Distributional Cloning for Stabilized Imitation Learning via ADMM

Xin Zhang
San Diego State University
xzhang19@sdsu.edu

Yanhua Li
Worcester Polytechnic Institute
yli15@wpi.edu

Ziming Zhang
Worcester Polytechnic Institute
zzhang15@wpi.edu

Christopher G. Brinton
Purdue University
cgb@purdue.edu

Zhenming Liu
College of William & Mary
zliu@cs.wm.edu

Zhi-Li Zhang
University of Minnesota
zhzhang@cs.umn.edu

***Abstract*—The two leading solution paradigms for imitation learning (IL), BC and GAIL, each suffers from notable drawbacks. BC, a supervised learning approach to mimic expert actions, is vulnerable to covariate shift. GAIL applies adversarial training to minimize the discrepancy between expert and learner behaviors, which is prone to unstable training and mode collapse. In this work, we propose DC – Distributional Cloning – a novel IL approach for addressing the covariate shift and mode collapse problems simultaneously. DC directly maximizes the likelihood of observed expert and learner demonstrations, and gradually encourages the learner to evolve towards expert behaviors based on an averaging effect. The DC solution framework contains two stages in each training loop, where in stage one the mixed expert and learner state distribution is estimated via SoftFlow, and in stage two the learner policy is trained to match both the expert's policy and state distribution via ADMM. Experimental evaluation of DC compared with several baselines in 10 different physics-based control tasks reveal superior results in learner policy performance, training stability, and mode distribution preservation.**

***Index Terms*—imitation learning, neural ordinary differential equations**

I. Introduction

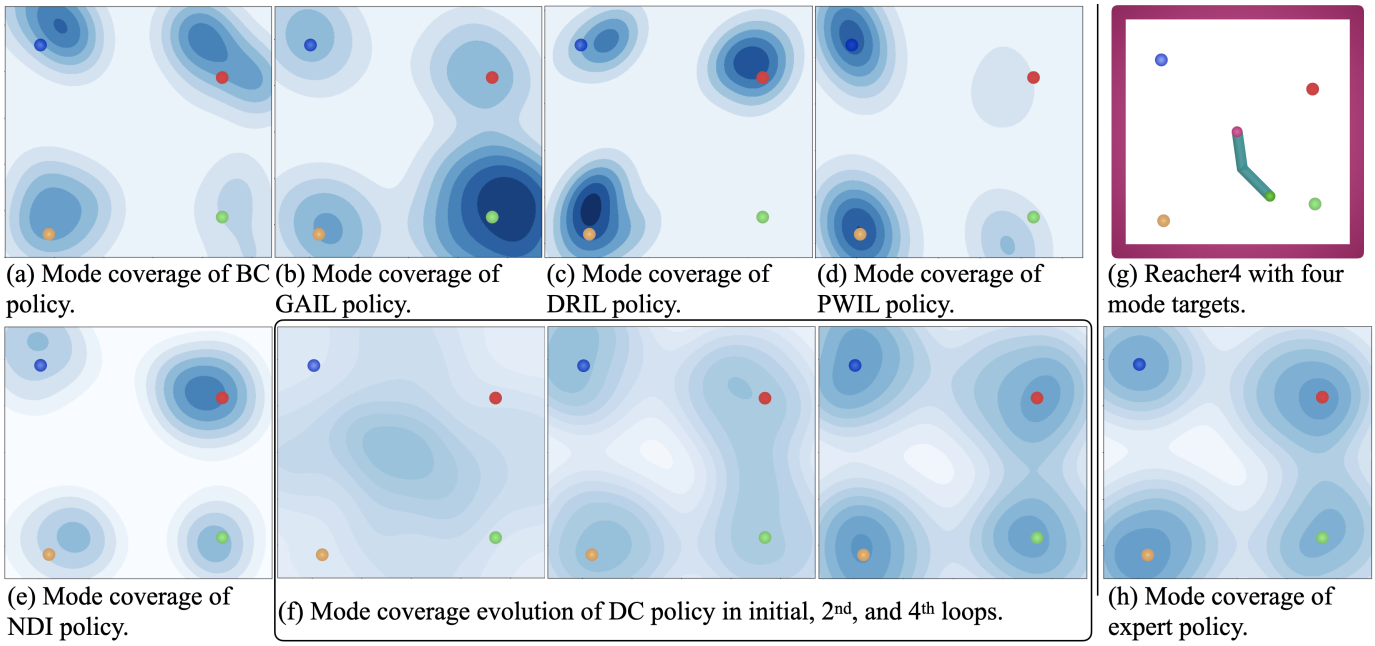
Imitation learning (IL) [2], [3] aims to learn sequential decision-making policies directly from expert demonstrations, without access to reward signals from the environment. State-of-the-art (SOTA) IL approaches have primarily followed one of two paradigms: behavior cloning (BC) [4] and generative adversarial imitation learning (GAIL) [3].

Motivation. While both BC and GAIL have been studied extensively, each has crucial limitations. On the one hand, BC approaches employ supervised learning, which requires a large amount of expert demonstrations to avoid compounding errors due to covariate shifts [5], [6]. On the other hand, GAIL approaches [7]–[18] connect IL with generative adversarial networks (GAN) [19], but adversarial training processes are intrinsically unstable [7], [15] and prone to mode collapse (especially when learning from multi-mode expert demonstrations) [8], [12], [13]. To see this, consider Figure 1, which shows an example of different IL algorithms on the Reacher task [10], [20] with four targets (Figure 1g). While the expert tends to visit all targets equally (Figure 1h), the policy learned by BC does not recover the expert distribution precisely due to covariate shift (Figure 1a), and that learned by GAIL is mode collapsed, primarily visiting the green target (Figure 1b). In this work, we are motivated to design an IL methodology that alleviates the covariate shift problem with a stabilized training process and mode distribution preservation property from expert demonstrations.

Our Distributional Cloning (DC). We propose DC – Distributional Cloning – a novel IL approach that directly maximizes the likelihood of observed expert and learner demonstrations, and encourages the learner to evolve towards expert behaviors via an averaging effect. DC contains two stages in each training loop, with the 1^{st} stage focusing on accurately estimating the expert and learner state distribution, and the 2^{nd} stage pushing the learner policy towards the expert's policy. Figure 1f shows a quick view of our results: DC gradually learns the expert behavior distribution and leads to a learner policy preserving the mode visitation distribution of the expert demonstrations. Our key contributions are as follows:

- We are the first to propose and formulate the distributional cloning (DC) problem as maximizing expert and learner state-action distribution likelihood. The DC learner policy evolves towards expert from guidance constantly averaged over the expert and learner behaviors. It tackles the covariate shift problem of BC and the training instability issue of GAIL.
- We design a DC solution framework based on alternating direction method of multipliers (ADMM) [21] to efficiently update the learner policy, and encourage it to match expert state-action distribution.
- Our evaluation on ten different physics-based control tasks reveals that DC obtains superior results compared with SOTA baselines in learner policy performances, training stability, and mode distribution preservation. *We made our code available to contribute to the research community via a Github link* [1].

[1]The code for our experiments is available at https://github.com/XinZhang525/DC.



