Appendix of CEIL

Anonymous Author(s)
Affiliation
Address
email

1 Additional Derivation

(*Repeat from the main paper.*) To gain more insight into Equation 4 that captures the quality of IL (the degree of similarity to the expert data), we define $D(\cdot,\cdot)$ as the sum of reverse KL and forward KL divergence, *i.e.*, $D(q,p) = D_{\text{KL}}(q\|p) + D_{\text{KL}}(p\|q)$, and derive an alternative form for Equation 4:

$$\arg\min_{\mathbf{z}^*} \; D(\pi_\theta(\boldsymbol{\tau}|\mathbf{z}^*), \pi_E(\boldsymbol{\tau})) = \arg\max_{\mathbf{z}^*} \; \underbrace{\mathcal{I}(\mathbf{z}^*;\boldsymbol{\tau}_E) - \mathcal{I}(\mathbf{z}^*;\boldsymbol{\tau}_\theta)}_{\mathcal{J}_{\text{MI}}} - \underbrace{D_{\text{KL}}(\pi_\theta(\boldsymbol{\tau}), \pi_E(\boldsymbol{\tau}))}_{\mathcal{J}_D},$$

where $\mathcal{I}(\mathbf{x};\mathbf{y})$ denotes the mutual information (MI) between $\mathbf{x}$ and $\mathbf{y}$, which measures the predictive power of $\mathbf{y}$ on $\mathbf{x}$ (or vice-versa), the latent variables are defined as $\boldsymbol{\tau}_E := \boldsymbol{\tau} \sim \pi_E(\boldsymbol{\tau})$, $\boldsymbol{\tau}_\theta := \boldsymbol{\tau} \sim p(\mathbf{z}^*)\pi_\theta(\boldsymbol{\tau}|\mathbf{z}^*)$, and $\pi_\theta(\boldsymbol{\tau}) = \mathbb{E}_{\mathbf{z}^*}\left[\pi_\theta(\boldsymbol{\tau}|\mathbf{z}^*)\right]$.

Below is our derivation:

$$\begin{aligned}
&\min_{\mathbf{z}^*} \; D(\pi_\theta(\boldsymbol{\tau}|\mathbf{z}^*), \pi_E(\boldsymbol{\tau})) \\
&= \min_{\mathbf{z}^*} \; \mathbb{E}_{p(\mathbf{z}^*)}\left[D_{\text{KL}}(\pi_\theta(\boldsymbol{\tau}|\mathbf{z}^*)\|\pi_E(\boldsymbol{\tau})) + D_{\text{KL}}(\pi_E(\boldsymbol{\tau})\|\pi_\theta(\boldsymbol{\tau}|\mathbf{z}^*))\right] \\
&= \min_{\mathbf{z}^*} \; \mathbb{E}_{p(\mathbf{z}^*)\pi_\theta(\boldsymbol{\tau}|\mathbf{z}^*)}\left[\log \pi_\theta(\boldsymbol{\tau}|\mathbf{z}^*) - \log \pi_E(\boldsymbol{\tau})\right] \\
&\qquad + \mathbb{E}_{p(\mathbf{z}^*)\pi_E(\boldsymbol{\tau})}\left[\log \pi_E(\boldsymbol{\tau}) - \log \pi_\theta(\boldsymbol{\tau}|\mathbf{z}^*)\right] \\
&= \min_{\mathbf{z}^*} \; \mathbb{E}_{p(\mathbf{z}^*)\pi_\theta(\boldsymbol{\tau}|\mathbf{z}^*)}\left[\log \frac{p(\mathbf{z}^*|\boldsymbol{\tau})\pi_\theta(\boldsymbol{\tau})}{p(\mathbf{z}^*)} - \log \pi_E(\boldsymbol{\tau})\right] \\
&\qquad + \mathbb{E}_{p(\mathbf{z}^*)\pi_E(\boldsymbol{\tau})}\left[\log \pi_E(\boldsymbol{\tau}) - \log \frac{p(\mathbf{z}^*|\boldsymbol{\tau})\pi_\theta(\boldsymbol{\tau})}{p(\mathbf{z}^*)}\right] \\
&= \min_{\mathbf{z}^*} \; \mathbb{E}_{p(\mathbf{z}^*)\pi_\theta(\boldsymbol{\tau}|\mathbf{z}^*)}\left[\log \frac{p(\mathbf{z}^*|\boldsymbol{\tau})}{p(\mathbf{z}^*)} + \log \frac{\pi_\theta(\boldsymbol{\tau})}{\pi_E(\boldsymbol{\tau})}\right] - \mathbb{E}_{p(\mathbf{z}^*)\pi_E(\boldsymbol{\tau})}\left[\log \frac{p(\mathbf{z}^*|\boldsymbol{\tau})}{p(\mathbf{z}^*)} + \log \frac{\pi_\theta(\boldsymbol{\tau})}{\pi_E(\boldsymbol{\tau})}\right] \\
&= \max_{\mathbf{z}^*} \; \mathcal{I}(\mathbf{z}^*;\boldsymbol{\tau}_E) - \mathcal{I}(\mathbf{z}^*;\boldsymbol{\tau}_\theta) - D_{\text{KL}}(\pi_\theta(\boldsymbol{\tau}), \pi_E(\boldsymbol{\tau})) + C,
\end{aligned}$$

where $\boldsymbol{\tau}_E := \boldsymbol{\tau} \sim \pi_E(\boldsymbol{\tau})$, $\boldsymbol{\tau}_\theta := \boldsymbol{\tau} \sim p(\mathbf{z}^*)\pi_\theta(\boldsymbol{\tau}|\mathbf{z}^*)$, and C is a constant.

2 More Comparisons and Ablation Studies

2.1 Offline Comparison on D4RL Expert Domain Dataset

In Table 1, we provide the normalized return of our method and baseline methods on the reward-free D4RL [4] expert dataset. Consistently, we can observe that CEIL achieves a significant improvement over the baseline methods in both *S-off-LfD* and *S-off-LfO* settings. Compared to the state-of-the-art offline IL baselines, CEIL also shows competitive results on the challenging cross-domain offline IL settings (*C-off-LfD* and *C-off-LfO*).

