A Coupled Flow Approach to Imitation Learning

Gideon Freund [1] **Elad Sarafian** [1] **Sarit Kraus** [1]

Abstract

In reinforcement learning and imitation learning, an object of central importance is the state distribution induced by the policy. It plays a crucial role in the policy gradient theorem, and references to it—along with the related state-action distribution—can be found all across the literature. Despite its importance, the state distribution is mostly discussed indirectly and theoretically, rather than being modeled explicitly. The reason being an absence of appropriate density estimation tools. In this work, we investigate applications of a normalizing flow-based model for the aforementioned distributions. In particular, we use a pair of flows coupled through the optimality point of the Donsker-Varadhan representation of the Kullback–Leibler (KL) divergence, for distribution matching based imitation learning. Our algorithm, Coupled Flow Imitation Learning (CFIL), achieves state-of-the-art performance on benchmark tasks with a single expert trajectory and extends naturally to a variety of other settings, including the subsampled and state-only regimes.

1. Introduction

Reinforcement learning (RL) (Sutton & Barto, 2018) concerns the optimization of an agent's behavior in an environment. Its characterizing difficulties of exploration vs exploitation and credit assignment, stem from its typical incarnations where the agent must learn from sparse feedback. In order to provoke desired behavior, one may need to craft a sophisticated reward function or provide demonstrations for the agent to imitate. Imitation Learning (IL) deals precisely with the latter: learning from expert demonstrations. Although RL and IL share the same ultimate goal of producing a good policy, they differ fundamentally in that RL is guided by the environment's feedback, while IL is guided only by the ability of the agent to reproduce the expert's behavior.

Central to both RL and IL are the state and state-action distributions induced by the policy. Their importance cannot be overstated, with the state distribution forming the basis of policy gradient methods through the policy gradient theorem (Sutton et al., 2000), and the state-action distribution being core to the common distribution matching formulation of IL (Ke et al., 2020; Ghasemipour et al., 2020). They are also foundational to other applications, like curiosity-based exploration (Pathak et al., 2017), constrained RL (Qin et al., 2021) and Batch RL (Fujimoto et al., 2019), some of which have recently been unified under the umbrella of convex RL (Zhang et al., 2020; Zahavy et al., 2021; Mutti et al., 2022), which studies objectives that are convex functions of the state distribution.

Despite their ubiquity across the literature, explicit modeling of the distributions is scarce. Instead, they mostly find use as a theoretical tool for derivations. This is, of course, barring some approaches that do attempt to model them (Hazan et al., 2019; Qin et al., 2021; Lee et al., 2019; Kim et al., 2021b) or their ratios (Nachum et al., 2019; Liu et al., 2018; Gangwani et al., 2020), further discussion of which is delegated to the related work. The reason for this lack of modeling is due to the difficulty of density estimation, especially in the case of complex and high-dimensional distributions. This is where normalizing flows (NF) (Dinh et al., 2014; 2016; Papamakarios et al., 2017), a recent approach to density estimation, will be of service.

We believe there is no shortage of applications for such modeling, but we focus on imitation learning, a quite natural and suitable place, given its modern formulation as state-action distribution matching (Ghasemipour et al., 2020).

Many approaches to distribution matching based imitation have been presented (Ho & Ermon, 2016; Fu et al., 2017; Kostrikov et al., 2018; Ke et al., 2020; Ghasemipour et al., 2020; Kostrikov et al., 2019; Sun et al., 2021; Kim et al., 2021b; Dadashi et al., 2020; Schroecker, 2020). The common theme for such methods begins with the selection of a divergence, followed by the development of a unique approach. This may involve a direct attack on the objective by reformulating it (Kostrikov et al., 2019), or by derivation of a surrogate objective (Zhu et al., 2020; Dadashi et al.,

[1]Department of Computer Science, Bar-Ilan University, Israel. Correspondence to: Gideon Freund <gideonfreund@gmail.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

2020; Kim et al., 2021b), with some utilizing mechanisms such as an inverse action model (Zhu et al., 2020) and focusing on learning from states alone (a setting our approach naturally lends to). Other methods first derive estimates for the gradient of the state distribution with respect to the policy's parameters (Schroecker, 2020), while some devise unifying algorithms and frameworks encompassing previous approaches (Ke et al., 2020; Ghasemipour et al., 2020).

The most popular divergence of choice is the reverse KL, which some favor due to its mode-seeking behavior (Ghasemipour et al., 2020). Others attempt to get the best of both worlds, combining both mode-seeking and mode-covering elements (Zhu et al., 2020). A priori, it is difficult to say which choice of divergence is advantageous, it's more about the ensuing approach to its minimization.

In this work, we propose a unique approach to distribution matching based imitation, by coupling a pair of flows through the optimality point of the Donsker-Varadhan (Donsker & Varadhan, 1976) representation of the KL. More specifically, by noting this point occurs at the log distribution ratio, while the IL objective with the reverse KL can be seen as an RL problem with the ratio inverted. We propose setting the point of optimality as the difference of two normalizing flows, then training in an alternating fashion akin to other adversarial IL methods (Ho & Ermon, 2016; Kostrikov et al., 2018; Ghasemipour et al., 2020; Kostrikov et al., 2019; Sun et al., 2021). This method proves far more accurate than estimating the log distribution ratio by naively training a pair of flows independently. We show this in part by analyzing their respective BC graphs: a simple tool we present for gauging how well a proposed estimator captures the expert's behavior. While most IL works neglect analysis of their learned reward function, we think this can be a potential guiding tool for future IL researchers.

Our resulting algorithm, Coupled Flow Imitation Learning (CFIL) shows strong performance on standard benchmark tasks, while extending naturally to the subsampled and state-only regimes. In the state-only regime in particular, CFIL exhibits significant advantage over prior state-of-the-art work, despite the competition being specifically designed for that domain. This work also aims to inspire more research incorporating explicit modeling of the state-action distribution.

2. Background

2.1. Markov Decision Process

Environments in RL are typically expressed as a Markov Decision process (MDP). An MDP is a tuple $(S, A, P, R, p_0, \gamma)$, where S is a set of states termed the state space, A is a set of actions termed the action space, $P : S \times A \times S \to [0, \infty]$ is a transition density function describing the environment's Markovian dynamics, $R : S \times A \times S \to \mathbb{R}$ is a reward function, p_0 is an initial state distribution, and $\gamma \in [0, 1)$ is a discount factor.