(a) Mode coverage of BC policy. (b) Mode coverage of GAIL policy. (c) Mode coverage of DRIL policy. (d) Mode coverage of PWIL policy. (g) Reacher4 with four mode targets.

(e) Mode coverage of NDI policy. (f) Mode coverage evolution of DC policy in initial, 2nd, and 4th loops. (h) Mode coverage of expert policy.

Fig. 1: Example results obtained by DC (Ours) and baselines on mode coverage. Right: (g) shows a Reacher task, with four targets in different colors, (h) shows the mode coverage (*i.e.*, state distribution) with expert policy. Left: (a)-(e) show the mode coverage with BC policy (a), GAIL policy (b), DRIL policy (c), PWIL policy (d), and NDI policy (e). (f) shows the mode coverage evolution (*i.e.*, initial, 2nd and 4th loops) of a DC policy with the number of training loops. All the distributions are visualized using kernel density estimation (KDE) [1]. A darker color indicates a more densely distributed region in the state space. None of the compared approaches solve the mode collapse problem.

II. Preliminaries

Notations. We denote $\mathcal{S}$ as a set of states, $\mathcal{A}$ as a set of actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ as the transition probability distribution, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ as the reward function, $\rho_0 : \mathcal{S} \mapsto \mathbb{R}$ as the distribution of the initial state s_0, and $\gamma \in [0, 1]$ as the discount factor. An agent makes decisions following a policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, which specifies a probability distribution of choosing an action $a \in \mathcal{A}$ at a state $s \in \mathcal{S}$. With $s_0 \sim \rho_0$, then over time t, $a_t \sim \pi(a_t|s_t)$ and $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)$ according to the policy π. We will denote the expert policy as π_E, and the learner policy as π. We denote $P_\pi(s, a)$ as the probability of observing a state-action pair (s, a) when executing the learner policy π, and denote $P_E(s, a)$ to represent $P_{\pi_E}(s, a)$ for brevity. Moreover, for a function $h(s, a)$ of interest, we use an expectation over a policy π to denote an expectation with respect to the trajectories it generates, *i.e.*, $\mathbb{E}_\pi[h(s, a)] \triangleq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t h(s_t, a_t)]$.

Behavior Cloning (BC). BC [4], [22] is a supervised learning solution to IL. It learns expert policy π_E via minimizing the KL divergence [23] between the learner policy π and expert trajectories $\mathcal{D}_E$, *i.e.*,

$$\max_{\pi} \mathbb{E}_{(s,a)\in\mathcal{D}_E}[\log \pi(a|s)]. \tag{1}$$

BC indeed maximizes the likelihood of observed trajectories in $\mathcal{D}_E$ when viewing transition probabilities as optimization constants[2]. Considering limited amount of expert demonstration, BC can not view all states during training and suffers from the covariate shift problem [5], [6].

III. Our DC Approach

A. Problem Definition: Distributional Cloning

To avoid the covariate shift from supervised approaches (*e.g.*, in BC) and the instability from adversarial training (*e.g.*, in GAIL), we introduce a *distributional cloning* paradigm to learn the policy π by directly maximizing the likelihood of the state-action pairs from the expert demonstration data $\mathcal{D}_E$ and learner demonstrations $\mathcal{D}_\pi$. The DC objective can be formally modeled as the following optimization problem:

$$\max_{\pi, P} \mathbb{E}_{(s,a)\in\mathcal{D}_E\cup\mathcal{D}_\pi}[\log P(s, a; \pi)]. \tag{2}$$

Here, $P(s, a; \pi)$ denotes the state-action distribution under the learner policy π. Eq.(2) encourages the learner policy to match expert state-action distribution.

Differences from BC. Notice that instead of maximizing trajectory likelihood as in BC, DC maximizes the likelihood of observed state-action pairs from both the expert and the learner demonstrations. Enforcing information from learner demonstrations $\mathcal{D}_\pi$ familiarizes the learner policy with more

[2]The log likelihood of a trajectory τ is $\log P(\tau) = \log \rho_0(s_0) + \sum_{t=0}^{\infty} \log \pi(a_t|s_t) + \sum_{t=0}^{\infty} \mathcal{P}(s_{t+1}|s_t, a_t)$. Terms $\log \rho_0(s_0)$ and $\log \mathcal{P}(s_t|s_{t-1}, a_{t-1})$ are irrelevant constants.

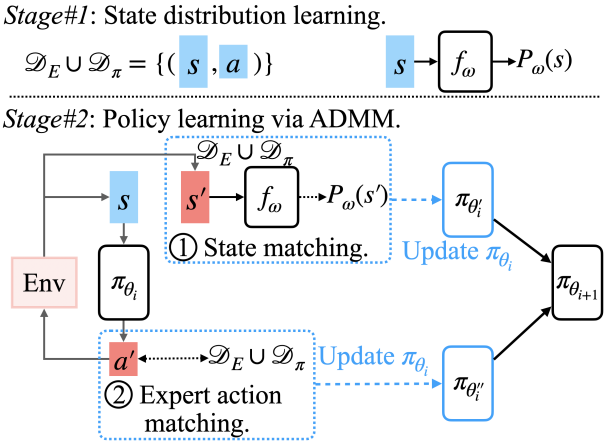


Fig. 2: Illustration of DC framework in one training loop.

expert-like behaviors and can help alleviate the covariate shift problem. In addition, rather than stochastic gradient descent (SGD) update, DC updates the learner policy π via ADMM to combine and coordinate influences from expert action matching and expert state matching.

Challenges. The DC problem is challenging from two aspects: *C#1 Implicit learner policy:* Eq.(2) optimizes the joint distribution of state-action pairs $P(s, a; \pi)$ without an explicit form of the learner policy π. *C#2 Non-differentiability:* The optimization of the DC problem in Eq.(2) requires sampling over the learner policy π which breaks its differentiabilty and makes the optimization hard.

To address these challenges, we propose the Distributional Cloning (DC) framework shown in Figure 2. DC solves the DC problem by decomposing the state-action joint distribution $P(s, a; \pi)$ with the state distribution $P(s)$ and the learner policy $\pi(a|s)$. It iteratively imposes expert guidance to the learner, and updates the learner policy via ADMM [21] to handle the non-differentiability problem formalized next.

B. DC Solution Framework

We decompose the state-action joint distribution $P(s, a; \pi)$ with a state distribution $P(s)$ and a learner policy $\pi(a|s)$ and formulate $P(s, a; \pi) = P(s)\pi(a|s)$. Then, we derive from the DC problem in Eq.(2) to the following DC objective:

$$\max_{\pi, P} \mathbb{E}_{(s,a)\in\mathcal{D}_E\cup\mathcal{D}_\pi}[\log P(s) + \log \pi(a|s)]. \tag{3}$$

To solve this problem, we model the learner policy π as a deep neural network parameterized by θ, and the expert-learner mixed state distribution P using SoftFlow [24] parameterized by ω. We apply SoftFlow as it is able to deal with the manifold hypothesis [25], [26] and is more effective in uncovering distribution modes. The training process in each loop involves two stages, described in Figure 2 and Alg. 1: *(Stage #1) state distribution learning*, where we train P_ω to learn an expert-learner mixed state distribution based on SoftFlow; *(Stage #2) policy learning*, where we use expert and learner demonstrations $\mathcal{D}_E$ and $\mathcal{D}_\pi$ and the learned state distribution P_ω to train the learner policy π_θ with the objective in Eq.(3).