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

Table 1: Normalized scores (averaged over 30 trails for each task) on D4RL expert dataset. Scores within two points of the maximum score are highlighted. hop: Hopper-v2. hal: HalfCheetah-v2. wal: Walker2d-v2. ant: Ant-v2.

		hop expert	hal expert	wal expert	ant expert	sum
S-off-LfD	ORIL (TD3+BC)	97.5	91.8	14.5	76.8	280.6
	SQIL (TD3+BC)	25.5	14.4	8.0	44.3	92.1
	IQ-Learn	37.3	9.9	46.6	85.9	179.7
	ValueDICE	65.6	2.9	28.2	90.5	187.1
	DemoDICE	107.3	87.1	104.8	114.2	413.3
	SMODICE	111.0	93.5	108.2	122.0	**434.7**
	CEIL	106.0	96.0	115.6	117.8	**435.4**
S-off-LfO	ORIL (TD3+BC)	64.2	92.1	12.2	44.3	212.8
	SMODICE	111.3	93.7	108.0	122.0	**435.0**
	CEIL	103.3	96.8	110.0	126.4	**436.5**
C-off-LfD	ORIL (TD3+BC)	24.4	78.3	29.3	32.1	164.1
	SQIL (TD3+BC)	12.2	19.9	8.8	21.2	62.0
	IQ-Learn	25.9	31.2	31.7	55.8	144.6
	ValueDICE	18.6	9.8	8.3	22.3	59.0
	DemoDICE	111.5	88.7	107.9	122.5	430.6
	SMODICE	111.1	93.8	108.2	120.9	**434.0**
	CEIL	105.8	97.1	108.6	112.2	423.7
C-off-LfO	ORIL (TD3+BC)	22.5	76.6	11.2	28.2	138.6
	SMODICE	111.2	93.7	108.1	117.7	430.7
	CEIL	113.0	90.1	108.7	125.2	**437.0**

Table 2: Normalized scores (*evaluated on the expert dataset* over 30 trails for each task) on 2 cross-domain offline IL settings: *C-off-LfD* and *C-off-LfO*. Scores within two points of the maximum score are highlighted. m: medium. mr: medium-replay. me: medium-expert. e: expert.

		Hopper-v2				HalfCheetah-v2				sum
		m	mr	me	e	m	mr	me	e	
C-off-LfD	ORIL (TD3+BC)	74.7	16.7	45.0	21.4	2.2	0.8	-0.3	-2.2	158.3
	SQIL (TD3+BC)	33.6	21.6	14.5	14.5	18.2	7.5	20.9	20.9	151.8
	IQ-Learn	11.8	9.7	17.1	17.1	7.7	7.8	9.5	9.5	90.2
	ValueDICE	49.5	24.2	55.7	49.3	32.2	32.9	38.7	28.7	311.2
	DemoDICE	83.2	31.5	81.6	28.5	0.9	-1.1	-1.7	-2.4	220.6
	SMODICE	80.1	26.1	78.0	54.3	2.8	-1.0	1.0	-2.3	239.1
	CEIL	87.4	74.3	81.2	82.4	44.0	30.4	25.0	17.1	**441.9**
*C-off-Lf*O	ORIL (TD3+BC)	62.3	18.7	57.0	28.2	0.2	1.1	-0.3	-2.3	165.0
	SMODICE	77.6	22.5	80.2	71.0	2.0	-0.9	0.8	-2.3	250.9
	CEIL	56.4	58.6	56.7	65.2	5.5	36.5	5.0	5.0	**288.7**
		Walker2d-v2				Ant-v2				sum
		m	mr	me	e	m	mr	me	e	
C-off-LfD	ORIL (TD3+BC)	22.0	24.5	23.9	33.1	16.0	18.6	2.5	0.4	141.0
	SQIL (TD3+BC)	32.4	14.9	10.3	10.3	71.4	63.6	60.1	60.1	323.1
	IQ-Learn	8.4	5.0	10.2	10.2	19.4	18.4	16.1	16.1	103.8
	ValueDICE	31.7	21.9	22.9	27.7	70.5	68.5	69.3	68.5	380.9
	DemoDICE	12.8	31.5	12.9	86.9	15.7	24.2	2.3	1.4	187.7
	SMODICE	43.6	16.1	62.0	85.3	23.7	22.9	2.3	-5.9	249.9
	CEIL	102.8	94.8	101.9	100.7	82.0	77.0	76.4	79.8	**715.3**
C-off-LfO	ORIL (TD3+BC)	22.4	15.2	17.8	12.6	13.6	20.7	5.5	-6.2	101.6
	SMODICE	42.4	17.0	55.5	88.7	15.7	22.6	2.5	-6.3	238.1
	CEIL	67.9	12.0	68.4	50.8	31.7	57.0	18.0	-1.9	**304.0**

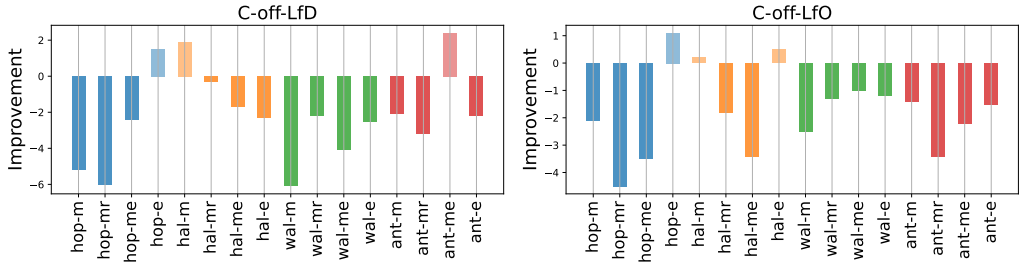

Figure 1: Normalized performance improvement (left: *C-off-LfD*, right: *C-off-LfO*) when we ablate the cross-domain regularization (Equation 9 in the main paper) in cross-domain IL settings. We can observe the general trend (in 26 out of 32 tasks) that ablating the cross-domain regularization causes negative performance improvement. hop: Hopper-v2. hal: HalfCheetah-v2. wal: Walker2d-v2. ant: Ant-v2. m: medium. me: medium-expert. mr: medium-replay. e: expert.

2.2 Generalizability on Cross-domain Offline IL Settings

In the standard cross-domain IL setting, the goal is to extract expert-relevant information from the mismatched expert demonstrations/observations (expert domain) and to mimic such expert behaviors in the training environment (training domain). Thus, we validate the performance of the learned policy in the training environment (*i.e.*, the environment where the offline data was collected). Here, we also study the generalizability of the learned policy by evaluating the learned policy in the expert environment (*i.e.*, the environment where the mismatched expert data was collected). We provide the normalized scores (*evaluated in the expert domain*) in Table 2. We can find that across a range of cross-domain offline IL tasks, CEIL consistently demonstrates better (zero-shot) generalizability compared to baselines.

2.3 Ablating the Cross-domain Regularization

We now conduct ablation studies to evaluate the importance of cross-domain regularization in Equation 9 (in the main paper). In Figure 1, we provide the performance improvement when we ablate the cross-domain regularization in two cross-domain offline IL tasks (*C-off-LfD* and *C-off-LfO*). We can find that in 26 out of 32 cross-domain tasks, ablating the regularization can cause performance to decrease (negative performance improvement), thus verifying the benefits of encouraging task-relevant embeddings.

2.4 Aggregate Results

According to Agarwal et al. [1], we report the aggregate statistics (for 16 offline IL tasks) in Figure 2. We can find that CEIL provides competitive performance consistently across a range of offline IL settings (*S-off-LfD*, *S-off-LfO*, *C-off-LfD*, and *C-off-LfO*) and outperforms prior offline baselines.