A policy $\pi : S \to A$ dictates an agent's actions when roaming the environment, and the goal in an MDP is to find the optimal one, denoted π^*. The standard optimality criterion for a policy is the expected discounted reward: $J(\pi, r) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$ where $\tau \sim \pi$ symbolizes the trajectory distribution induced by the policy π, and r_t is the reward at time t along a trajectory. Each policy has an associated value function and Q-function: $V^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s\right]$ and $Q^\pi(s, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a\right]$, where the value function represents the expected discounted reward of following π from state s, while the Q-function represents the expected discounted reward of following π after taking action a from state s.

Another object induced by the policy is the improper discounted state distribution $d^\gamma_\pi(s) = \sum_{t=0}^{\infty} \gamma^t Pr(s_t = s | s_0 \sim p_0)$, as well as its undiscounted counterpart, $d_\pi(s) = \sum_{t=0}^{\infty} Pr(s_t = s | s_0 \sim p_0)$, in which we take greater interest for reasons described in Appendix C. $d_\pi(s)$ is called the state distribution and is of central interest to this work. Closely related to $d_\pi(s)$ is the state-action distribution $p_\pi(s, a) = \pi(a|s)d_\pi(s)$, which is also of interest to this work.

2.2. Normalizing Flows

Normalizing Flows (NF) (Dinh et al., 2014; 2016) are exact-likelihood generative models that use a simple base distribution $p_Z(z)$ and a sequence of invertible and differentiable transformations $f_\theta = f_k \circ f_{k-1} \circ \cdots \circ f_1$ to model an arbitrary target distribution $p_X(x)$ as $z = f_\theta(x)$. [1] This means the target log-likelihood can be written using the change of variables formula:

$$\log p_X(x) = \log p_Z(z) + \sum_{i=1}^{k} \log \left| \det \frac{df_i}{df_{i-1}} \right| \quad (1)$$

allowing for parameters θ to be trained using maximum likelihood. With a trained flow at hand, generation is performed by sampling from the base distribution and applying f_θ^{-1}, and density estimation can be done by evaluating the RHS of (1). RealNVP (Dinh et al., 2016), which is of particular interest to us, uses coupling layers (not to be confused with our coupled approach) to construct f_θ. Masked Autoregressive Flow (MAF) (Papamakarios et al., 2017) is a generalization of RealNVP which substitutes its coupling layers with autoregressive ones. While improving expressiveness, this impedes single pass sampling, but still allows efficient

[1] More often presented as $x = g(z)$ (Papamakarios et al., 2021; Kobyzev et al., 2020), which is equivalent with $f = g^{-1}$, due to their invertibility. Under this model, Equation (1) would look similar with only the "+" substituted by a "−".

density evaluation—our main concern—using masked autoencoders (Germain et al., 2015). RealNVP and MAF are reviewed more thoroughly in Appendix D.

Although much research (Kingma & Dhariwal, 2018; Huang et al., 2018; Durkan et al., 2019; Ho et al., 2019) has gone into improving the capacity of MAF, due to its simplicity, efficiency and adequacy for our task, it is the architecture used in this work to model the state and state-action distributions $d_\pi(s)$ and $p_\pi(s, a)$.

2.3. Imitation Learning

Imitation Learning (IL) concerns the optimization of an agent's behavior in an environment, given expert demonstrations. Perhaps the simplest approach to imitation, behavioral cloning (BC), performs supervised regression or maximum-likelihood on given expert state-action pairs $\{(s_e, a_e)\}_{t=1}^N$:

$$\min_\pi \sum_{t=0}^{N} Loss(\pi(s_t), a_t) \quad \text{or} \quad \max_\pi \sum_{t=0}^{N} \log \pi(a_t|s_t) \quad (2)$$

for deterministic and stochastic policies, respectively. BC suffers from compounding errors and distributional shift, wherein unfamiliar states cause the policy to misstep, leading to even less familiar states and an eventual complete departure from the expert trajectory (Ross et al., 2011).

Recent distribution matching (DM) approaches (Ho & Ermon, 2016; Kostrikov et al., 2018; 2019; Kim et al., 2021b) successfully overcome these issues. One common formulation (Ke et al., 2020; Ghasemipour et al., 2020) encompassing most of these methods, views DM as an attempt to match the agent's state-action distribution p_π, with the expert's p_e, by minimizing some f-divergence[2] D_f:

$$\arg\min_\pi \; D_f(p_\pi || p_e). \quad (3)$$

These methods hinge on the one-to-one relationship between a policy and its state-action distribution and have shown significant improvement over BC, particularly when few expert trajectories are available (Ghasemipour et al., 2020) or expert trajectories are subsampled (Li et al., 2022).

3. Related Work

Attempts at modeling the state distribution have been made for various applications and countless distribution matching approaches to imitation learning exist, with varying degrees of relevancy to our own. The most pertinent works—those intersecting both the former and the latter—are reviewed in most detail towards the end of this section.

[2]Other distances, such as the Wasserstein metric have also been used (Sun et al., 2021; Dadashi et al., 2020).

Some general attempts to model the state distribution include (Hazan et al., 2019) who use discretization and kernel density estimates (KDE) for curiosity-based exploration, while (Lee et al., 2019) uses variational autoencoders (VAE) in the same context. VAE's have also been used by (Islam et al., 2019) to model $d_\pi(s)$ for constraining the distribution shift in off-policy RL algorithms, and KDE's have also been used by (Qin et al., 2021) for density constrained reinforcement learning.

Approaches to distribution matching-based imitation include GAIL (Ho & Ermon, 2016) who uses a GAN-like (Goodfellow et al., 2014) objective to minimize the JS divergence. Originally born out of max-entropy inverse reinforcement learning (Ziebart et al., 2008), GAIL paved the way for later works such as AIRL (Fu et al., 2017) which modified GAIL's objective to allow reward recovery; DAC (Kostrikov et al., 2018), an off-policy extension of GAIL also addressing reward bias; and general f-divergence approaches like (Ke et al., 2020; Ghasemipour et al., 2020). Other works include the DICE (Nachum et al., 2019) family comprised of ValueDICE (Kostrikov et al., 2019) and its successors SoftDICE (Sun et al., 2021), SparseDICE (Camacho et al., 2021) and DemoDICE (Kim et al., 2021a). Using the Donsker-Varadhan representation along with a change of variables, ValueDICE derives a fully off-policy objective from the reverse KL, completely drubbing BC in the offline regime. Its successors are of tangential relevancy here, each augmenting it for a different domain: SoftDICE makes various amendments notably using the Wasserstein metric (also used by PWIL (Dadashi et al., 2020)), while SparseDice adds a host of regularizers to extend ValueDICE to subsampled trajectories and DemoDICE focuses on imitation with supplementary imperfect demonstrations. We note the apparent advantages of the DICE family over BC in the fully offline regime have recently been questioned (Li et al., 2022).