Algorithm 1 Distributional Cloning (DC)

Require: Initial parameters ω_0 for the ODE system function f_ω of P_ω; initial parameters θ_0 for the learner policy π_θ; expert trajectories $\mathcal{T}_E$ containing sequences of state-action pairs; total number of training loops L and number of learner policy updates N.

Ensure: The ODE system function f_ω of the state distribution P_ω, learned policy π_θ.

1: Sample state-action pairs $\mathcal{D}_E$ from $\mathcal{T}_E$.
2: Update ω_0 to ω_1 by ascending with the gradients: $\Delta_{\omega_0} = \mathbb{E}_{s\in\mathcal{D}_E}[\nabla_{\omega_0} \log P_{\omega_0}(s)]$. // *Stage #1*
3: Update θ_0 to θ_1 for N iterations via ADMM in Alg. 2. // *Stage #2*
4: **for** each loop $i = 1, 2, \cdots, L$ **do**
5: Sample trajectories with π_{θ_i} and store in $\mathcal{T}_{\pi_{\theta_i}}$.
6: Sample state-action pairs $\mathcal{D}_E$ and $\mathcal{D}_{\pi_{\theta_i}}$ respectively from $\mathcal{T}_E$ and $\mathcal{T}_{\theta_i}$ with the same batch size.
7: Update ω_i to ω_{i+1} by ascending with the gradients: $\Delta_{\omega_i} = \mathbb{E}_{s\in\mathcal{D}_E\cup\mathcal{D}_\pi}[\nabla_{\omega_i} \log P_{\omega_i}(s)]$. // *Stage #1*
8: Update θ_i to θ_{i+1} for N iterations via ADMM in Alg. 2. // *Stage #2*
9: **end for**

Considering that initial random learner policy does not provide meaningful information for learning expert state distribution, we only use expert demonstrations $\mathcal{D}_E$ to train P_ω in the initial loop. After the first loop, the learner policy π obtains some knowledge of the expert whose demonstrations will be used to train the next loop state distribution P_ω to amplify the influence of expert state distributions and further assist learner policy training.

However, $P(s, a; \pi)$ decomposition does not solve the non-differentiability challenge: updating the policy π_θ with P_ω guidance involves action sampling which breaks the differentiablity when updating π_θ from expert demonstrations $\mathcal{D}_E$. Thus, we apply ADMM [21] and breaks the optimization problem over the learner policy π in Eq.(3) into two pieces, *i.e.*, BC (as expert action matching), and RL with P_ω guidance (as state matching). With the learned P_ω, we follow an RL approach and employ proximal policy gradient (PPO) [27] for back-propagation, with the Adam optimizer [28] for updating the policy π_θ as is detailed in the following section.

C. ADMM for Learner Policy Update

Eq.(3) suggests two directions of learner policy π_θ update, *i.e.*, an expert action matching update, and a state matching update from learner demonstrations based on demonstration state distribution $\log P_\omega(s)$. The sampling process in the state matching step breaks the differentiability for learner policy π_θ update, making it incompatible with the expert action matching update. Consequently, we cannot use direct gradient ascent. Additionally, directly viewing $\log P(s) + \log \pi(a|s)$ as a reward signal and applying PPO [27] to update learner policy π does not consider gradients from $\log \pi(s)$ as a direct

supervision. Therefore, we cannot only use policy gradient for policy optimization.

To deal with such a problem, we utilize ADMM [21] and treat the policy update in Eq.(3) as below constrained optimization problem, *i.e.*,

$$\max_{\pi,\pi'} \mathbb{E}_{(s,a)\in\mathcal{D}_E\cup\mathcal{D}_\pi}[\log\pi(a|s)] + \mathbb{E}_{(s,a)\in\mathcal{D}_E\cup\mathcal{D}_{\pi'}}[\log P(s)],$$
$$\text{s.t., } \pi = \pi' = z, \tag{4}$$

where z is a global variable coordinating the equality between local variables π and π'. Here the learner policy π tries to mimic expert behavior, and its counterpart policy π' tries to match expert state distribution using guidance from expert-learner state distribution P_ω. The learner policy π and its counterpart π' are expected to be the same for the optimality in Eq.(3) over π. Then, the augmented Lagrangian [29] of Eq.(3) is,

$$\mathcal{L}(\pi,\pi',\mu) = \mathbb{E}_{(s,a)\in\mathcal{D}_E\cup\mathcal{D}_\pi}[\log\pi(a|s)]$$
$$+ \mathbb{E}_{(s,a)\in\mathcal{D}_E\cup\mathcal{D}_{\pi'}}[\log P(s)] - \frac{\mu}{2}||\pi - z||_2^2 - \frac{\mu}{2}||\pi' - z||_2^2.$$

Based on the consensus optimization via ADMM, we have $z = \frac{\pi+\pi'}{2}$[3] where the learner policy π and its counterpart π' are encouraged to be the same via the averaging effect. Therefore, in each π training iteration, we use BC to update π, and follow the RL approach to employ PPO [27] for back-propagation over π'. The Adam optimizer [28] is used for both π and π'. The updated π and π' are averaged for the next ADMM training iteration as is shown in Alg. 2. Consequently, the learner policy averages information obtained via BC and expert-learner state distribution. The expert portion of information leads the learner policy towards expert behaviors.

Though there is no theoretical guarantee of the convergence of ADMM in optimizing Eq.(4), we have tried to regularize training to enforce convergence. In implementation, we included weight decay during training with ADMM which encourages the Hessian matrices of the $\mathbb{E}_{(s,a)\in\mathcal{D}_E\cup\mathcal{D}_\pi}[\log\pi(a|s)]$ and $\mathbb{E}_{(s,a)\in\mathcal{D}_E\cup\mathcal{D}_{\pi'}}[\log P(s)]$ functions to approach positive semidefinite. This likely entails approximately convex functions where ADMM has better convergence guarantee [30]. In other words, the implementation trick roughly ensures the three assumptions, *i.e.*, bounded Hessian matrix, subproblem solution equality and bounded function value (page 5 Assumption A by [30]) in a global consensus problem, and has a promise in convergence. Empirical results in Sec. IV-A validates the good convergence property in DC.

IV. Experiment

To evaluate our proposed DC methodology, we conduct experiments on ten physics-based control tasks, including CartPole [31], Reacher (with 1, 2 and 4 targets), Hopper, Walker, HalfCheetah (with 1 and 2 running directions), Ant, and Humanoid all simulated with MuJoCo [20]. From these experiments, we show that: i) DC performance improves with increasing P_ω update loop number whose ADMM-based learner update approach converges empirically, and DC is stable with different learning rates; ii) our DC learner policies avoid mode collapse and covariate shift, by accurately preserving the expert mode distribution; iii) DC learner policies have comparable or better performance than IL baselines.

[3]Here we use $\frac{\pi+\pi'}{2}$ to express the element-wise parameter averaging operation between two networks.

Algorithm 2 Policy Learning via ADMM

Require: Parameters θ_i for the learner policy π_θ in loop i; Parameters ω_{i+1} for the state distribution P_ω; Expert demonstrations $\mathcal{T}_E$ containing sequences of state-action pairs; total number of learner policy updates N.

