \left[\frac{d^B(s, a)}{d^H(s, a)} \right]$, which can be computed with a discriminator:

$$\tilde{r}(s, a) = -\log \left[\frac{d^B(s, a)}{d^H(s, a)} \right] = -\log \left[\frac{1 - c^*(s, a)}{c^*(s, a)} \right],$$

6 where c^* is the optimal discriminator derived by:

$$\max_c \mathbb{E}_{(s, a) \sim d^H} [\log c(s, a)] + \mathbb{E}_{(s, a) \sim d^B} [\log(1 - c(s, a))].$$

7 By imposing the Bellman-flow constraint, we have the following complete unconstrained optimization
8 problem with respect to $d(s, a)$, which serves as a surrogate optimization variable for the occupancy
9 measure $d^\pi(s, a)$:

$$\begin{aligned} & \max_{d \geq 0} \mathbb{E}_{(s, a) \sim d} [\tilde{r}(s, a)] - D_{KL}(d(s, a) \| d^B(s, a)) \\ \text{s.t. } & \sum_{a \in \mathcal{A}} d(s, a) = (1 - \gamma)d_0(s) + \gamma \sum_{s', a'} d(s', a') p(s | s', a'), \forall s \in \mathcal{S}. \end{aligned}$$

10 We use the more general f-divergence form, denoted by D_f , for the KL divergence D_{KL} calculations.
11 By applying Lagrangian duality, we can get the following optimization target:

$$\min_{V(s)} \max_{d \geq 0} \mathbb{E}_{(s, a) \sim d} [\tilde{r}(s, a)] - D_f(d(s, a) \| d^B(s, a)) \quad (2a)$$

$$+ \sum_s V(s) \left((1 - \gamma)d_0(s) + \gamma \sum_{s', a'} d(s', a') p(s | s', a') - \sum_{a \in \mathcal{A}} d(s, a) \right) \quad (2b)$$

$$= \min_{V(s)} \max_{d \geq 0} (1 - \gamma) \mathbb{E}_{d_0(s)} [V(s)] \quad (2c)$$

$$+ \mathbb{E}_{(s, a) \sim d} \left[\tilde{r}(s, a) + \gamma \sum_{s'} p(s' | s, a) V(s') - V(s) \right] - \mathbb{E}_{(s, a) \sim d^B} \left[f\left(\frac{d(s, a)}{d^B(s, a)}\right) \right] \quad (2d)$$

$$= \min_{V(s)} \max_{\omega \geq 0} (1 - \gamma) \mathbb{E}_{d_0(s)} [V(s)] \quad (2e)$$

$$+ \mathbb{E}_{(s, a) \sim d^B} \left[\omega(s, a) \left(r(s, a) + \gamma \sum_{s'} p(s' | s, a) V(s') - V(s) \right) \right] - \mathbb{E}_{(s, a) \sim d^B} [f(\omega(s, a))], \quad (2f)$$

12 where weight $\omega(s, a) = \frac{d(s, a)}{d^B(s, a)}$.

13 Note that the inner maximization over the weighted occupancy measure can be written in the general
14 form of

$$\max_{\omega \geq 0} \omega x - f(\omega), \quad (3)$$

15 where $x := r(s, a) + \gamma \sum_{s'} p(s' | s, a) V(s') - V(s)$ for brevity. This form actually is the definition
16 of the convex conjugate f^* of f , i.e.,

$$f^*(x) = \max_{\omega \geq 0} \omega x - f(\omega). \quad (4)$$

Therefore, applying this conjugate form allows us to eliminate the inner maximization over ω and reduce the bilevel problem to a single-layer optimization objective involving f^* . Thus we have the following tractable optimization form:

$$\min_V (1 - \gamma) \mathbb{E}_{s \sim d_0} V(s) + \mathbb{E}_{(s,a) \sim d^B} [f^* (\mathcal{T}_{\tilde{r}} V(s, a) - V(s))], \quad (5)$$

where $\mathcal{T}_{\tilde{r}} V(s, a) = \tilde{r}(s, a) + \gamma \mathbb{E}_{s' \sim p(s'|s,a)} [V(s')]$ is the Bellman operator.

Following the practical trick in ODICE [25], $1 - \gamma$ can be omitted, the initial state distribution d_0 can be replaced with dataset distribution d^B to stabilize learning and value network can be updated with orthogonal gradient. We can get the following optimization form:

$$\min_V \mathbb{E}_{(s,a) \sim d^B} [(1 - \lambda) V(s) + \lambda f^* (\mathcal{T}_{\tilde{r}} V(s, a) - V(s))], \quad (6)$$

where $\lambda \in (0, 1)$ trades off linearly between the first term and the second term.