Our exploitation of Donsker-Varadhan was certainly inspired by ValueDICE and is somewhat reminiscent of MINE (Belghazi et al., 2018) who utilize it to estimate mutual information with a regular neural estimator. Their context and precise method however, differ substantially. Another work of relation is the often overlooked (Schroecker, 2020) which includes three approaches: SAIL, GPRIL and VDI (Schroecker & Isbell, 2017; Schroecker et al., 2019; Schroecker & Isbell, 2020). All three attack the IL problem via the forward KL. Essentially, the forward KL is broken into a behavioral cloning term as well as a state distribution term, and optimizing the objective then requires an estimate of $\nabla_\theta \log d_{\pi_\theta}(s)$. SAIL, GPRIL and VDI each propose a unique way of estimating this gradient, with GPRIL and VDI incorporating flows. (Chang et al., 2022) also utilize flows, proposing a state-only IL approach that models the state-next-state transition using conditional flows, but require dozens of expert trajectories to find success.

Finally, the most similar work is NDI (Kim et al., 2021b) who rewrite the reverse KL as $-D_{KL}(p_\pi||p_e) = \mathbb{E}_{p_\pi}[\log p_e - \log p_\pi] = J(\pi, r{=}\log p_e) + \mathcal{H}(p_\pi)$. That is, RL with rewards $\log p_e$, along with a state-action entropy term. They continue by deriving a lower bound on $\mathcal{H}(p_\pi)$, termed the SAELBO. NDI+MADE is then proposed, where in a first phase $\log p_e$ is estimated using flows followed by a second phase of optimization.

NDI has many limitations and differs dramatically from our approach. NDI claims to be non-adversarial, while true, it is only due to their loose bound. Moreover, this bound was proven superfluous in their ablation where setting $\lambda_f{=}0$ showed no reduction in performance. On top of that, we far outperform them, our evaluation is far more comprehensive and although we both use flows, the method and employment differ significantly in CFIL given the coupling of a pair with Donsker-Varadhan.

Pausing the deturpation, each of these methods above has strong merits and heavily inspired our own: We adopted the use of MAF over RealNVP from NDI+MADE (our work began in ignorance of NDI); and it was studying ValueDICE that inspired our direction to exploit the Donsker-Varadhan representation of the KL. The way in which we do however, is unique, and all the approaches above are distinct from our own.

4. Our Approach

We begin with the reverse KL as our divergence of choice, since, noting as others have (Kostrikov et al., 2019; Hazan et al., 2019; Kim et al., 2021b; Camacho et al., 2021), its minimization may be viewed as an RL problem with rewards being the log distribution ratios:

$$\arg\min_\pi D_{KL}(p_\pi||p_e) = \arg\max_\pi \mathbb{E}_{p_\pi(s,a)}\left[\log \frac{p_e(s,a)}{p_\pi(s,a)}\right] = \arg\max_\pi J(\pi, r{=}\log\frac{p_e}{p_\pi}). \quad (4)$$

Thus permitting the use of any RL algorithm for solving the IL objective, provided one has an appropriate estimate of the ratio.

Before continuing however, we first motivate our coupled approach for such estimation, by illustrating the failure of what is perhaps a more natural next step: using two independent density estimators—say flows—for each of the densities p_e and p_π directly. Practically, this would mean alternating between learning the flows and using their log-ratio as reward in an RL algorithm. The table in Section 6 concisely showcases a clear failure of this approach on all the standard benchmarks we later evaluate with.

The failure can be further understood through analyzing what we term the BC graph corresponding to an estimator of the log distribution ratio. That is, we first train a behavioral cloning agent once on sufficient expert trajectories, while rolling it out every few iterations. We then train the estimator analogously to an RL run, using a single expert trajectory along with the N saved BC rollouts. This quick process yields N estimators corresponding to N intermediate BC agents. The BC graph is then, for all i, the i'th estimator's evaluation of an i'th BC agent's trajectory, scattered against the true environment reward of that same trajectory. [3] Intuitively, a non increasing graph, means a potential RL learner with an analogously trained reward may struggle to overcome those misleading behaviors ranked high by the synthetic reward, thus preventing them from reaching expert level. In practice of course, one would want some form of approximate monotonicity or upward trend. Though importantly, a BC graph's monotonicity by no means implies the success of an RL learner with the correspondingly constructed reward. This is more than a theoretical idiosyncrasy: Many estimators will emit a perfect BC graph while completely failing all attempts in RL (see Appendix F.1). Only the reverse is true: its complete lack of an upward trend will usually imply an agent's failure.

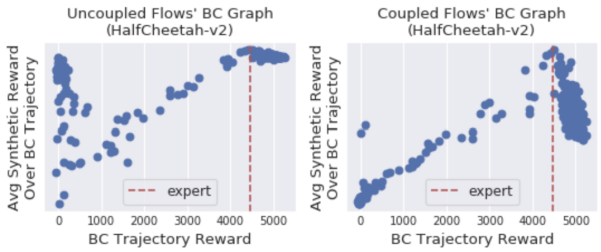

Figure 1. Left: The BC graph of an uncoupled flow for the HalfCheetah-v2 environment. Right: The BC graph of a coupled flow for the HalfCheetah-v2 environment. BC graphs for an estimator are generated by updating the estimator analogously to an RL run using N saved BC rollouts. This yields N estimators corresponding to N intermediate BC agents. The BC graph is then, for all i, the scatter of the i'th estimator's evaluation of an i'th BC agent's trajectory against its true environment reward.

Loosely formalized, given BC's objective in Equation 2 and assuming a stationary (time independent) reward function r, a monotonically increasing BC graph essentially[4] means that for all policies π_1, π_2: $J(\pi_1, r) > J(\pi_2, r)$ if $\mathbb{E}_{p_e}[\log \pi_1(a|s)] > \mathbb{E}_{p_e}[\log \pi_2(a|s)]$. Thus, further assuming continuity, a non monotonically increasing graph implies it either monotonically decreases or the existence of a local maximum. In both cases an RL learner with ob-

[3] Meaningfulness of the BC graph depends on how erratic the training was and how iteration number, loss and true environment reward line up during training. Our choice of Cheetah for illustration is motivated due to all these measures mostly aligning.

[4] Once again assumes alignment of environment reward with loss.

jective $\arg\max_\pi J(\pi, r)$ may converge on a policy with suboptimal BC loss. Since only at optimality BC recovers the expert policy, this would guarantee the agent will not meet its truly intended goal of expert mimicry. Of course, in reality, RL can overcome a certain lack of an upward trend. Moreover, the rewards are neither stationary nor identical between the BC and RL runs, only analogously constructed, so such graphs are only loosely representative. Nonetheless, we find they can be highly insightful.