2.5 Varying the Number of Expert Trajectories

As a complement to the experimental results in the main paper, we continue to compare the performance of CEIL and baselines on more tasks when we vary the number of expert trajectories. Considering offline IL settings, we provide the results in Table 3 for the number of expert trajectories of 5, 10, 15, and 20 respectively. We can find that when varying the number of expert behaviors, CEIL can still obtain higher scores compared to baselines, which is consistent with the findings in Figure 3 in the main paper.

2.6 Limitation (Failure Modes in Online LfO Setting)

Meanwhile, we find that in the online LfO settings, CEIL's performance deteriorates severely on a few tasks, as shown in Figure 3 (Walker2d). In LfD (either on single-domain or on cross-domain IL) settings, CEIL can consistently achieve expert-level performance, but when migrating to LfO settings, CEIL suffers collapsing performance under the same number of environmental interactions.

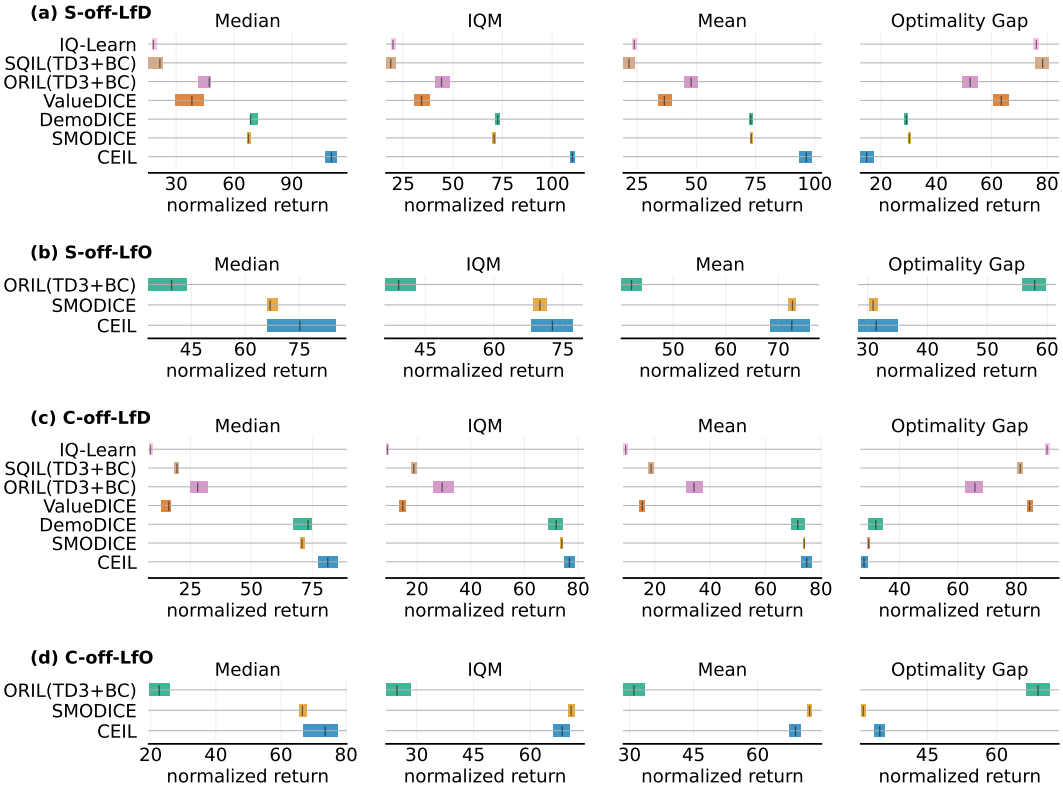

Figure 2: Aggregate median, IQM, mean, and optimality gap over 16 offline IL tasks. Higher median, higher IQM, and higher mean and lower optimality gap are better. The shaded bar shows 95% stratified bootstrap confidence intervals. We can see that CEIL achieves consistently better performance across a wide range of offline IL settings.

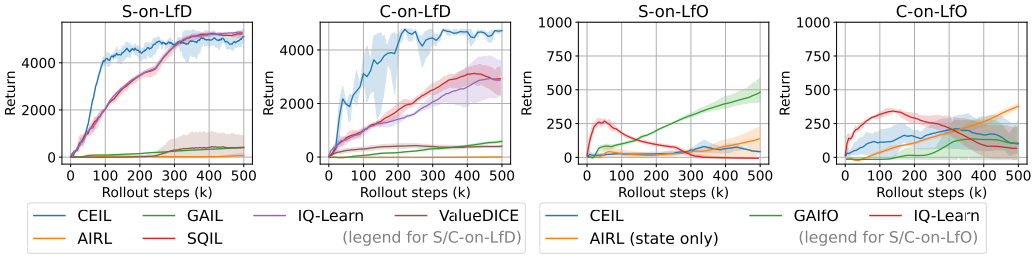

Figure 3: Return curves in Walker2d-v2 (from left to right: *S-on-LfD*, *C-on-LfD*, *S-on-LfO*, and *C-on-LfO*), where the shaded area represents a 95% confidence interval over 30 trails. We can see that CEIL consistently achieves expert-level performance in LfD (*S-on-LfD* and *C-on-LfD*) tasks. Due to the lack of explicit exploration in online LfO settings, CEIL exhibits drastic performance degradation (in *S-on-LfO* and *C-on-LfO*) under the same environmental interaction steps.

We believe that this is due to the lack of expert actions in LfO settings, which causes the agent to stay in the collapsed state region and therefore deteriorates performance. Thus, we believe a rich direction for future research is to explore the online exploration ability.

Table 3: Normalized scores (averaged over 30 trails for each task) when we vary the number of the expert demonstrations (#5, #10, #15, and #20). Scores within two points of the maximum score are highlighted