B Practical Algorithm

In section 4, we proposed FaithDaIL with active human involvement. However, the optimization problems in Algorithm 1 rely on an offline optimization oracle, which calls for more practical implementation strategies in real-world settings. Therefore, we present a more practical version of FaithDaIL (see Algorithm 2), in which the optimization is solved by multiple steps of stochastic gradient descent (or Adam). This procedure approximates the perturbation-based FTPL update and enables efficient training of neural network policies.

In Algorithm 2, we first initialize the ensemble of K novice policies, K discriminators and K value functions (Lines 1-2). we also initialize their adaptive weights $\alpha_{1,k}$ and interval parameter $\tau_k = 2^{k-1}$ (Lines 3). Same as in Algorithm 1, in each round, we collect behavior data by combining novice trajectories with real-time human interventions, and compute the imitation loss of each policy. The ensemble weights are then updated based on these losses using the Hedge algorithm (Lines 5-14).

For the practical implementation of updating the novice policy, we perform multiple steps of stochastic gradient descent. We first construct training set according to each novice policy’s interval, defined by $\mu_{\tau_k} = \tau_k \lfloor (i - 1) / \tau_k \rfloor + 1$ (Line 17-18). We then initialize the networks and perform multiple mini-batch stochastic gradient descent (SGD) steps. We first sample mini-batches \mathcal{B}_n^B and \mathcal{B}_n^H from each novice policy’s training set (Line 21). We then compute the losses for the current discriminator, value function, and policy, and take a gradient step to update each network accordingly (Line 22-27). After N_m steps, we get the networks for the next online round (Line 29). Finally, after M rounds of training, FaithDaIL returns the novice policy with the highest ensemble weight (Line 32).

C Experiment details

Environment Details. We evaluate our approach on two challenging driving simulators: MetaDrive Safety Benchmark [46] and CARLA Town01 [47].

MetaDrive offers a variety of driving scenarios. For training, we utilize a pool of 100 distinct scenes (IDs 100-200), from which scenarios are randomly sampled for each human-in-the-loop (HIL) session. A separate set of 100 distinct scenes (IDs 1000-1100) is reserved for testing. During these HIL training sessions in MetaDrive, participants monitor the learning agent and provide interventions via a keyboard (Fig. C.1) upon detecting potential violations of traffic regulations, safety constraints, or human-aligned intentions. Each HIL session with a participant comprises approximately 30,000 agent-environment interaction steps (around 70 minutes). In the testing phase, performance is primarily evaluated using episodic return and safety cost. The episodic return is computed following the default configuration detailed in PVP [9]. The safety cost also adheres to default settings, where each collision (e.g., with other vehicles, obstacles, or sidewalks) incurs a +1 penalty to the episodic cost for every timestep the collision persists.

CARLA Town01 provides 25 predefined routes with varying start and end points, as well as diverse conditions such as road geometry, lighting, and weather. In each HIL session, scenarios are randomly sampled for a total of 24,000 agent-environment interaction steps (around 60 minutes). Participants monitor and intervene via a Logitech G923 racing wheel (Fig. C.2). We test on the scenarios randomly sampled from the same set of 25 routes without human involvement.

Algorithm 2 Practical Version of FaithDaIL Algorithm

1: Set $K = \lfloor \log_2 M \rfloor + 1$, $\mathcal{D}_0^H = \emptyset$
 2: Initialize K novice policies $\{\pi_{1,k}^N\}_{k=1}^K$, discriminators $\{c_{1,k}\}_{k=1}^K$ and value functions $\{V_{1,k}\}_{k=1}^K$
 3: Initialize weights $\alpha_{1,k} \leftarrow \frac{1}{K}$, interval parameter $\tau_k = 2^{k-1}$
 4: **for** $i = 1$ to M **do** ▷ Online learning rounds
 5: Sample policy index $k' \sim \text{Categorical}(\{\alpha_{i,k}\})$
 6: Execute $\pi_{i,k'}^N$ to interact with the environment and collect novice policy data \mathcal{D}_i^N
 7: Human expert provides real-time interventions, collect intervention data $\mathcal{D}_i^{H,\text{new}}$
 8: Update $\mathcal{D}_i^H = \mathcal{D}_{i-1}^H \cup \mathcal{D}_i^{H,\text{new}}$
 9: Construct behavior data $\mathcal{D}_i^B = \mathcal{D}_i^N \cup \mathcal{D}_i^H$
 10: **for** $k = 1$ to K **do** ▷ Evaluate loss of each policy on current data
 11: Estimate the proxy reward using the discriminator for \mathcal{D}_i^B :

$$\tilde{r}(s, a) = -\log[(1 - c_{i,k}(s, a))/c_{i,k}(s, a)]$$

 12: Compute importance weights for \mathcal{D}_i^B :

$$\omega_{i,k}^*(s, a) = \max(0, (f')^{-1}(\mathcal{T}_{\tilde{r}} V_{i,k}(s, a) - V_{i,k}(s)))$$