Ensure: The learned policy $\pi_{\theta_{i+1}}$.

1: Assign initial π_θ parameter as $\theta_{i,0} = \theta_i$.
2: **for** each iteration $n = 0, \cdots, N$ **do**
3: Copy $\pi_{\theta_{i,n}}$ to get its counterpart $\pi_{\theta'_{i,n}}$.
4: Sample trajectories with $\pi_{\theta'_{i,n}}$ and store in $\mathcal{T}_{\pi_{\theta'_{i,n}}}$.
5: Sample state-action pairs $\mathcal{D}_E$ and $\mathcal{D}_{\pi_{\theta'_{i,n}}}$ with the same batch size.
6: Update $\pi_{\theta_{i,n}}$ to $\pi_{\theta_{i,n+1}}$ by ascending with gradients: $\Delta_{\theta_{i,n}} = \mathbb{E}_{(s,a)\in\mathcal{D}_E\cup\mathcal{D}_{\pi_{\theta_{i,n}}}}[\nabla_{\theta_{i,n}}\log\pi_{\theta_{i,n}}(a|s)]$.
7: Take a policy step from $\pi_{\theta'_{i,n}}$ to $\pi_{\theta'_{i,n+1}}$, using PPO update rule to increase the objective: $\mathbb{E}_{(s,a)\in\mathcal{D}_E\cup\mathcal{D}_{\pi_{\theta'_{i,n}}}}[\log P_{\omega_{i+1}}(s)]$.
8: Assign π_θ as $\pi_{\theta_{i,n+1}} = \frac{\pi_{\theta_{i,n+1}}+\pi_{\theta'_{i,n+1}}}{2}$.
9: **end for**

Implementation Settings and IL Baselines. We obtain expert policies of all tasks by running TRPO [32] with their ground-truth reward functions defined in the OpenAI Gym [33]. Then, we use the expert policies to generate expert demonstrations. We use Reacher with two and four target modes, and HalfCheetah with two target modes [20] respectively (*i.e.*, *Reacher2*, *Reacher4* and *HalfCheetah2*) to analyze the mode coverage of our DC vs IL baselines. Each expert policy obtained by TRPO has a particular distribution of reaching different target modes. We relegate more implementation details to Appx. A. Below are the five IL baselines we use to compare with DC:

- *Behavior Cloning (BC)* [4]: Expert demonstrations as a set of state-action pairs are split into 70% training data and 30% validation data. The learner policy is trained with supervised learning where actions are viewed as labels and states as input features.
- *Generative Adversarial Imitation Learning (GAIL)* [3]: GAIL is an IL method that consists of a generator as a policy network mimicking the expert behaviors, and a discriminator as a reward signal distinguishing between learner and expert behaviors.
- *Generative PRedecessor models for Imitation Learning (GPRIL)* [34]: GPRIL performs the state-action distribution matching by jointly training the learner policy and the corresponding multi-step predecessor state-action distribution.

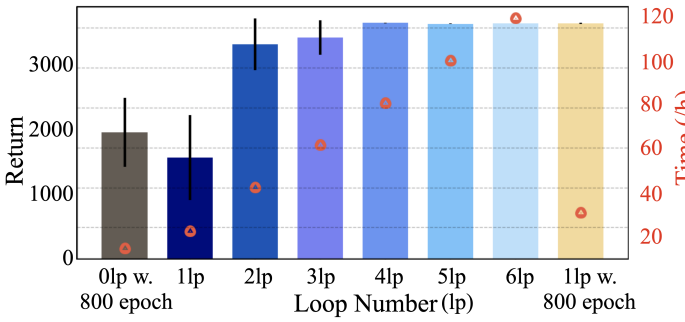


Fig. 3: Performance and time with different loop numbers in Hopper. Black bars show reward std.

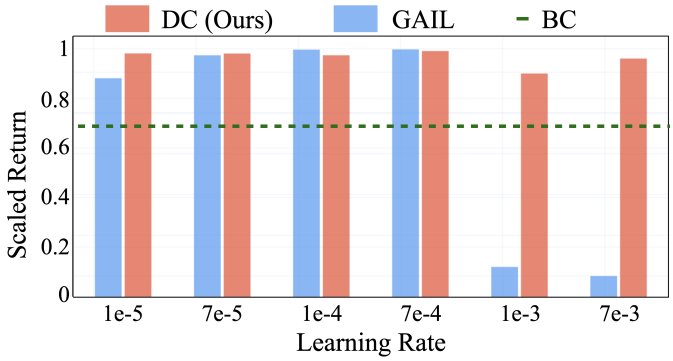


Fig. 4: Policy performances with different learning rates using GAIL and our DC in Hopper.

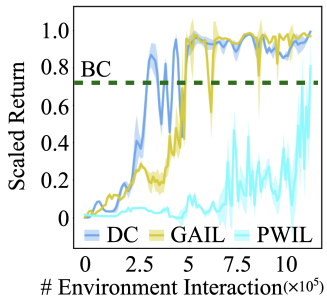


Fig. 5: Performance over environment interactions in Hopper.

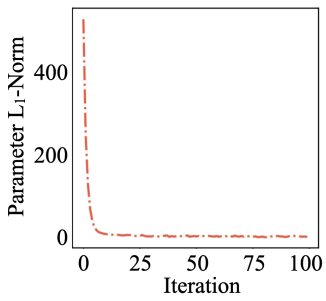


Fig. 6: π_θ parameter change in L_1-norm over ADMM iterations.

In each iteration, the predecessor state-action distribution is estimated using masked autoregressive flows [35].

- *Disagreement-Regularised Imitation Learning (DRIL)* [36]: DRIL pre-trains an ensemble of BC policies with expert demonstration data, and uses RL to train a learner policy whose cost function is proportional to the sum of the variance of ensemble policies' predictions.
- *Primal Wasserstein IL (PWIL)* [37]: PWIL ties the primal form of Wasserstein distance with IL to match learner's and expert's state-action distributions. It derives an upper-bound of the Wasserstein distance via greedy coupling as the offline reward for policy learning.
- *Neural Density Imitation (NDI)* [38]: NDI learns expert's occupancy measure as a reward and applies maximum occupancy entropy RL to train a learner policy. It maximizes a lower bound RKL divergence [23] between occupancy measures of the expert and learner.

A. Ablation Studies

In this section, we explore how the training design, such as *the number of P_ω update loops* and *learning rate changes* affect the model performance. We also investigate the convergence property of the learner policy π_θ update via ADMM in terms of *the model parameter changes* and *the training curve*. Below, we show our results from the Hopper task with 11 expert trajectories; similar observations were made for other tasks.

Impact of the P_ω update loop number. Figure 3 shows the performance and time usage of DC with different loop numbers of P_ω update where each loop contains 100 iterations of learner policy π_θ update with ADMM [21]. It shows that P_ω guidance is necessary for improving learner policy performance, and with increasing loop number, the performance of DC increases and converges at the 4th loop for the Hopper task. The consumed time is roughly linear to the loop number. We also compare DC when trained with one loop where the learner policy π_θ is updated for 1,000 iterations. It shows that increasing learner policy update number is able to attain comparable performance with that trained with multiple loops while decreasing the computation time significantly. Therefore, training the DC model with one loop for multiple learner policy π_θ update iterations via ADMM serves as a practical choice to balance performance vs. computation cost. We employ the single P_ω update loop with 1,000 iterations of learner policy π_θ update implementation of DC for the following experiments.