Offline IL settings		Hopper-v2			Halfcheetah-v2			Walker2d-v2			Ant-v2			sum
		m	mr	me	m	mr	me	m	mr	me	m	mr	me	
S-off-LfD #5	ORIL (TD3+BC)	42.1	26.7	51.2	45.1	2.7	79.6	44.1	22.9	38.3	25.6	24.5	6.0	408.8
	SQIL (TD3+BC)	45.2	27.4	5.9	14.5	15.7	11.8	12.2	7.2	13.6	20.6	23.6	-5.7	192.0
	IQ-Learn	17.2	15.4	21.7	6.4	4.8	6.2	13.1	10.6	5.1	22.8	27.2	18.7	169.2
	ValueDICE	59.8	80.1	72.6	2.0	0.9	1.2	2.8	0.0	7.4	27.3	32.7	30.2	316.9
	DemoDICE	50.2	26.5	63.7	41.9	38.7	59.5	66.3	38.8	101.6	82.8	68.8	112.4	751.2
	SMODICE	54.1	34.9	64.7	42.6	38.4	63.8	62.2	40.6	55.4	86.0	69.7	112.4	724.7
	CEIL	94.5	45.1	80.8	45.1	43.3	33.9	103.1	81.1	99.4	99.8	101.4	85.0	**912.5**
S-off-LfD #10	ORIL (TD3+BC)	42.0	21.6	53.4	45.0	2.1	82.1	44.1	27.4	80.4	47.3	24.0	44.9	514.1
	SQIL (TD3+BC)	50.0	34.2	7.4	8.8	10.9	8.2	20.0	15.2	9.7	35.3	36.2	11.9	247.6
	IQ-Learn	11.3	18.6	20.1	4.1	6.5	6.6	18.3	12.8	12.2	30.7	53.9	23.7	218.7
	ValueDICE	56.0	64.1	54.2	-0.2	2.6	2.4	4.7	4.0	0.9	31.4	72.3	49.5	341.8
	DemoDICE	53.6	25.8	64.9	42.1	36.9	60.6	64.7	36.1	100.2	87.4	67.1	114.3	753.5
	SMODICE	55.6	30.3	66.6	42.6	38.0	66.0	64.5	44.6	53.8	86.9	69.5	113.4	731.8
	CEIL	113.2	53.0	96.3	64.0	43.6	44.0	120.4	82.3	104.2	119.3	70.0	90.1	**1000.4**
S-off-LfD #15	ORIL (TD3+BC)	38.9	22.3	46.8	44.7	1.9	83.8	37.9	4.2	69.9	59.4	22.3	12.4	444.6
	SQIL (TD3+BC)	42.8	44.4	5.2	6.8	17.1	9.1	16.9	13.5	6.9	21.2	17.2	12.6	213.6
	IQ-Learn	14.6	8.2	29.3	4.0	3.4	5.1	7.3	14.5	11.4	54.2	15.2	61.6	228.6
	ValueDICE	66.3	58.3	53.6	2.3	2.3	1.2	5.2	-0.1	17.0	45.2	72.0	74.3	397.8
	DemoDICE	52.2	29.6	67.3	41.9	37.6	58.1	66.4	42.9	103.5	86.6	68.3	114.3	768.7
	SMODICE	55.9	25.7	72.7	42.5	37.6	66.4	67.0	43.2	55.1	86.7	69.7	118.2	740.6
	CEIL	116.4	56.7	103.7	80.4	43.0	43.8	120.3	84.8	103.8	126.8	87.0	90.6	**1057.3**
S-off-LfD #20	ORIL (TD3+BC)	50.9	22.1	72.7	44.7	30.2	87.5	47.1	26.7	102.6	46.5	31.4	61.9	624.3
	SQIL (TD3+BC)	32.6	60.6	25.5	13.2	25.3	14.4	25.6	15.6	8.0	63.6	58.4	44.3	387.1
	IQ-Learn	21.3	19.9	24.9	5.0	7.5	7.5	22.3	19.6	18.5	38.4	24.3	55.3	264.5
	ValueDICE	73.8	83.6	50.8	1.9	2.4	3.2	24.6	26.4	44.1	79.1	82.4	75.2	547.5
	DemoDICE	54.8	32.7	65.4	42.8	37.0	55.6	68.1	39.7	95.0	85.6	69.0	108.8	754.6
	SMODICE	56.1	28.7	68.0	42.7	37.7	66.9	66.2	40.7	58.2	87.4	69.9	113.4	735.9
	CEIL (ours)	110.4	103.0	106.8	40.0	30.3	63.9	118.6	110.8	117.0	126.3	122.0	114.3	**1163.5**

3 Implementation Details

3.1 Imitation Learning Tasks

In our paper, we conduct experiments across a variety of IL problem domains: single/cross-domain IL, online/offline IL, and LfD/LfO IL settings. By arranging and combining these IL domains, we obtain 8 IL tasks in all: *S-on-LfD*, *S-on-LfO*, *S-off-LfD*, *S-off-LfO*, *C-on-LfD*, *C-on-LfO*, *C-off-LfD*, and *C-off-LfO*, where S/C denotes single/cross-domain IL, on/off denotes online/offline IL, and LfD/LfO denote learning from demonstrations/observations respectively.

***S-on-LfD*.** We have access to a limited number of expert demonstrations and an online interactive training environment. The goal of *S-on-LfD* is to learn an optimal policy that mimics the provided demonstrations in the training environment.

***S-on-LfO*.** We have access to a limited number of expert observations (state-only demonstrations) and an online interactive training environment. The goal of *S-on-LfO* is to learn an optimal policy that mimics the provided observations in the training environment.

***S-off-LfD*.** We have access to a limited number of expert demonstrations and a large amount of pre-collected offline (reward-free) data. The goal of *S-off-LfD* is to learn an optimal policy that mimics the provided demonstrations in the environment in which the offline data was collected. Note that here *the environment* that was used to collect the expert demonstrations and *the environment* that was used to collect the offline data are the same environment.

***S-off-LfO*.** We have access to a limited number of expert observations and a large amount of pre-collected offline (reward-free) data. The goal of *S-off-LfO* is to learn an optimal policy that mimics the provided observations in the environment in which the offline data was collected. Note that here *the environment* that was used to collect the expert observations and *the environment* that was used to collect the offline data are the same environment.

C-on-LfD*.** We have access to a limited number of expert demonstrations and an online interactive training environment. The goal of *C-on-LfD* is to learn an optimal policy that mimics the provided demonstrations in the training environment. Note that here *the environment* that was used to collect the expert demonstrations and *the online training environment* are ***not *the same environment*.

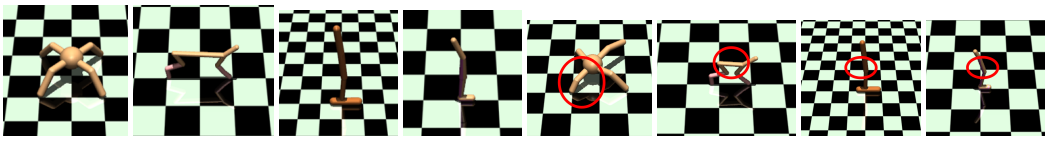

Figure 4: MuJoCo environments and our modified versions. From left to right: Ant-v2, HalfCheetah-v2, Hopper-v2, Walker2d-v2, our modified Ant-v2, our modified HalfCheetah-v2, our modified Hopper-v2, and our modified Walker2d-v2.

C-on-LfO*.** We have access to a limited number of expert observations (state-only demonstrations) and an online interactive training environment. The goal of *C-on-LfO* is to learn an optimal policy that mimics the provided observations in the training environment. Note that here *the environment* that was used to collect the expert observations and *the online training environment* are ***not *the same environment*.

C-off-LfD*.** We have access to a limited number of expert demonstrations and a large amount of pre-collected offline (reward-free) data. The goal of *C-off-LfD* is to learn an optimal policy that mimics the provided demonstrations in the environment in which the offline data was collected. Note that here *the environment* that was used to collect the expert demonstrations and *the environment* that was used to collect the offline data are ***not *the same environment*.

C-off-LfO*.** We have access to a limited number of expert observations and a large amount of pre-collected offline (reward-free) data. The goal of *C-off-LfO* is to learn an optimal policy that mimics the provided observations in the environment in which the offline data was collected. Note that here *the environment* that was used to collect the expert observations and *the environment* that was used to collect the offline data are ***not *the same environment*.