As Figure 1 suggests, the independently trained flows' BC graph is quite lacking. An agent would have no incentive according to the synthetic reward to make any progress, which is precisely what occurs as the table in Section 6 demonstrates. This poor BC graph is due in part to each flow being evaluated on data completely out of its distribution (OOD), which flows are known to struggle with (Kirichenko et al., 2020). Since the two flows estimates lack true meaning when evaluated on each others data, we need to tie them together somehow: They must be *coupled*.

To perform our coupling, we employ the Donsker-Varadhan (Donsker & Varadhan, 1976) form of the KL divergence:

$$D_{KL}(p_\pi||p_e) = \sup_{x:S\times A\to\mathbb{R}} \mathbb{E}_{p_\pi(s,a)}\left[x(s,a)\right] - \log \mathbb{E}_{p_e(s,a)}\left[e^{x(s,a)}\right]. \tag{5}$$

In the above, optimality occurs with $x^* = \log \frac{p_\pi}{p_e} + C$ for any $C \in \mathbb{R}$ (Kostrikov et al., 2019; Gangwani et al., 2020; Belghazi et al., 2018). Thus after computing x^*, one recovers the log distribution ratio by simple negation, enabling use of $-x^*$ as reward in an RL algorithm to optimize our IL objective. This leads directly to our proposed approach for estimating the log distribution ratio, by coupling two flows through $x(s, a)$. That is, instead of training two flows independently, we propose to do so through maximization of Equation 5. More specifically, we inject the following inductive bias, modeling x as $x_{\psi,\phi}(s, a) = \log p_\psi(s, a) - \log q_\phi(s, a)$, where p_ψ and q_ϕ are normalizing flows.

This coupling guarantees more meaningful values when the flows are evaluated on each others data, since it has already occurred during the maximization phase, hence sidestepping the OOD issue described earlier. Figure 1 illustrates this advantage. The right shows the coupled flows' BC graph, clearly remedying the issue with their uncoupled counterparts: A potential learner will now have the proper incentive to reproduce the expert's behavior.

The drop in synthetic reward (i.e. $-x^*$) towards the end of the BC graph may seem daunting, but it actually expresses how well our estimator captures the expert's behavior: The drop occurs precisely beyond expert level, where the agent, while good in the environment, diverges from the true expert's behavior.[5]

Given this improved estimator, our full IL objective can then be written as:

$$\arg\max_\pi \min_{p_\psi, q_\phi} \log \mathbb{E}_{p_e(s,a)}\left[e^{\log \frac{p_\psi}{q_\phi}}\right] - \mathbb{E}_{p_\pi(s,a)}\left[\log \frac{p_\psi}{q_\phi}\right]. \tag{6}$$

As is commonplace in adversarial like approaches (Goodfellow et al., 2014; Ho & Ermon, 2016; Kostrikov et al., 2019), the max-min objective above is trained in an alternating fashion, switching between learning x and using $-x$ as reward in an RL algorithm. Moreover, we find it useful to use a squashing function on x, avoiding a potentially exploding KL, due to a lack of common support (subtly different then the earlier issue of OOD).

Our approach still enables training each flow independently along the way. We call this flow regularization and importantly, our method succeeds without such regularization. More specifically, for expert and agent batches of size M, this regularization involves incorporating the following additional loss function into the minimization step of Equation 6:

$$\mathcal{L} = -\frac{1}{M}\sum_{i=1}^{M} \log q_\phi(s_e^i, a_e^i) + \log p_\psi(s^i, a^i), \tag{7}$$

with $\mathcal{L}$ to be weighted by a coefficient α.

Noting that training flows is a delicate process, our approach further benefits from—though again does not require—use of a smoothing akin to the dequantization used when training normalizing flows on discrete data (Papamakarios et al., 2017). More specifically, since our input is unnormalized, we smooth each dimension with uniform noise scaled to its value. That is, if (s, a) is the vector to be smoothed, we sample uniform noise with dimension $dim((s, a))$, multiply them element-wise and add that to the original vector:

$$(s, a) \mathrel{+}= \beta \cdot (s, a) \odot u, \;\; u \sim Uniform(-\frac{1}{2}, \frac{1}{2})^{dim((s,a))}, \tag{8}$$

with weight β controlling the smoothing level. Note if regularization is also present, smoothing still applies within the additional loss $\mathcal{L}$.

Finally, combining all the above is our resulting algorithm, Coupled Flow Imitation Learning (CFIL). It is summarized in Algorithm 1, with only the number of batches per density update omitted. As in ValueDICE (Kostrikov et al., 2019),

[5] This both illustrates the tendency of BC to overfit (Li et al., 2022), while also raising concerns about the tendency for IL papers to report a table showing slight advantage in asymptotic reward as being meaningful. In truth, once at expert level, an advantage should not be claimed for slightly higher performance, unless the stated goal of the work is to do so, but that would no longer be imitation.

Algorithm 1 CFIL

Input: Expert demos $\mathcal{R}_E = \{(s_e, a_e)\}_{t=1}^N$; parameterized flow pair p_ψ, q_ϕ; off-policy RL algorithm $\mathcal{A}$; density update rate k; squashing function σ; regularization and smoothing coefficients α, β.
Define: $x_{\psi,\phi} = \sigma(\log p_\psi - \log q_\phi)$

for timestep $t = 0, 1, \ldots,$ **do**
  Take a step in $\mathcal{A}$ with reward $r = -x_{\psi,\phi}$, while filling agent buffer $\mathcal{R}_A$ and potentially updating the policy and value networks according to $\mathcal{A}$'s settings.
  if $t \mod k = 0$ **then**
    Sample expert and agent batches:
    $\{(s_e^t, a_e^t)\}_{t=1}^M \sim \mathcal{R}_E$ and $\{(s^t, a^t)\}_{t=1}^M \sim \mathcal{R}_A$
    if smooth **then**
      $(s, a) \mathrel{+}= \beta \cdot (s, a) \odot u, \;\; u \sim U(-\frac{1}{2}, \frac{1}{2})^{dim((s,a))}$
    end if
    Compute loss:
    $\mathcal{J} = \log \frac{1}{M} \sum_{i=1}^M e^{x(s_e^i, a_e^i)} - \frac{1}{M} \sum_{i=1}^M x(s^i, a^i)$
    if flow reg **then**
      Compute regularization loss:
      $\mathcal{L} = -\frac{1}{M} \sum_{i=1}^M \log q_\phi(s_e^i, a_e^i) + \log p_\psi(s^i, a^i)$
      $\mathcal{J} = \mathcal{J} + \alpha\mathcal{L}$
    end if
    Update $\psi \leftarrow \psi - \eta \nabla_\psi \mathcal{J}$
    Update $\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{J}$
  end if
end for

we found the bias due to the log-exp over the mini-batches did not hurt performance and was therefore left unhandled.