Impact of the learning rate. We further study DC's robustness in choosing different learning rates. Figure 4 shows the performance of DC and GAIL given different learning rates. It shows that DC works in a wide range of learning rates, while GAIL tends to fail/crash when learning rates are $1e-3$ and higher. GAIL with adversarial training is less robust to the change of learning rate due to vanishing gradient and the complex interactions between the discriminator and the generator [7], [15]. However, our DC updates the learner policy π_θ via ADMM which provides better robustness to learning rate changes.

Convergence property. Figure 5 shows the performance change with the number of environment interactions (self-supervision steps). This figure shows that DC is able to attain a higher return than GAIL given a small number of environment interactions (at around 3–6$\times 10^5$ steps). An explanation is that the trained P_ω is good at guiding the learner policy π_θ to explore on those states frequently visited by the experts. In addition, the ADMM update for the learner policy π_θ

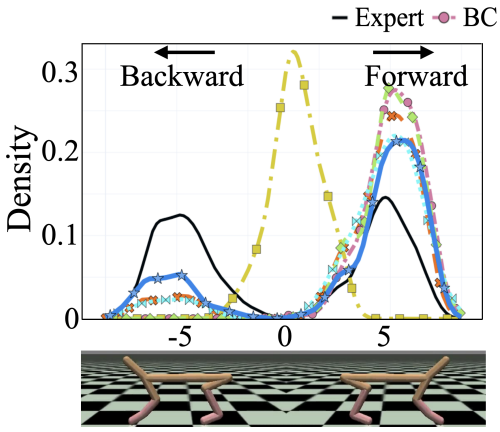


Fig. 7: Results of DC (Ours) and baselines on mode coverage in HalfCheetah2. All the distributions are visualized using KDE [1].

Fig. 8: EMD *vs*. scaled return in tasks with multiple modes. The x-axis is the EMD [39] between expert and learner policy state distribution. The y-axis is the expected return (*i.e*., total reward), scaled so that the expert achieves 1 and a random policy achieves 0.

demonstrates good model convergence property as is shown in Figure 6. Moreover, differing from GAIL with two generator and discriminator trained alternatively, DC avoids adversarial training and employs the iterative two-stage training process to training P_ω and π_θ separately, thus leading to fast and stable convergence.

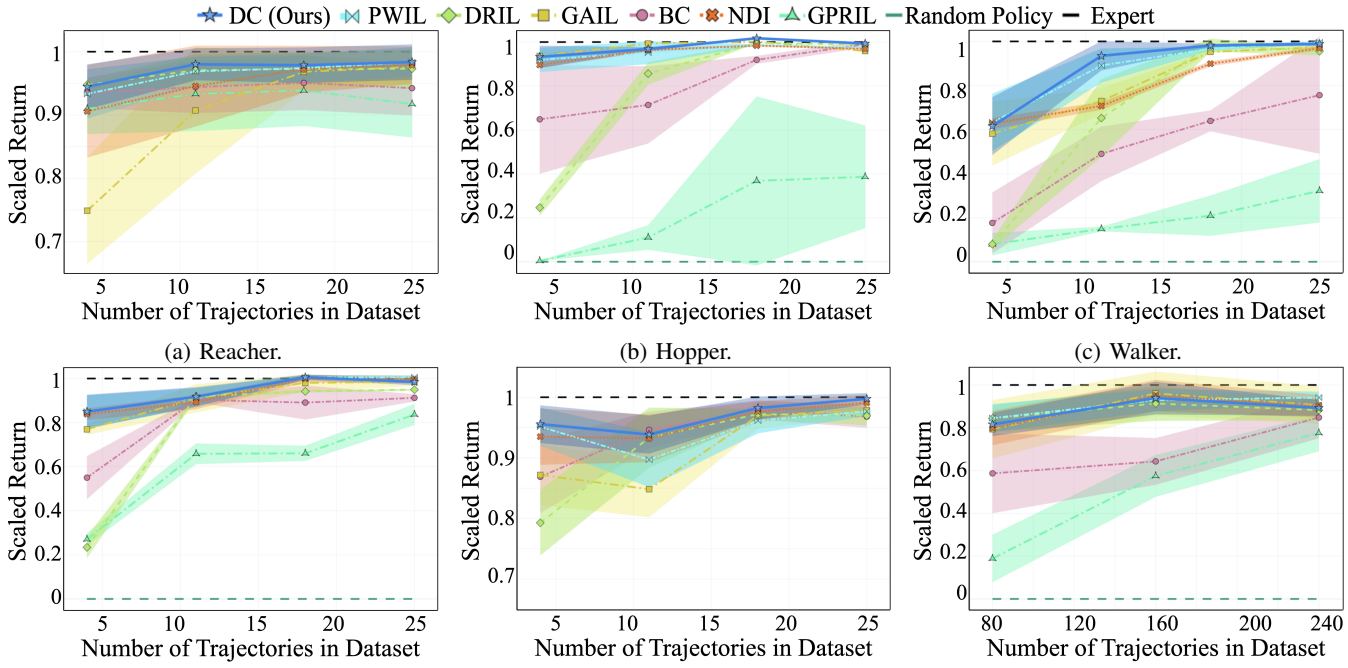
B. Mode Coverage

Figure 1 shows the results of Reacher4 with four mode targets in different colors in Figure 1g, and Figure 7 shows the results of HalfCheetah2 with two mode directions as running forward and backward. Results of Reacher2 are in Appx. B as its observations are similar to the Reacher4 task. Figure 1h shows the mode coverage (*i.e*., the state distribution) of the expert policy, where the expert tends to cover all four targets (as four modes) evenly. The learner policy obtained by DC preserves the expert mode coverage very well as shown in the 3rd figure in Figure 1f. On the contrary, BC learned policy tends to visit the blue and red targets more as BC only matches expert action distribution and is vulnerable to covariate shift. The learner policy from GAIL is prone to mode collapse. It only focuses on the green target out of the four. Consistent with results in SOTA works [7], [8], [15], this mode collapse is due to the adversarial training process in GAIL. DRIL, PWIL and NDI also fail to cover the green target mode, the red one and the yellow one as their defined reward function likely encourages a mode seeking behavior. DC successfully preserves the mode coverage from the expert, because it uses SoftFlow [24] to accurately estimate the expert state distribution, and the stabilized two-stage training to update the learner policy.

In the HalfCheetah2 task in Figure 7, the x-axis represents the running velocity, and the plots show the velocity distributions of expert and IL policies. The black curve demonstrates two modes (*i.e*., running forward and backward) in expert demonstrations. DC (blue curve) is able to preserve all modes, while DRIL and BC are collapsed to running forward, GAIL fails to reveal any mode, and PWIL and NDI concentrate more on the forward mode. We further calculated the earth mover's distance (EMD) [39] between expert and learned policies' state distributions as the x-axis in Figure 8 in the Reacher and HalfCheetah tasks with multiple modes. A lower EMD value indicates a better learner policy at recovering expert demonstration modes, and the results echo the above observations quantitatively. The EMD, KL and RKL divergences [23] between expert and learned policy state distribution are listed in Table I. In all tasks, DC obtains the lowest EMD, KL and RKL scores indicating its superior ability to recover expert behaviors on tasks with multiple modes. We omit GPRIL as no meaningful results are obtained.