3.2 Online IL Environments, Offline IL Datasets, and One-shot tasks

Our experiments are conducted in four popular MuJoCo environments (Figure 4): Hopper-v2, HalfCheetah-v2, Walker2d-v2, and Ant-v2. For offline IL tasks, we take the standard (reward-free) D4RL dataset [4] (medium, medium-replay, medium-expert, and expert domains) as the offline dataset. For cross-domain (online/offline) IL tasks, we collect the expert behaviors (demonstrations or observations) on a modified MuJoCo environment. Specifically, we change the height of the agent's torso (as shown in Figure 4). We refer the reader to our code submission, which includes our modified MuJoCo assets. For one-shot IL tasks, we train the policy only in the single-domain IL settings (*S-on-LfD*, *S-on-LfO*, *S-off-LfD*, and *S-off-LfO*). Then we collect only one expert trajectory in the modified MuJoCo environment, and roll out the fine-tuned/inferred policy in the modified environment to test the one-shot performance.

Collecting expert behaviors. In our implementation, we use the publicly available rlkit[1] implementation of SAC to learn an expert policy and use the learned policy to collect expert behaviors (demonstrations in LfD or observations in LfO).

3.3 CEIL Implementation Details

Trajectory self-consistency loss. To learn the embedding function f_ϕ and a corresponding contextual policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z})$, we minimize the following trajectory self-consistency loss:

$$\pi_\theta, f_\phi = \min_{\pi_\theta, f_\phi} -\mathbb{E}_{\boldsymbol{\tau}_{1:T}\sim\mathcal{D}(\boldsymbol{\tau}_{1:T})}\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\boldsymbol{\tau}_{1:T}}\left[\log \pi_\theta(\mathbf{a}|\mathbf{s}, f_\phi(\boldsymbol{\tau}_{1:T}))\right],$$

where $\boldsymbol{\tau}_{1:T}$ denotes a trajectory segment with window size of T. In the online setting, we sample trajectory $\boldsymbol{\tau}$ from the experience replay buffer $\mathcal{D}(\boldsymbol{\tau})$; in the offline setting, we sample trajectory $\boldsymbol{\tau}$ directly from the given offline data $\mathcal{D}(\boldsymbol{\tau})$. Meanwhile, if we can access the expert actions (*i.e.*, LfD settings), we also incorporate the expert demonstrations into the empirical expectation (*i.e.*, storing the expert demonstrations into the online/offline experience $\mathcal{D}(\boldsymbol{\tau})$).

In our implementation, we use a 4-layer MLP (with ReLU activation) to encode the trajectory $\boldsymbol{\tau}_{1:T}$ and a 4-layer MLP (with ReLU activation) to predict the action respectively. To regularize the learning of the encoder function f_ϕ, we additionally introduce a decoder

[1] https://github.com/rail-berkeley/rlkit.

network (4-layer MLP with ReLU activation) $\pi'_\theta(\mathbf{s}'|\mathbf{s}, f_\phi(\boldsymbol{\tau}_{1:T}))$ to predict the next states: $\min_{\pi'_\theta, f_\phi} -\mathbb{E}_{\boldsymbol{\tau}_{1:T}\sim\mathcal{D}(\boldsymbol{\tau}_{1:T})}\mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\boldsymbol{\tau}_{1:T}}[\log \pi'_\theta(\mathbf{s}'|\mathbf{s}, f_\phi(\boldsymbol{\tau}_{1:T}))]$. Further, to circumvent issues of "posterior collapse" [15], we encourage learning quantized latent embeddings. In a similar spirit to VQ-VAE [15], we incorporate ideas from vector quantization (VQ) and introduce the following regularization: $\min_{f_\phi} ||\text{sg}[z_e(\boldsymbol{\tau}_{1:T})] - e||^2 + ||z_e(\boldsymbol{\tau}_{1:T}) - \text{sg}[e]||^2$, where e is a dictionary of vector quantization embeddings (we set the size of this embedding dictionary to be 4096), $z_e(\boldsymbol{\tau}_{1:T})$ is defined as the nearest dictionary embedding to $f_\phi(\boldsymbol{\tau}_{1:T})$, and $\text{sg}[\cdot]$ denotes the stop-gradient operator.

Out-level embedding inference. In Section 4.2 (main paper), we approximate $\mathcal{J}_{\text{MI}}$ with $\mathcal{J}_{\text{MI}(f_\phi)} \triangleq \mathbb{E}_{p(\mathbf{z}^*)\pi_E(\boldsymbol{\tau}_E)\pi_\theta(\boldsymbol{\tau}_\theta|\mathbf{z}^*)}\left[-\|\mathbf{z}^* - f_\phi(\boldsymbol{\tau}_E)\|^2 + \|\mathbf{z}^* - f_\phi(\boldsymbol{\tau}_\theta)\|^2\right]$, where we replace the mutual information with $-\|\mathbf{z}^* - f_\phi(\boldsymbol{\tau})\|^2$ by leveraging the learned embedding function f_ϕ. Empirically, we find that we can ignore the second loss $\|\mathbf{z}^* - f_\phi(\boldsymbol{\tau}_\theta)\|^2$, and directly conduct outer-level embedding inference with $\max_{\mathbf{z}^*, f_\phi} \mathbb{E}_{p(\mathbf{z}^*)\pi_E(\boldsymbol{\tau}_E)}\left[-\|\mathbf{z}^* - f_\phi(\boldsymbol{\tau}_E)\|^2\right]$. Meanwhile, this simplification makes the support constraints ($\mathcal{R}(\mathbf{z}^*)$ in Equation 7 in the main paper) for the offline OOD issues naturally satisfied, since $\max_{\mathbf{z}^*} \mathbb{E}_{p(\mathbf{z}^*)\pi_E(\boldsymbol{\tau}_E)}\left[-\|\mathbf{z}^* - f_\phi(\boldsymbol{\tau}_E)\|^2\right]$ and $\min_{\mathbf{z}^*} \mathcal{R}(\mathbf{z}^*)$ are equivalent.

Cross-domain IL regularization. To encourage f_ϕ to capture the task-relevant embeddings and ignore the domain-specific factors, we set the regularization $\mathcal{R}(f_\phi)$ in Equation 5 to be: $\mathcal{R}(f_\phi) = \mathcal{I}(f_\phi(\boldsymbol{\tau}); \mathbf{n})$, where we couple each trajectory $\boldsymbol{\tau}$ in $\{\boldsymbol{\tau}_E\} \cup \{\boldsymbol{\tau}_{E'}\}$ with a label $\mathbf{n} \in \{\mathbf{0}, \mathbf{1}\}$, indicating whether it is noised. In our implementation, we apply MINE [2] to estimate the mutual information and conduct encoder regularization. Specifically, we estimate $\mathcal{I}(\mathbf{z}; \mathbf{n})$ with $\hat{\mathcal{I}}(\mathbf{z}; \mathbf{n}) := \sup_\delta \mathbb{E}_{p(\mathbf{z},\mathbf{n})}[f_\delta(\mathbf{z}, \mathbf{n})] - \log \mathbb{E}_{p(\mathbf{z})p(\mathbf{n})}[\exp(f_\delta(\mathbf{z}, \mathbf{n}))]$ and regularize the encoder f_ϕ with $\max_{f_\phi} \hat{\mathcal{I}}(f_\phi(\boldsymbol{\tau}); \mathbf{n})$, where we model f_δ with a 4-layer MLP (using ReLU activations).