Another setting of interest is learning from observations (LFO) alone (Zhu et al., 2020; Torabi et al., 2018; 2021). That is, attempting to minimize:

$$\arg\min_\pi \; D_{KL}(d_\pi(s, s') || d_e(s, s')). \tag{9}$$

While this objective is clearly underspecified in a non injective and deterministic MDP, in practice, recovering the expert's behavior is highly feasible (Zhu et al., 2020). Seeing as none of our description above is specific to the domain of states and actions, CFIL naturally extends to LFO with no need of modification. This is in stark contrast to previous works in the LFO setting which have been highly tailored (Zhu et al., 2020). We shall demonstrate CFIL's utility in the LFO setting in the following section, where remarkably, we even find success when the flows model the single state distribution $d(s)$.

5. Experiments

We evaluate CFIL on the standard Mujoco benchmarks (Todorov et al., 2012), first comparing it to state-of-the-art imitation methods, including ValueDICE (Kostrikov et al., 2019) and their optimized implementation of DAC (Kostrikov et al., 2018), along with a customary behavioral cloning (BC) baseline. We then move to evaluation on a variety of other settings described below. We use ValueDICE's original expert demonstrations, with exception to the Humanoid environment, for which we train our own expert, since they did not originally evaluate on it. We use ValueDICE's open-source implementation to comfortably run all three baselines. NDI (Kim et al., 2021b) would be the ideal candidate for comparison, given the similarities, however no code was available. Still, we reference some relevant results described in their paper.

For CFIL, our RL algorithm of choice is SpinningUps's (Achiam, 2018) SAC (Haarnoja et al., 2018). We leave all hyper-parameters unchanged, only reducing the start-steps down to 2000, matching that of the baselines above. Our choice for both flows is a single layered MAF (Papamakarios et al., 2017), amending no hyper-parameters from the following open-source implementation (Bliznashki, 2019). This lack of tuning highlights the robustness of our method. Our density update rate is 10 batches of 100, every 1000 timesteps. We use the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 0.001. For squashing we use $\sigma = 6tanh(\frac{x}{15})$, while the smoothing and regularization coefficients are 0.5 and 1 respectively.

For all algorithms, we run 80 epochs, each consisting of 4000 timesteps, evaluating over 10 episodes after each. We do this across 5 random seeds and plot means and standard deviations. The plots are smoothed by a rolling mean with window length 5. All results are using a single expert trajectory. Appendix E shows CFIL's performance with more expert trajectories.

The top of Figure 2 shows a comparison to the baselines in the standard state-action setting. Clearly, CFIL achieves expert level performance on all tasks with merely a single expert trajectory—something the baselines fail to do. CFIL either outperforms or performs similarly to the competition in terms of asymptotic performance (though slight outperformace is not meaningful once all are at expert level), with a massive advantage in the Ant and Humanoid environments.

In terms of environment interactions required until reaching expert level: Our approach slightly lags behind the optimized DAC and largely lags behind ValueDICE. That is of course, only for the tasks they succeed on. ValueDICE—where it succeeds—clearly wins in terms of interactions needed to reach expert level, though its performance then occasionally degrades as seen in Figure 2. Being semi-offline (i.e. it can be instantiated fully offline), ValueDICE is difficult to compete with in terms of environment interactions. However, minimizing interactions was not the objective of this work, yet still we are on par.

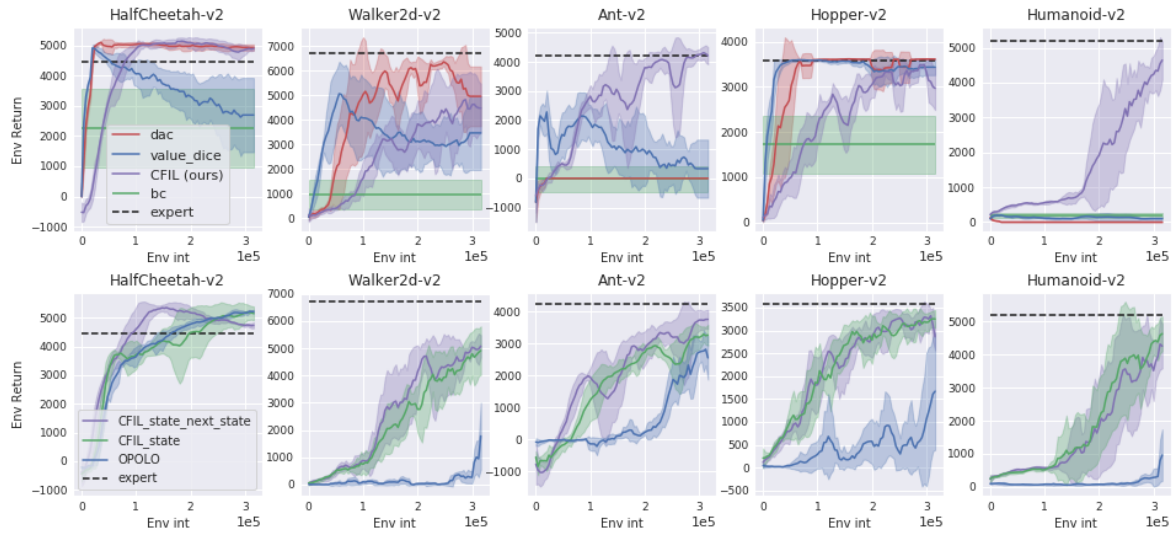

Figure 2. Top: A comparison of CFIL to ValueDICE and DAC on a single expert trajectory in the standard state-action setting. Bottom: A comparison of two versions of CFIL to OPOLO on a single expert trajectory in the LFO setting, with one version limiting itself only to single states. CFIL uses identical hyperparameters on all environments in all three incarnations, showing outstanding results, particularly in the LFO setting where it far outperforms the highly tailored competitor OPOLO.

Despite not being our aim, we do note that CFIL far outperforms NDI which requires an order of magnitude more interactions (see their appendix), who in turn outperform GAIL (Ho & Ermon, 2016) by another order of magnitude. Clearly, CFIL is still very impressive in terms of environment interactions and the lag between us and DAC may simply be due to a lack of tuning of CFIL's RL component. We deliberately avoid this tuning, preferring to keep the RL algorithm a black box, since too much tuning may jeopardize robustness and extendability to other settings: We aspire for robust competitiveness rather than a tailored, minor and frail improvement.