Task	Approach	Measure		
		EMD	KL	RKL
Reacher2	BC	1.01	2.60	4.38
	GAIL	0.84	2.47	4.51
	DRIL	1.00	2.51	4.25
	PWIL	0.67	2.47	3.46
	NDI	0.68	2.49	3.54
	DC	**0.58**	**2.47**	**3.35**
Reacher4	BC	0.81	4.23	6.33
	GAIL	0.41	4.49	6.22
	DRIL	0.55	5.16	6.16
	PWIL	0.52	5.13	5.95
	NDI	0.54	5.15	6.02
	DC	**0.40**	**4.04**	**5.77**
HalfCheetah2	BC	1.81	6.74	12.78
	GAIL	1.75	4.52	18.13
	DRIL	1.83	6.90	12.76
	PWIL	1.56	4.23	12.56
	NDI	1.60	4.26	12.63
	DC	**1.52**	**4.06**	**12.43**

TABLE I: The EMD [39], KL and RKL between expert and learned policy state distribution.



(a) Reacher. (b) Hopper. (c) Walker.

(d) HalfCheetah. (e) Ant. (f) Humanoid.

Fig. 9: Performance of learner policies in tasks with one mode. The y-axis is the expected return (*i.e.*, total reward), scaled so that the expert achieves 1 and a random policy achieves 0.

C. Performance of the Learner Policy from DC

Figure 8 shows the EMD [39] vs scaled return results in Reacher and HalfCheetah with different numbers of demonstration modes. In all the tasks, DC has both lower EMD and higher scaled return compared with baselines. This is because SoftFlow is able to recover expert state distribution P_E with multiple modes well, and thus provides useful feedback for the learner policy to recover expert behaviors. Comparing between Reacher with 2 and 4 targets, DC shows a larger return margin when the target modes are more distant from each other in Reacher2.

Figure 9 shows the performances of the learner policies from our DC and IL baselines under different numbers of expert trajectories when they only contain one mode. In all tasks, our DC learner policies have comparable performances with GAIL, which is because expert demonstrations are able to push learner policies towards expert behaviors without using adversarial training. DC outperforms DRIL particularly with a limited number of expert trajectories because the use of a SoftFlow learned P_E has good generalization ability and provides useful feedback to learner policy π for a better policy. Moreover, in both easy tasks (Reacher) and complex tasks (Hopper and Walker), DC consistently outperforms BC with different numbers of expert demonstrations, which is because DC uses the state distribution matching on top of the BC objective to overcome the covariate shift problem. The performance of the BC learner policies increase with more expert demonstrations, as more training data mitigates the overfitting problem and compounding errors. However, the GPRIL learner policies have the lowest performances in all tasks. This is primarily because the policy is jointly learned with its multi-step predecessor state-action distribution. With random initial parameters for these two functions, it is hard to progressively improve them jointly.

V. Related Work

Imitation Learning (IL) has two solution paradigms, *i.e.*, Behavior Cloning (BC) [4], [22] and Generative Adversarial Imitation Learning (GAIL) [3]. BC learns the expert policy via maximizing expert demonstration likelihood. It suffers from covariate shift with limited demonstration [5], [6]. [36], [40] aim to address this problem with learner-environment interactions during training. GAIL [3] employs GAN to minimize the JS divergence between expert and learner state-action distributions. Using the variational lower bound of an f-divergence, several studies [5], [11], [12], [14], [41], [42] extended GAIL from JS to any f-divergence. However, these methods apply adversarial training, which leads to training instability and mode collapse [8], [13].

Several recent works try to avoid adversarial training in IL [34], [37], [38], [43], [44]. For this, Primal Wasserstein IL (PWIL) [37] considers the primal form of Wasserstein distance to match learner's and expert's behaviors. Neural Density Imitation (NDI) [38] estimates expert's occupancy measure with which as a reward for reinforcement learning. Energy-Based IL (EBIL) [43] stems from Max-Entropy IRL [45] and estimates a surrogate reward function with score matching from expert demonstrations. Imitative Models (IM) [44] learns a flow model that assigns high likelihoods to expert-like

trajectories for test time goal-directed planning. GPRIL [34] applies masked autoregressive flows [35] to learn predecessor state-action distribution in each training iteration, adding complexity to learner policy update and burdening training. Unlike these methods, we maximize expert and learner state-action likelihood to avoid adversarial training and solve the mode collapse problem simultaneously.

Normalizing Flow [46]–[50] is a generative model paradigm that features data transformations between random noises (*e.g.*, in Gaussian distribution) and training data samples. It explicitly learns a data distribution based on the (instantaneous) change of variables formula and is trained via maximum likelihood estimation [46], [49]. However, these models are vulnerable to the manifold hypothesis and cannot deal with data dimension mismatch. Consequently, they are not suitable for estimating the distribution of the data on a lower-dimensional manifold [24]. SoftFlow [24] tackles these problems via estimating a conditional distribution of the perturbed data samples to better capture the innate structure of the manifold data. Therefore, we apply SoftFlow in this work.

Alternating Direction Method of Multipliers (ADMM) [21] solves an optimization problem via breaking them into multiple subproblems. It coordinates the subproblems globally to arrive at a solution. ADMM can serve as an alternative to stochastic gradient descent as it allows parallel data processing and avoids gradient vanishing [21], [51]. It has been applied in RL [21], [51]–[53]. Under the guided policy search (GPS) framework of RL, GPS with Bregman ADMM (GPS-BADMM) [52] applies a variant of ADMM to decompose the GPS objective into subproblems optimized over a trajectory distribution and a learner policy. [53] solves the GPS problem from a supervised learning perspective and introduces an adversarial regularization to improve learner policy robustness and generalizability. It decomposes the problem via ADMM as trajectory optimization and policy learning. Unlike these works, we employ ADMM on IL to help coordinate information from expert supervision and learner exploration.

VI. Conclusion

In this work, we proposed DC – distributional cloning – which trains a learner policy that employs expert averaging effect by maximizing the likelihood of expert and learner demonstrations. DC learned policies approach the expert behavior gradually during training. Each training loop of DC contains two stages, *i.e.*, *Stage #1* estimates the expert-learner state distribution using SoftFlow, and *Stage #2* trains the learner policy to match both expert's policy and state distribution via ADMM. Comparing our DC with baselines in ten different physics-based control tasks, we present superior evaluation results in learner policy performance, training stability, and mode distribution preservation.

VII. Acknowledgement

Yanhua Li was supported in part by NSF grants IIS-1942680 (CAREER), CNS-1952085, and DGE-2021871. Ziming Zhang was supported by NSF grant CCF-2006738. C. Brinton acknowledges support from NSF grants CNS-2146171, CPS-2313109 and ONR grant N000142212305. Zhenming Liu was supported by NSF grant IIS 2008557. Zhi-Li Zhang was supported in part by NSF grants CNS-1901103, CCF-221231, and CNS-222029.