Hyper-parameters. In Table 4, we list the hyper-parameters used in the experiments.

Table 4: CEIL hyper-parameters.

Parameter	Value
size of the embedding dictionary	4096
size of the embedding dimension	16
trajectory window size	2
encoder: optimizer	Adam
encoder: learning rate	3e-4
encoder: learning rate scheduler	CosineAnnealingWarmRestarts(T_0 = 1000,T_mult=1, eta_min=1e-5)
encoder: number of hidden layers	4
encoder: number of hidden units per layer	512
encoder: nonlinearity	ReLU
policy: optimizer	Adam
policy: learning rate	3e-4
policy: learning rate scheduler	CosineAnnealingWarmRestarts(T_0 = 1000,T_mult=1, eta_min=1e-5)
policy: number of hidden layers	4
policy: number of hidden units per layer	512
policy: nonlinearity	ReLU
decoder: optimizer	Adam
decoder: learning rate	3e-4
decoder: learning rate scheduler	CosineAnnealingWarmRestarts(T_0 = 1000,T_mult=1, eta_min=1e-5)
decoder: number of hidden layers	4
decoder: number of hidden units per layer	512
decoder: nonlinearity	ReLU

Table 5: Baseline methods and their code-bases.

Baselines	Code-bases
GAIL, GAIfO, AIRL	https://github.com/HumanCompatibleAI/imitation
SAIL	https://github.com/FangchenLiu/SAIL
IQ-Learn, SQIL	https://github.com/Div99/IQ-Learn
ValueDICE	https://github.com/google-research/google-research/tree/master/value_dice
DemoDICE	https://github.com/KAIST-AILab/imitation-dice
SMODICE, ORIL	https://github.com/JasonMa2016/SMODICE

3.4 Baselines Implementation Details

We summarize our code-bases of our baseline implementations in Table 5 and describe each baseline as follows:

Generative Adversarial Imitation Learning (GAIL). GAIL [8] is a GAN-based online LfD method that trains a policy (generator) to confuse a discriminator trained to distinguish between generated transitions and expert transitions. While the goal of the discriminator is to maximize the objective below, the policy is optimized via an RL algorithm to match the expert occupancy measure (minimize the objective below):

$$\mathcal{J}(\pi, D) = \mathbb{E}_{\pi}\left[\log(D(s,a))\right] + \mathbb{E}_{\pi_E}\left[1 - \log(D(s,a))\right] - \lambda H(\pi).$$

We used the implementation by Gleave et al. [7] on the GitHub page[2], where there are two modifications introduced with respect to the original paper: 1) a higher output of the discriminator represents better, 2) PPO is used to optimize the policy instead of TRPO.

Generative Adversarial Imitation from Observations (GAIfO). GAIfO [14] is an online LfO method that applies the principle of GAIL and utilizes a state-only discriminator to judge whether the generated trajectory matches the expert trajectory in terms of states. We provide the objective of GAIfO as follows:

$$\mathcal{J}(\pi, D) = \mathbb{E}_{\pi}\left[\log(D(s,s'))\right] + \mathbb{E}_{\pi_E}\left[1 - \log(D(s,s'))\right] - \lambda H(\pi).$$

Based on the implementation of GAIL, we implement GAIfO by changing the input of the discriminator to state transitions.

Adversarial Inverse Reinforcement Learning (AIRL). AIRL [3] is an online LfD/LfO method using an adversarial learning framework similar to GAIL. It modifies the form of the discriminator to explicitly disentangle the task-relevant information from the transition dynamics. To make the policy more generalized and less sensitive to dynamics, AIRL proposes to learn a parameterized reward function using the output of the discriminator:

$$f_{\theta,\phi}(s,a,s') = g_{\theta}(s,a) + \lambda h_{\phi}(s') - h_{\phi}(s),$$
$$D_{\theta,\phi}(s,a,s') = \frac{\exp(f_{\theta,\phi}(s,a,s'))}{\exp(f_{\theta,\phi}(s,a,s')) + \pi(a|s)}.$$

Similarly to GAIL, we used the code provided by Gleave et al. [7], and the RL algorithm is also PPO.

State Alignment-based Imitation Learning (SAIL). SAIL [11] is an online LfO method capable of solving cross-domain tasks. SAIL aims to minimize the divergence between the policy rollout and the expert trajectory from both local and global perspectives: 1) locally, a KL divergence between the policy action and the action predicted by a state planner and an inverse dynamics model, 2) globally, a Wasserstein divergence of state occupancy between the policy and the expert. The policy is optimized using:

$$\begin{aligned}\mathcal{J}(\pi) &= -D_{\mathcal{W}}(\pi(s)\|\pi_E(s)) - \lambda D_{KL}(\pi(\cdot|s_t)\|\pi_E(\cdot|s_t)) \\ &= \mathbb{E}_{\pi(s_t,a_t,s_{t+1})}\left(\sum_{t=1}^{T}\frac{D(s_{t+1}) - \mathbb{E}_{\pi_E(s)}D(s)}{T}\right) - \lambda D_{KL}\Big(\pi\big(\cdot\,|s_t\big)\|g_{\text{inv}}\big(\cdot\,|s_t, f(s_t)\big)\Big),\end{aligned}$$

where D is a state-based discriminator trained via $\mathcal{J}(D) = \mathbb{E}_{\pi_E}\left[D(s)\right] - \mathbb{E}_{\pi}\left[D(s)\right]$, f is the pretrained VAE-based state planner, and g_{inv} is the inverse dynamics model trained by supervised regression.

In the online setting, we use the official implementation published by the authors[3], where SAIL is optimized using PPO with the reward definition: $r(s_t, s_{t+1}) = \frac{1}{T}\left[D(s_{t+1}) - \mathbb{E}_{\pi_E(s)}D(s)\right]$. Besides, we further implement SAIL in the offline setting by using TD3+BC [5] to maximize the reward defined above.

In our experiments, we empirically discover that SAIL is computationally expensive. While SAIL is able to learn tasks in the typical IL setting (*S-on-LfD*), our early experimental results find that

[2] https://github.com/HumanCompatibleAI/imitation

[3] https://github.com/FangchenLiu/SAIL

SAIL(TD3+BC) with heavy hyperparameter tuning failed on the offline setting. This indicates that SAIL is rather sensitive to the dataset composition, which also coincides with the results gathered in Ma et al. [12]. Thus, we do not include SAIL in our comparison results.