 13: Compute loss $\ell_{i,k}$:

$$\ell_{i,k} = -\mathbb{E}_{(s,a) \sim \mathcal{D}_i^B} [\omega_{i,k}^*(s, a) \log \pi_{i,k}^N(a|s)]$$

 14: **end for**
 15: Update ensemble weights:

$$\alpha_{i+1,k} \leftarrow \frac{\alpha_{i,k} \cdot e^{-\rho \cdot \ell_{i,k}}}{\sum_{k'=1}^K \alpha_{i,k'} \cdot e^{-\rho \cdot \ell_{i,k'}}}$$

 16: **for** $k = 1$ to K **do** ▷ Update novice policies over each interval
 17: Let $\mu_{\tau_k} = \tau_k \lfloor (i-1)/\tau_k \rfloor + 1$
 18: Construct training sets $\mathcal{H}_{i,k}^B = \{\mathcal{D}_j^B\}_{j=\mu_{\tau_k}}^i$, $\mathcal{H}_{i,k}^H = \{\mathcal{D}_j^H\}_{j=\mu_{\tau_k}}^i$
 19: Initialize $c^{(1)} = c_{i,k}$, $V^{(1)} = V_{i,k}$, $\pi^{(1)} = \pi_{i,k}^N$
 20: **for** $n = 1$ to N_m **do** ▷ Mini-batch SGD steps
 21: Sample mini-batch \mathcal{B}_n^B from $\mathcal{H}_{i,k}^B$, \mathcal{B}_n^H from $\mathcal{H}_{i,k}^H$
 22: Compute loss of the discriminator $c^{(n)}$:

$$\mathcal{L}_{\text{disc}}^{(n)} = -\mathbb{E}_{(s,a) \sim \mathcal{B}_n^H} [\log c^{(n)}(s, a)] - \mathbb{E}_{(s,a) \sim \mathcal{B}_n^B} [\log(1 - c^{(n)}(s, a))]$$

 23: Compute reward: $\tilde{r}^{(n)}(s, a) = -\log[(1 - c^{(n)}(s, a))/c^{(n)}(s, a)]$
 24: Compute loss of the value function $V^{(n)}$:

$$\mathcal{L}_{\text{DICE}}^{(n)} = \mathbb{E}_{(s,a) \sim \mathcal{B}_n^B} \left[(1 - \lambda)V^{(n)}(s) + \lambda f^* \left(\mathcal{T}_{\tilde{r}^{(n)}} V^{(n)}(s, a) - V^{(n)}(s) \right) \right]$$

 25: Compute importance weight:

$$\omega^{(n)}(s, a) = \max(0, (f')^{-1}(\mathcal{T}_{\tilde{r}^{(n)}} V^{(n)}(s, a) - V^{(n)}(s)))$$

 26: Compute loss of the policy $\pi^{(n)}$:

$$\mathcal{L}_{\text{WBC}}^{(n)} = -\mathbb{E}_{(s,a) \sim \mathcal{B}_n^B} [\omega^{(n)}(s, a) \log \pi^{(n)}(a|s)]$$

 27: Take one gradient step on $\mathcal{L}_{\text{disc}}^{(n)}$, $\mathcal{L}_{\text{DICE}}^{(n)}$, and $\mathcal{L}_{\text{WBC}}^{(n)}$ to update $c^{(n)}$, $V^{(n)}$, and $\pi^{(n)}$
 28: **end for**
 29: $c_{i+1,k} = c^{(N_m+1)}$, $V_{i+1,k} = V^{(N_m+1)}$ and $\pi_{i+1,k}^N = \pi^{(N_m+1)}$
 30: **end for**
 31: **end for**
 32: **return** Policy π_{M+1,k^*}^N , where $k^* = \arg \max_k \alpha_{M+1,k}$

Table D.2: Network Architecture in CARLA (CNN)

Layer	Filter Numbers	Kernel-size	Stride	Activation Functions
1	16	4×4	3	ReLU
2	32	3×3	2	ReLU
3	64	3×3	2	ReLU
4	128	3×3	2	ReLU
5	256	4×4	4	ReLU

We list the hyper-parameters of ours and other baselines in Tables D.3–D.10. Given the simplicity of its state representation, all networks are reinitialized every 12 rounds to prevent overfitting and enhance training stability. Hyperparameters for the baseline methods follow the configurations reported in PVP [9] and HACO [11].