We now turn to the state-only and subsampled regimes. Settings in which ValueDICE finds no dice: By its very nature, ValueDICE is inapplicable to the state-only scenario (see Appendix 9.8 of (Zhu et al., 2020)) and extensive experiments already showed its failure when demonstrations are subsampled (Li et al., 2022). SparseDICE (Camacho et al., 2021) attempted to remedy this by adding a series of regularizers. However, they require at least 10 demonstrations, each subsampled at a rate of 0.1, not nearly the sparsity of the settings we next describe.

First in the state-only regime, we compare against OPOLO (Zhu et al., 2020), a state-of-the-art LFO method. We avoid comparison with on-policy LFO methods, like GAIfO (Torabi et al., 2018), due to the sheer number of interactions they require. We use OPOLO's open-source implementation, with the same setup described previously, once again using only a single expert trajectory. Importantly, CFIL uses identical hyperparameters to the previous setting.

The bottom of Figure 2 shows two versions of CFIL compared with OPOLO. One estimating the state-next-state distribution ratio, while the other limiting itself to the single state distribution. As can be seen, both incarnations of CFIL once again achieve expert level performance on all environments. This is in sharp contrast to OPOLO which, despite being highly tailored towards the LFO setting, employing inverse action models, discriminators and a host of practical considerations, struggles immensely when provided with a single expert trajectory (they originally report success with 4 trajectories). All the while, CFIL requires no amendment whatsoever, extending simply and elegantly, while distinctly outdoing OPOLO on every imitation learning metric. CFIL's results in the LFO setting are nothing short of remarkable. We effectively set a new state-of-the-art for learning from observations alone, while our method was not originally designed for it.

We finally turn to the subsampled regime. Comparing again to DAC with a similar setup as before. Our comparison here involves four different subsampling rates, sampling every 10, 20, 50 and 100'th transition. Since we are still working with a single expert trajectory, the final rate implies learning from 10 state-action pairs alone. For this setting, we found the hyperparameters used above did not extend well enough. Specifically, the use of flow regularization seemed to hurt performance in some environments. We therefore drop the regularization to 0, leaving the smoothing at 0.5 and amending the squasher to be $\sigma = 3tanh(\frac{x}{10})$. We use these parameters for all environments in all four new settings, with exception to cheetah who, when subsampled,

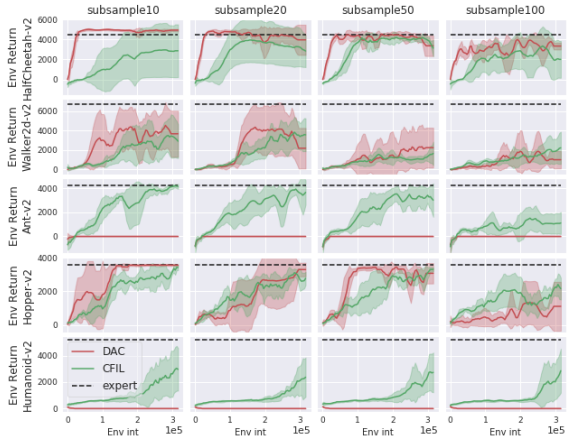

Figure 3. A comparison of CFIL and DAC on four subsampling rates with a single expert trajectory. Subsample$N$ refers to sampling every N'th transition in the trajectory.

actually struggles without the flow regularization.

Figure 3 shows the comparison in the subsampled setting, with CFIL once again demonstrating stellar performance. While DAC has the advantage in the Cheetah environment, where single crashing seeds significantly reduced the apparent performance of CFIL, CFIL still generally outperforms it, remarkably able to recover the expert behavior when only provided with a measly 10 state-action pairs.

6. Ablation

We now concisely ablate various aspects of our approach. We first put into question the need for our squasher, our coupling and our inductive bias, by comparing to NoSquash; IndFlow and IndFlowNS; and RegularNet. NoSquash is CFIL stripped of the squasher. IndFlow and IndFlowNS refer to learning the log distribution ratio directly using independent flows (as in section 4), with and without a squasher, respectively. RegularNet alters CFIL's inductive bias by setting x to a regular MLP, completely avoiding the use of flows (reminiscent of MINE (Belghazi et al., 2018)). Along with these we also run a numerator only approach termed Numerator, which simply involves direct learning of the expert's distribution with a single flow, then using it alone as reward in an RL algorithm: $J(\pi, r = \log p_e)$ (akin to NDI (Kim et al., 2021b) with $\lambda_f = 0$).

We run these with our usual setup of a single expert trajectory and compute their overall score as the average normalized asymptotic reward over all 25 seeds (5 for each environment). Table 1 summarizes these results, showing all the CFIL alternatives fail, demonstrating the necessity of its components.

Next we vary CFIL's smoothing and regularization coefficients to test its sensitivity. Specifically, we run CFIL (that of the three settings in Figure 2), but with pairs $\alpha, \beta \in \{0, 0.25, 0.5, 0.75, 1\}$, along with select others, and compute their averages and standard deviations of normalized asymptotic rewards over all 25 seeds. Figure 4 illustrates the results for these runs, showcasing both the utility of the smoothing and regularization as well as CFIL's robustness to them.

Table 1. A comparison of CFIL to some alternatives, ablating its squasher, coupling and inductive bias. The score is the average normalized asymptotic reward over 25 seeds (5 for each environment). Note: rows with two values indicate runs with smoothing of 0 (left) and 0.5 (right).

	SCORE
EXPERT	1
CFIL	**1.012**
NOSQUASH	-0.091
REGULARNET	$0.196 \mid 0.190$
INDFLOW	$0.158 \mid 0.127$
INDFLOWNS	$0.090 \mid 0.072$
NUMERATOR	$-0.051 \mid -0.001$

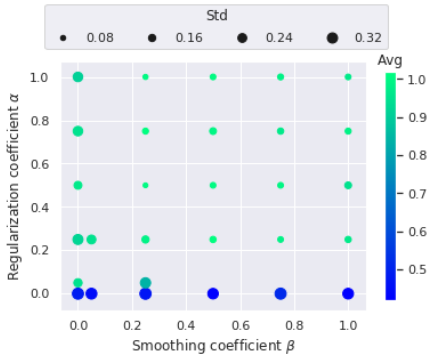

Figure 4. Averages (as color) and standard deviations (as size) of normalized asymptotic rewards for CFIL with varied levels of smoothing and regularization. Each point summarizes 25 seeds (5 per environment). The utility of the smoothing and regularization is apparent as well as CFIL's lack of sensitivity to them.

7. Conclusion

We presented CFIL, a unique approach to imitation learning based on the coupling of a pair of flows. CFIL introduced many novelties including its estimator for the log ratio, its smoothing and regularization and more generally its employment of flows, while we also performed a unique analysis using BC graphs and raised concerns about trends in relevant literature. Crucially, CFIL was empirically shown to outperform state-of-the-art baselines in a variety of settings with only a single expert trajectory. A future work could include coupled flows for general ratio estimation.