Soft-Q Imitation Learning (SQIL). SQIL [13] is a simple but effective single-domain LfD IL algorithm that is easy to implement with both online and offline Q-learning algorithms. The main idea of SQIL is to give sparse rewards (+1) only to those expert transitions and zero rewards (0) to those experiences in the replay buffer. The Q-function of SQIL is updated using the squared soft Bellman Error:

$$\delta^2(\mathcal{D}, r) \triangleq \frac{1}{|\mathcal{D}|} \sum_{(s,a,s') \in \mathcal{D}} \left(Q(s,a) - \left(r + \gamma \log \big(\sum_{a' \in \mathcal{A}} \exp(Q(s',a')) \big) \right) \right)^2.$$

The overall objective of the Q-function is to maximize the following objective:

$$\mathcal{J}(Q) = -\delta^2(\mathcal{D}_E, 1) - \delta^2(\mathcal{D}_\pi, 0).$$

In our experiments, the online imitation policy is optimized using SAC which is also used in the original paper. To make a fair comparison among the offline IL baselines, the offline policy is optimized via TD3+BC.

Offline Reinforced Imitation Learning (ORIL). ORIL [16] is an offline single-domain IL method that solves both LfD and LfO tasks. To relax the hard-label assumption (like the sparse rewards made in SQIL), ORIL treats the experiences stored in the replay buffer as unlabelled data that could potentially include both successful and failed trajectories. More specifically, ORIL aims to train a reward function to distinguish between the expert and the suboptimal data without explicitly knowing the negative labels. By incorporating Positive-unlabeled learning (PU-learning), the objective of the reward model can be written as follows (for the LfD setting):

$$\mathcal{J}(R) = \eta \mathbb{E}_{\pi_E(s,a)} \left[\log(R(s,a)) \right] + \mathbb{E}_{\pi(s,a)} \left[\log(1 - R(s,a)) \right] - \eta \mathbb{E}_{\pi_E(s,a)} \left[\log(1 - R(s,a)) \right],$$

where η is the relative proportion of the expert data and we set it as 0.5 throughout our experiments. In the original paper, the policy learning algorithm of ORIL is Critic Regularized Regression (CRR), while in this paper, we implemented ORIL using TD3+BC for fair comparisons. Besides, we adapted ORIL to the LfO setting by learning a state-only reward function:

$$\mathcal{J}(R) = \eta \mathbb{E}_{\pi_E(s,s')} \left[\log(R(s,s')) \right] + \mathbb{E}_{\pi(s,s')} \left[\log(1 - R(s,s')) \right] - \eta \mathbb{E}_{\pi_E(s,s')} \left[\log(1 - R(s,s')) \right].$$

Inverse soft-Q learning (IQ-Learn). IQ-Learn [6] is an IRL-based method that can solve IL tasks in the online/offline and LfD/LfO settings. It proposes to directly learn a Q-function from demonstrations and avoid the intermediate step of reward learning. Unlike GAIL optimizing a min-max objective defined in the reward-policy space, IQ-Learn solves the expert matching problem directly in the policy-Q space. The Q-function is trained to maximize the objective:

$$\mathbb{E}_{\pi_E(s,a,s')} \left[Q(s,a) - \gamma V^\pi(s') \right] - \mathbb{E}_{\pi(s,a,s')} \left[Q(s,a) - \gamma V^\pi(s') \right] - \psi(r),$$

where $V^\pi(s) \triangleq \mathbb{E}_{a \sim \pi(\cdot|s)} \left[Q(s,a) - \log \pi(a|s) \right]$, $\psi(r)$ is a regularization term calculated over the expert distribution. Then, the policy is learned by SAC.

We use the code provided in the official IQ-learn repository[4] and reproduce the online-LfD results reported in the original paper. For online tasks, we empirically find that penalizing the Q-value on the initial states gives the best and most stabilized performance. The learning objective of the Q-function for the online tasks is:

$$\mathcal{J}(Q) = \mathbb{E}_{\pi_E(s,a,s')} \left[Q(s,a) - \gamma V^\pi(s') \right] - (1-\gamma) \mathbb{E}_{\rho_0} \left[V^\pi(s_0) \right] - \psi(r).$$

In the offline setting, we find that using the above objective easily leads to an overfitting issue, causing collapsed performance. Thus, we follow the instruction provided in the paper and only penalize the expert samples:

$$\begin{aligned} \mathcal{J}(Q) &= \mathbb{E}_{\pi_E(s,a,s')} \left[Q(s,a) - \gamma V^\pi(s') \right] - \mathbb{E}_{\pi_E(s,a,s')} \left[V^\pi(s) - \gamma V^\pi(s') \right] - \psi(r) \\ &= \mathbb{E}_{\pi_E(s,a,s')} \left[Q(s,a) - V^\pi(s) \right] - \psi(r). \end{aligned}$$

[4] https://github.com/Div99/IQ-Learn

Imitation Learning via Off-Policy Distribution Matching (ValueDICE). ValueDICE [10] is a DICE-based[5] LfD algorithm which minimizes the divergence of state-action distributions between the policy and the expert. In contrast to the state-conditional distribution of actions $\pi(\cdot|s)$ used in the above methods, the state-action distribution, $d^\pi(s,a) : \mathcal{S} \times \mathcal{A} \to [0,1]$, can uniquely characterize a one-to-one correspondence,

$$d^\pi(s,a) \triangleq (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \mathbf{Pr}(s_t = s, a_t = a \,|s_0 \sim \rho_0, a_t \sim \pi(s_t), s_{t+1} \sim P(s_t, a_t)).$$

Thus, the plain expert matching objective can be reformulated and expressed in the Donsker-Varadhan representation:

$$\begin{aligned}\mathcal{J}(\pi) &= -D_{KL}(d^\pi(s,a)\|d^{\pi_E}(s,a)) \\ &= \min_{x:\mathcal{S}\times\mathcal{A}\to\mathbb{R}} \log \mathbb{E}_{(s,a)\sim d^{\pi_E}}\left[\exp(x(s,a))\right] - \mathbb{E}_{(s,a)\sim d^\pi}\left[x(s,a)\right].\end{aligned}$$

The objective above can be expanded further by defining $x(s,a) = v(s,a) - \mathcal{B}^\pi v(s,a)$ and using a zero-reward Bellman operator $\mathcal{B}^\pi$ to derive the following (adversarial) objective:

$$\mathcal{J}_{DICE}(\pi, v) = \log \mathbb{E}_{(s,a)\sim d^{\pi_E}}\left[\exp\big(v(s,a) - \mathcal{B}^\pi v(s,a)\big)\right] - (1-\gamma)\mathbb{E}_{s_0\sim\rho_0, a_0\sim\pi(\cdot|s_0)}\left[v(s_0,a_0)\right].$$

We use the official Tensorflow implementation[6] in our experiments. In the online setting, the rollouts collected are used as an additional replay regularization. The overall objective in the online setting is:

$$\begin{aligned}&\mathcal{J}^{mix}_{DICE}(\pi, v) \\ &= -D_{KL}\big((1-\alpha)d^\pi(s,a) + \alpha d^{RB}(s,a)\|(1-\alpha)d^{\pi_E}(s,a) + \alpha d^{RB}(s,a)\big) \\ &= \log \mathbb{E}_{(s,a)\sim d^{mix}}\left[\exp\big(v(s,a) - \mathcal{B}^\pi v(s,a)\big)\right] - (1-\alpha)(1-\gamma)\,\mathbb{E}_{s_0\sim\rho_0,\, a_0\sim\pi(\cdot|s_0)}\left[v(s_0,a_0)\right] \\ &\qquad\qquad -\alpha\,\mathbb{E}_{(s,a)\sim d^{RB}}\left[v(s,a) - \mathcal{B}^\pi v(s,a)\right],\end{aligned}$$

where $d^{mix} \triangleq (1-\alpha)d^{\pi_E} + \alpha d^{RB}$ and α is a non-negative regularization coefficient (we set α as 0.1 following the specification of the paper).