References

[1] S. J. Sheather and M. C. Jones, "A reliable data-based bandwidth selection method for kernel density estimation," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 53, no. 3, pp. 683–690, 1991.

[2] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, p. 1, ACM, 2004.

[3] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, pp. 4565–4573, 2016.

[4] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Computation*, vol. 3, no. 1, pp. 88–97, 1991.

[5] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, 2011.

[6] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 661–668, 2010.

[7] R. Jena and K. Sycara, "Loss-annealed gail for sample efficient and stable imitation learning," *arXiv preprint arXiv:2001.07798*, 2020.

[8] C. Fei, B. Wang, Y. Zhuang, Z. Zhang, J. Hao, H. Zhang, X. Ji, and W. Liu, "Triple-gail: a multi-modal imitation learning framework with generative adversarial nets," *arXiv preprint arXiv:2005.10622*, 2020.

[9] Y. Li, J. Song, and S. Ermon, "Infogail: Interpretable imitation learning from visual demonstrations," in *Advances in Neural Information Processing Systems*, pp. 3812–3822, 2017.

[10] K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. Lim, "Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets," *arXiv preprint arXiv:1705.10479*, 2017.

[11] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," *arXiv preprint arXiv:1710.11248*, 2017.

[12] L. Ke, M. Barnes, W. Sun, G. Lee, S. Choudhury, and S. Srinivasa, "Imitation learning as f-divergence minimization," *arXiv preprint arXiv:1905.12888*, 2019.

[13] S. Ghasemipour, R. Zemel, and S. Gu, "A divergence minimization perspective on imitation learning methods," *arXiv preprint arXiv:1911.02256*, 2019.

[14] X. Zhang, Y. Li, Z. Zhang, and Z.-L. Zhang, "f-gail: Learning f-divergence for generative adversarial imitation learning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[15] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[16] T. Franzmeyer, P. Torr, and J. F. Henriques, "Learn what matters: cross-domain imitation learning with task-relevant embeddings," *Advances in Neural Information Processing Systems*, vol. 35, pp. 26283–26294, 2022.

[17] J. Liu, Q. Wang, and Y. Xu, "Ar-gail: Adaptive routing protocol for fanets using generative adversarial imitation learning," *Computer Networks*, vol. 218, p. 109382, 2022.

[18] J. Lacotte, M. Ghavamzadeh, Y. Chow, and M. Pavone, "Risk-sensitive generative adversarial imitation learning," in *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2154–2163, PMLR, 2019.

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[20] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.

[21] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, 2011.

[22] J. Bohg, M. Pavone, and D. Sadigh, "Principles of robot autonomy ii," *URL https://web.stanford.edu/class/cs237b/pdfs/lecture/lecture_10111213.pdfimitationlearning.pdf.[Online]*, 2020.
[23] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, 1991.
[24] H. Kim, H. Lee, W. H. Kang, J. Y. Lee, and N. S. Kim, "Softflow: Probabilistic framework for normalizing flow on manifolds," *Advances in Neural Information Processing Systems*, vol. 33, 2020.
[25] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
[26] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.
[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[29] M. Fortin and R. Glowinski, *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*. Elsevier, 2000.
[30] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 337–364, 2016.
[31] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 834–846, 1983.
[32] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, 2015.
[33] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.
[34] Y. Schroecker, M. Vecerik, and J. Scholz, "Generative predecessor models for sample-efficient imitation learning," *arXiv preprint arXiv:1904.01139*, 2019.
[35] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," *arXiv preprint arXiv:1705.07057*, 2017.
[36] K. Brantley, W. Sun, and M. Henaff, "Disagreement-regularized imitation learning," in *International Conference on Learning Representations*, 2019.
[37] R. Dadashi, L. Hussenot, M. Geist, and O. Pietquin, "Primal wasserstein imitation learning," *arXiv preprint arXiv:2006.04678*, 2020.
[38] K. Kim, A. Jindal, Y. Song, J. Song, Y. Sui, and S. Ermon, "Imitation with neural density models," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5360–5372, 2021.
[39] H. Ling and K. Okada, "An efficient earth mover's distance algorithm for robust histogram comparison," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 5, pp. 840–853, 2007.
[40] S. Reddy, A. D. Dragan, and S. Levine, "Sqil: Imitation learning via reinforcement learning with sparse rewards," *arXiv preprint arXiv:1905.11108*, 2019.
[41] S. Nowozin, B. Cseke, and R. Tomioka, "f-gan: Training generative neural samplers using variational divergence minimization," in *Advances in neural information processing systems*, pp. 271–279, 2016.
[42] D. Arumugam, D. Dey, A. Agarwal, A. Celikyilmaz, E. Nouri, E. Horvitz, and B. Dolan, "Reparameterized variational divergence minimization for stable imitation," 2019.
[43] M. Liu, T. He, M. Xu, and W. Zhang, "Energy-based imitation learning," *arXiv preprint arXiv:2004.09395*, 2020.
[44] N. Rhinehart, R. McAllister, and S. Levine, "Deep imitative models for flexible inference, planning, and control," *arXiv preprint arXiv:1810.06544*, 2018.
[45] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning.," in *AAAI*, vol. 8, pp. 1433–1438, Chicago, IL, USA, 2008.
[46] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International Conference on Machine Learning*, pp. 1530–1538, PMLR, 2015.
[47] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

Task	Training iterations	Expert performance	Random policy performance
CartPole-v0	200	200±0	17± 4
Reacher-v2	200	-4.5±1.7	-93.7 ±4.8
Hopper-v2	1000	3624±19	8± 6
Walker-v2	1000	7002±33	-2±3
Ant-v2	1000	4838± 231	8 ± 124
HalfCheetah-v2	1000	4501±111	101±80
Humanoid-v2	1500	10400±89	101±35

TABLE II: Environment setup.

[48] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *arXiv preprint arXiv:1410.8516*, 2014.
[49] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *arXiv preprint arXiv:1806.07366*, 2018.
[50] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "Ffjord: Free-form continuous dynamics for scalable reversible generative models," *arXiv preprint arXiv:1810.01367*, 2018.
[51] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, "Training neural networks without gradients: A scalable admm approach," in *International conference on machine learning*, pp. 2722–2731, PMLR, 2016.
[52] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.
[53] Z. Zhao, S. Zuo, T. Zhao, and Y. Zhao, "Adversarially regularized policy learning guided by trajectory optimization," *arXiv preprint arXiv:2109.07627*, 2021.