References

Achiam, J. Spinning Up in Deep Reinforcement Learning. 2018.

Belghazi, M. I., Baratin, A., Rajeswar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, R. D. Mine: mutual information neural estimation. *arXiv preprint arXiv:1801.04062*, 2018.

Bliznashki, K. `https://github.com/kamenbliznashki/normalizing_flows`, 2019.

Camacho, A., Gur, I., Moczulski, M. L., Nachum, O., and Faust, A. Sparsedice: Imitation learning for temporally sparse data via regularization. In *ICML 2021 Workshop on Unsupervised Reinforcement Learning*, 2021.

Chang, W.-D., Higuera, J. C. G., Fujimoto, S., Meger, D., and Dudek, G. Il-flow: Imitation learning from observation using normalizing flows. *arXiv preprint arXiv:2205.09251*, 2022.

Dadashi, R., Hussenot, L., Geist, M., and Pietquin, O. Primal wasserstein imitation learning. *arXiv preprint arXiv:2006.04678*, 2020.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Donsker, M. D. and Varadhan, S. S. Asymptotic evaluation of certain markov process expectations for large time—iii. *Communications on pure and applied Mathematics*, 29 (4):389–461, 1976.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. Neural spline flows. In *Advances in Neural Information Processing Systems*, pp. 7511–7522, 2019.

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

Fujimoto, S., Van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062. PMLR, 2019.

Gangwani, T., Peng, J., and Zhou, Y. Harnessing distribution ratio estimators for learning agents with quality and diversity. *arXiv preprint arXiv:2011.02614*, 2020.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. Made: Masked autoencoder for distribution estimation. In *International conference on machine learning*, pp. 881–889. PMLR, 2015.

Ghasemipour, S. K. S., Zemel, R., and Gu, S. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*, pp. 1259–1277. PMLR, 2020.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Hazan, E., Kakade, S., Singh, K., and Van Soest, A. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pp. 2681–2691. PMLR, 2019.

Ho, J. and Ermon, S. Generative adversarial imitation learning. *arXiv preprint arXiv:1606.03476*, 2016.

Ho, J., Chen, X., Srinivas, A., Duan, Y., and Abbeel, P. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pp. 2722–2730. PMLR, 2019.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural autoregressive flows. In *International Conference on Machine Learning*, pp. 2078–2087. PMLR, 2018.

Islam, R., Teru, K. K., Sharma, D., and Pineau, J. Off-policy policy gradient algorithms by constraining the state distribution shift. *arXiv preprint arFXiv:1911.06970*, 2019.

Ke, L., Choudhury, S., Barnes, M., Sun, W., Lee, G., and Srinivasa, S. Imitation learning as f-divergence minimization. In *International Workshop on the Algorithmic Foundations of Robotics*, pp. 313–329. Springer, 2020.

Kim, G.-H., Seo, S., Lee, J., Jeon, W., Hwang, H., Yang, H., and Kim, K.-E. Demodice: Offline imitation learning with supplementary imperfect demonstrations. In *International Conference on Learning Representations*, 2021a.

Kim, K., Jindal, A., Song, Y., Song, J., Sui, Y., and Ermon, S. Imitation with neural density models. *Advances in Neural Information Processing Systems*, 34:5360–5372, 2021b.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pp. 10215–10224, 2018.

Kirichenko, P., Izmailov, P., and Wilson, A. G. Why normalizing flows fail to detect out-of-distribution data. *arXiv preprint arXiv:2006.08545*, 2020.

Kobyzev, I., Prince, S. J., and Brubaker, M. A. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

Kostrikov, I., Agrawal, K. K., Dwibedi, D., Levine, S., and Tompson, J. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*, 2018.

Kostrikov, I., Nachum, O., and Tompson, J. Imitation learning via off-policy distribution matching. *arXiv preprint arXiv:1912.05032*, 2019.

Lee, L., Eysenbach, B., Parisotto, E., Xing, E., Levine, S., and Salakhutdinov, R. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

Li, Z., Xu, T., Yu, Y., and Luo, Z.-Q. Rethinking valuedice: Does it really improve performance? *arXiv preprint arXiv:2202.02468*, 2022.

Liu, Q., Li, L., Tang, Z., and Zhou, D. Breaking the curse of horizon: Infinite-horizon off-policy estimation. *Advances in Neural Information Processing Systems*, 31, 2018.

Mutti, M., De Santi, R., De Bartolomeis, P., and Restelli, M. Challenging common assumptions in convex reinforcement learning. *arXiv preprint arXiv:2202.01511*, 2022.

Nachum, O., Chow, Y., Dai, B., and Li, L. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. *Advances in Neural Information Processing Systems*, 32, 2019.

Nota, C. and Thomas, P. S. Is the policy gradient a gradient? *arXiv preprint arXiv:1906.07073*, 2019.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.

Papamakarios, G., Nalisnick, E. T., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22(57):1–64, 2021.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 16–17, 2017.

Qin, Z., Chen, Y., and Fan, C. Density constrained reinforcement learning. In *International Conference on Machine Learning*, pp. 8682–8692. PMLR, 2021.

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

Schroecker, Y. and Isbell, C. Universal value density estimation for imitation learning and goal-conditioned reinforcement learning. *arXiv preprint arXiv:2002.06473*, 2020.

Schroecker, Y. and Isbell, C. L. State aware imitation learning. In *Advances in Neural Information Processing Systems*, pp. 2911–2920, 2017.

Schroecker, Y., Vecerik, M., and Scholz, J. Generative predecessor models for sample-efficient imitation learning. *arXiv preprint arXiv:1904.01139*, 2019.

Schroecker, Y. K. D. *Manipulating State Space Distributions for Sample-Efficient Imitation-Learning*. PhD thesis, Georgia Institute of Technology, 2020.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sun, M., Mahajan, A., Hofmann, K., and Whiteson, S. Softdice for imitation learning: Rethinking off-policy distribution matching. *arXiv preprint arXiv:2106.03155*, 2021.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

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

Torabi, F., Warnell, G., and Stone, P. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

Torabi, F., Warnell, G., and Stone, P. Dealio: Data-efficient adversarial learning for imitation from observation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2391–2397. IEEE, 2021.

Zahavy, T., O'Donoghue, B., Desjardins, G., and Singh, S. Reward is enough for convex mdps. *Advances in Neural Information Processing Systems*, 34:25746–25759, 2021.

Zhang, J., Koppel, A., Bedi, A. S., Szepesvari, C., and Wang, M. Variational policy gradient method for reinforcement learning with general utilities. *Advances in Neural Information Processing Systems*, 33:4572–4583, 2020.