In the offline setting, ValueDICE only differs in the source of sampling data. We change the online replay buffer to the offline pre-collected dataset.

Offline Imitation Learning with Supplementary Imperfect Demonstrations (DemoDICE). DemoDICE [9] is a DICE-based offline LfD method that assumes to have access to an offline dataset collected by a behavior policy π_β. Using this supplementary dataset, the expert matching objective of DemoDICE is instantiated over ValueDICE:

$$-D_{KL}(d^\pi(s,a)\|d^{\pi_E}(s,a)) - \alpha D_{KL}(d^\pi(s,a)\|d^{\pi_\beta}(s,a)),$$

where α is a positive weight for the constraint.

The above optimization objective can be transformed into three tractable components: 1) a reward function $r(s,a)$ derived by pre-training a binary discriminator $D : \mathcal{S} \times \mathcal{A} \to [0,1]$:

$$\begin{aligned}r(s,a) &= -\log(\frac{1}{D^*(s,a)} - 1), \\ D^*(s,a) &= \arg\max_D = \mathbb{E}_{d^{\pi_E}}\left[\log D(s,a)\right] + \mathbb{E}_{d^{\pi_\beta}}\left[\log(1 - D(s,a))\right],\end{aligned}$$

2) a value function optimization objective:

$$\mathcal{J}(v) = -(1-\gamma)\mathbb{E}_{s\sim\rho_0}\left[v(s)\right] - (1+\alpha)\log \mathbb{E}_{(s,a)\sim d^{\pi_\beta}}\left[\exp(\frac{r(s,a) + \mathbb{E}_{s'\sim P(s,a)}(v(s')) - v(s)}{1+\alpha})\right],$$

and 3) a policy optimization step:

$$\begin{aligned}\mathcal{J}(\pi) &= \mathbb{E}_{(s,a)\sim d^{\pi_\beta}}\left[v^*(s,a)\log\pi(a|s)\right], \\ v^*(s,a) &= \arg\max_v \mathcal{J}(v).\end{aligned}$$

[5] DICE refers to stationary DIstribution Estimation Correction

[6] https://github.com/google-research/google-research/tree/master/value_dice

We report the offline results using the official Tensorflow implementation[7].

State Matching Offline DIstribution Correction Estimation (SMODICE). SMODICE [12] proposes to solve offline IL tasks in LfO and cross-domain settings and it optimizes the following state occupancy objective:

$$-D_{KL}(d^{\pi}(s)\|d^{\pi_E}(s)).$$

To incorporate the offline dataset, SMODICE derives an f-divergence regularized state-occupancy objective:

$$\mathbb{E}_{s\sim d^{\pi}(s)}\left[\log(\frac{d^{\pi_\beta}(s)}{d^{\pi_E}(s)})\right] + -D_f(d^{\pi}(s,a)\|d^{\pi_\beta}(s,a)).$$

Intuitively, the first term can be interpreted as matching the offline states towards the expert states, while the second regularization term constrains the policy close to the offline distribution of state-action occupancy. Similarly, we can divide the objective into three steps: 1) deriving a state-based reward by learning a state-based discriminator:

$$\begin{aligned} r(s,a) &= -\log(\frac{1}{D^*(s)} - 1), \\ D^*(s,a) &= \arg\max_D = \mathbb{E}_{d^{\pi_E}}\left[\log D(s)\right] + \mathbb{E}_{d^{\pi_\beta}}\left[\log(1-D(s))\right], \end{aligned}$$

2) learning a value function using the learned reward:

$$\mathcal{J}(v) = -(1-\gamma)\mathbb{E}_{s\sim\rho_0}\left[v(s)\right] - \log \mathbb{E}_{(s,a)\sim d^{\pi_\beta}}\left[f_*(r(s,a) + \mathbb{E}_{s'\sim P(s,a)}(v(s')) - v(s))\right],$$

and 3) training the policy via weighted regression:

$$\begin{aligned} \mathcal{J}(\pi) &= \mathbb{E}_{(s,a)\sim d^{\pi_\beta}}\left[f'_*(r(s,a) + \mathbb{E}_{s'\sim P(s,a)}(v^*(s')) - v^*(s))\log\pi(a|s)\right], \\ v^*(s,a) &= \arg\max_v \mathcal{J}(v), \end{aligned}$$

where f_* is the Fenchel conjugate of f-divergence (please refer to Ma et al. [12] for more details).

We conduct experiments using the official Pytorch implementation [8], where the f-divergence used is $\mathcal{X}^2$-divergence. On the LfD tasks, we change the input of the discriminator to state-action pairs.

References

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

[2] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. Mine: mutual information neural estimation. *arXiv preprint arXiv:1801.04062*, 2018.

[3] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[4] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[5] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

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

[7] https://github.com/KAIST-AILab/imitation-dice

[8] https://github.com/JasonMa2016/SMODICE

[7] Adam Gleave, Mohammad Taufeeque, Juan Rocamonde, Erik Jenner, Steven H. Wang, Sam Toyer, Maximilian Ernestus, Nora Belrose, Scott Emmons, and Stuart Russell. imitation: Clean imitation learning implementations, 2022.

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

[9] Geon-Hyeong Kim, Seokin Seo, Jongmin Lee, Wonseok Jeon, HyeongJoo Hwang, Hongseok Yang, and Kee-Eung Kim. Demodice: Offline imitation learning with supplementary imperfect demonstrations. In *International Conference on Learning Representations*, 2022.

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

[11] Fangchen Liu, Zhan Ling, Tongzhou Mu, and Hao Su. State alignment-based imitation learning. *arXiv preprint arXiv:1911.10947*, 2019.

[12] Yecheng Jason Ma, Andrew Shen, Dinesh Jayaraman, and Osbert Bastani. Smodice: Versatile offline imitation learning via state occupancy matching. *arXiv e-prints*, pages arXiv–2202, 2022.

[13] Siddharth Reddy, Anca D Dragan, and Sergey Levine. Sqil: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*, 2019.

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

[15] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[16] Konrad Zolna, Alexander Novikov, Ksenia Konyushkova, Caglar Gulcehre, Ziyu Wang, Yusuf Aytar, Misha Denil, Nando de Freitas, and Scott Reed. Offline learning from demonstrations and unlabeled experience. *arXiv preprint arXiv:2011.13885*, 2020.