E Demo Video

A demo video is provided as the supplementary to the case studies (Fig. 2) in our paper. In this video, we present both bird’s-eye view and first-person view for all the methods. The video includes the following six scenarios: (a) Straight Road with Static Obstacles: includes traffic cones and a broken-down car with a warning triangle. (b) Straight Road with Dynamic Vehicles: features moving vehicles on the road. (c) Simple Curve: consists of a long, gentle curve. (d) Simple Roundabout: a basic roundabout layout. (e) Roundabout with Dynamic Vehicles: includes multiple moving vehicles navigating the roundabout. (f) Intersections with Dynamic Vehicles: involves multiple moving vehicles approaching from different directions.

F Ethics Statement

The study protocol was reviewed and approved by the university Institutional Review Board (IRB). All individuals participated voluntarily and provided written informed consent after receiving a comprehensive explanation of the study’s purpose, the nature of their tasks, and how the collected data would be utilized and managed. Data privacy was rigorously maintained: no personally identifiable information was recorded or included in either the dataset or the models trained, and all data were securely stored on university-maintained servers, preserved in line with institutional data retention policies. Prior to the experiments, participants were given detailed instructions and completed a training session with practice until a predefined proficiency level was met, ensuring their comfort and competence with the simulator interfaces and intervention mechanisms.

Table D.3: FaithDaIL (MetaDrive)

Hyper-parameter	Value
Discounted Factor γ	0.99
λ for Value Update	0.4
τ for Target Network Update	0.005
Learning Rate	0.0001
Steps before Learning Start	200
Steps per Iteration	256
Gradient Steps per Iteration	200
Train Batch Size	128
Reinit Round Number	12

Table D.4: FaithDaIL (CARLA)

Hyper-parameter	Value
Discounted Factor γ	0.99
λ for Value Update	0.4
τ for Target Network Update	0.005
Learning Rate	0.0001
Steps before Learning Start	200
Steps per Iteration	256
Gradient Steps per Iteration	40
Train Batch Size	128

Table D.5: PVP (MetaDrive)

Hyper-parameter	Value
Discounted Factor γ	0.99
τ for Target Network Update	0.005
Learning Rate	0.0001
Steps before Learning Start	200
Steps per Iteration	1
Gradient Steps per Iteration	1
Train Batch Size	128
Q Value Bound	1

Table D.6: PVP (CARLA)

Hyper-parameter	Value
Discounted Factor γ	0.99
τ for Target Network Update	0.005
Learning Rate	0.0001
Steps before Learning Start	200
Steps per Iteration	1
Gradient Steps per Iteration	1
Train Batch Size	128
Q Value Bound	1

Table D.7: HACO (MetaDrive)

Hyper-parameter	Value
Discounted Factor γ	0.99
τ for Target Network Update	0.005
Learning Rate Actor	0.0003
Learning Rate Critic	0.0003
Learning Rate Entropy	0.0003
Steps before Learning Start	200
Steps per Iteration	1
Gradient Steps per Iteration	1
Target Network Update Interval	1
Train Batch Size	128
CQL Loss Temperature	1.0

Table D.8: HACO (CARLA)

Hyper-parameter	Value
Discounted Factor γ	0.99
τ for Target Network Update	0.005
Learning Rate Actor	0.0003
Learning Rate Critic	0.0003
Learning Rate Entropy	0.0003
Steps before Learning Start	200
Steps per Iteration	1
Gradient Steps per Iteration	1
Target Network Update Interval	1
Train Batch Size	128
CQL Loss Temperature	1.0

Table D.9: TD3 (MetaDrive)

Hyper-parameter	Value
Discounted Factor γ	0.99
τ for Target Network Update	0.005
Learning Rate	0.0001
Steps before Learning Start	10000
Steps per Iteration	1
Gradient Steps per Iteration	1
Train Batch Size	100

Table D.10: TD3 (CARLA)

Hyper-parameter	Value
Discounted Factor γ	0.99
τ for Target Network Update	0.005
Learning Rate	0.0001
Steps before Learning Start	10000
Steps per Iteration	1
Gradient Steps per Iteration	1
Train Batch Size	100