APPENDIX

A. EXPERIMENT SETUPS

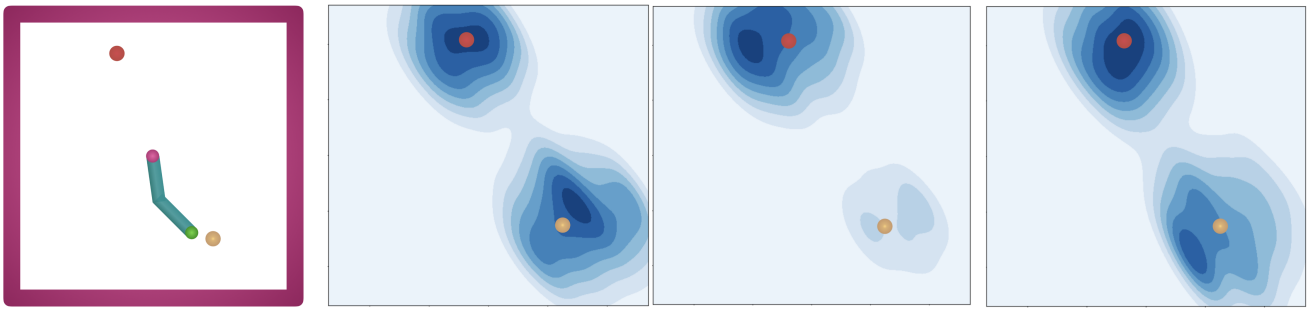
Resource usage. All experiments are run on GeForce RTX2080. P_ω training takes 6 hours for CartPole and Reacher, and 12 hours for Hopper and Walker for 200 iterations.

Evaluation setup. We use the same amount of environment interactions and expert demonstrations in GAIL, GPRIL, DRIL, PWIL, NDI and DC , where information of each task is shown in Table II. For all approaches, we evaluate their learner policy performances in every iteration. The experiments with multiple mode task settings all use 18 expert trajectories. The evaluation score is achieved via evaluating the mean and std of 50 trajectories generated with the learner policy.

Hyperparameter details. For training P_ω, we set a noise level ranges from 0 to 0.1. The learning rate is $1e-4$ for all tasks. The training of π also has a learning rate at $1e-4$ with gradient clip 0.1.

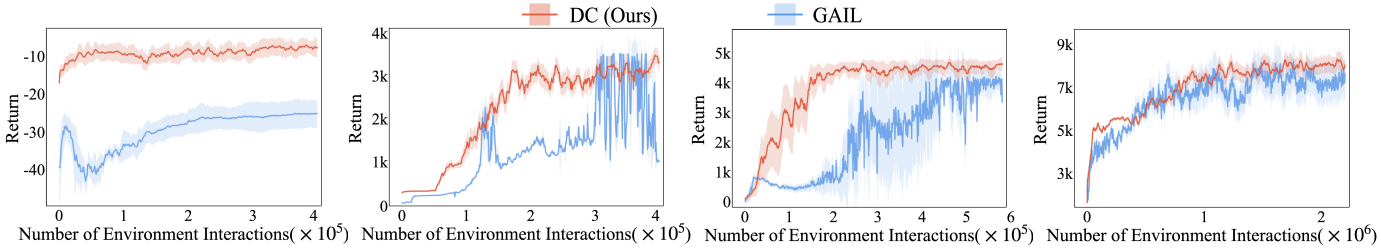
B. MORE EXPERIMENT RESULTS

Figure 10 shows the mode coverage results of DC (ours) and baselines over the Reacher2 task with two target modes. It shows two mode targets from expert demonstrations in Figure 10b. In this task, GAIL only visited the red target and DC visited both targets as are shown in Figure 10c and Figure 10d respectively. Table III shows detailed results of DC and baselines in tasks with single expert demonstration mode. Figure 11 shows the learning curves of the test tasks with a single mode when demonstration number is 4. In all tasks, DC has more stable training curves (with less mean return perturbation) with higher convergence speed.



(a) Reacher2 with two mode targets. (b) Mode coverage of expert policy. (c) Mode coverage of GAIL policy. (d) Mode coverage of our DC policy.

Fig. 10: Results obtained by DC (Ours) and baselines on mode coverage. (a): A Reacher task, with two targets in different colors. (b)-(d) show the mode coverage (*i.e.*, state distribution) with expert policy (b), GAIL policy (c), and our DC policy (d). All the distributions are visualized using KDE [1]. A darker color indicates a more densely distributed region in the state space.



(a) Reacher. (b) Hopper. (c) Ant. (d) Humanoid.

Fig. 11: Learning curve comparision between GAIL and DC (Ours). All tasks are shown 4 demonstrations. The y-axis is the obtained return (*i.e.*, total reward).

TABLE III: Learned policy performance.

Task	Datasize	BC	GAIL	GPRIL	DRIL	PWIL	NDI	DC (Ours)
CartPole	1	59±27	200±0	53±16	200±0	200±0	200±0	**200±0**
	4	81±31	200±0	187±8	200±0	200±0	200±0	**200±0**
	7	137 ±27	200 ±0	200 ±0	200±0	200±0	200±0	**200±0**
	10	167± 30	200 ±0	200 ±0	200±0	200±0	200±0	**200±0**
Reacher	4	-10.27±2.14	-26.90±7.48	-12.55±3.54	**-9.13±2.83**	-10.44±3.46	-12.90±6.51	-9.32 ±3.00
	11	-9.49±3.66	-12.77±8.90	-10.45±5.21	-7.11±2.23	-7.26 ± 2.06	-9.34±5.70	**-6.23± 3.05**
	18	-8.89±3.83	-7.34±2.63	-9.96±5.01	-6.93±2.37	-6.81±2.02	-6.97±3.30	**-6.21±2.49**
	25	-9.63±3.84	-6.64±2.47	-11.87±4.71	-6.90±2.65	-5.98 ± 2.10	-6.40±2.66	**-5.82± 2.43**
Hopper	4	2352± 894	**3394±37**	22± 1	898±132	3331±200	3248±53	3378± 164
	11	2589± 635	**3599± 4**	407± 202	3150±184	3452±189	3491±24	3510± 34
	18	3331± 66	3631± 3	1339± 1390	3611±3	3577±12	3565±17	**3686± 8**
	25	3589± 56	3476± 5	1406± 844	3580±8	3530± 9	3512±9	**3599± 8**
Walker2d	4	1233± 969	4070 ±1010	557 ±357	555±148	**4430±908**	4393±177	4312± 918
	11	3456± 863	5108± 410	1042± 75	4567±1231	6237±540	4953±121	**6545± 461**
	18	4477± 1329	6671± 39	1464± 637	**6886±202**	6773±65	6294±78	6869± 65
	25	5294± 1860	6815± 20	2254± 1006	6693±130	**6973±45**	6786±55	6920± 58
Ant	4	4204±289	4218±240	2722±36	3837±259	4602±151	4523±224	**4623±150**
	11	**4577±145**	4105±223	2510±27	4515±239	4342±211	4500±189	4541±152
	18	4736±75	4690±102	2755±183	4703±40	4649±101	4713±83	**4738±82**
	25	4682±89	4735±54	2656±85	4690±75	4733±35	4753±64	**4805±45**
HalfCheetah	4	2070±528	3254±133	558±148	359±266	3688±400	3625±142	**3660±406**
	11	3979±61	4015±344	2655±253	**4063±50**	4021±233	3922±173	4043±221
	18	3911±416	4393±212	2666±186	4185±30	4520±53	4496±95	**4523±65**
	25	4027±91	4423±104	3619±257	4227±26	4532±44	4417±90	**4436±67**
Humanoid	80	6145±1918	8268±1401	2048±1140	**8789±639**	8810±657	8321±822	8504±562
	160	6722±1126	**9994±1053**	6023±1006	9507±832	9677±535	9892±753	9774±785
	240	8834±998	9430±906	8091±878	9185±492	**9798±281**	9381±539	9294±385