Zhu, Z., Lin, K., Dai, B., and Zhou, J. Off-policy imitation learning from observations. *Advances in Neural Information Processing Systems*, 33:12402–12413, 2020.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. Maximum entropy inverse reinforcement learning. 2008.

A. Reproducibility

Code for reproducibility of CFIL, including a detailed description for reproducing our environment, is available at `https://github.com/gfreund123/cfil`.

B. Ablation Results per Environment

Table 2. Extended version of Table 1, showing the ablation results per environment. The table contains averages and standard deviations of normalized asymptotic rewards over 5 seeds.

	HALFCHEETAH-V2	WALKER2D-V2	ANT-V2	HOPPER-V2	HUMANOID-V2
CFIL	1.122±0.015	0.8542±0.063	1.095±0.022	0.986±0.011	1.004±0.015
NOSQUASH	-0.056±0.074	-0.002±0.001	-0.411±0.377	0.001±0.000	0.010±0.006
REGULARNET (SMOOTH=0)	-0.000±0.000	0.149±0.002	0.185±0.016	0.291±0.009	0.357±0.312
REGULARNET (SMOOTH=0.5)	-0.000±0.000	0.152±0.001	0.169±0.017	0.289±0.002	0.340±0.155
INDFLOW (SMOOTH=0)	0.642±0.094	0.000±0.002	0.000±0.001	0.138±0.306	0.012±0.000
INDFLOW (SMOOTH=0.5)	0.611±0.068	0.011±0.018	0.000±0.001	0.001±0.000	0.012±0.000
INDFLOWNS (SMOOTH=0)	0.203±0.183	0.082±0.058	0.126±0.176	0.026±0.031	0.017±0.001
INDFLOWNS (SMOOTH=0.5)	0.290±0.255	0.014±0.025	0.030±0.029	0.010±0.017	0.017±0.001
NUMERATOR (SMOOTH=0)	-0.047±0.050	0.026±0.034	-0.270±0.230	0.011±0.009	0.023±0.015
NUMERATOR (SMOOTH=0.5)	-0.006±0.006	0.007±0.015	-0.037±0.035	0.011±0.000	0.015±0.001

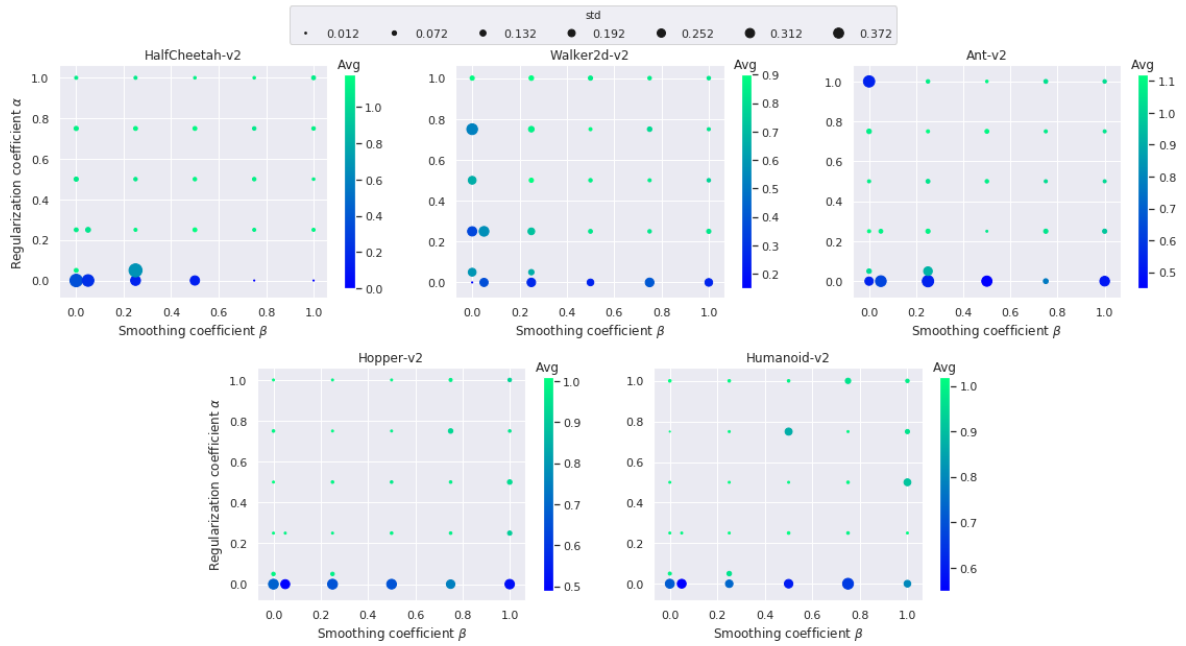

Figure 5. Extended version of Figure 4, showing per environment the average (as color) and standard deviations (as size) of normalized asymptotic rewards for CFIL with varied levels of smoothing and regularization. Each point summarizes 5 seeds. Once again, the importance of the smoothing and regularization is apparent, as well as CFIL lack of sensitivity.

C. Justification of Modeling the Undiscounted State Distribution

With the goal in an MDP being policy optimization, standard RL taxonomy divides between value-based and policy-based methods. The policy gradient theorem (Sutton et al., 2000), being the foundation of policy-based methods, provides the following expression for the gradient of J with respect to a parameterized policy π_θ:

$$\nabla_\theta J(\pi_\theta) = \sum_s d_\pi^\gamma(s) \sum_a \frac{d\pi_\theta(a|s)}{d\theta} Q^\pi(s,a) \tag{10}$$

where $d^\gamma_\pi(s) = \sum_{t=0}^{\infty} \gamma^t Pr(s_t = s | s_0 \sim p_0)$ is the improper discounted state distribution. Equation (10) enables the construction of policy optimization algorithms such as REINFORCE (Sutton & Barto, 2018), which are based on stochastic gradient descent (SGD). However, garnering some criticism (Nota & Thomas, 2019), most modern policy gradient methods (Schulman et al., 2017; Fujimoto et al., 2018; Haarnoja et al., 2018) opt to ignore the discount factor in $d^\gamma_\pi(s)$, which disqualifies them as SGD methods, hence losing guarantees it provides. Instead, they update the policy with a similar expression to (10) in which $d^\gamma_\pi(s)$ is exchanged with the undiscounted version $d_\pi(s) = \sum_{t=0}^{\infty} Pr(s_t = s | s_0 \sim p_0)$. The reason being due to stability and reduction of variance (see Appendix A of (Haarnoja et al., 2018)). We in this work follow suit, modeling the undiscounted $d_\pi(s)$.

D. More on RealNVP and MAF

In pursuit of inspiring more research incorporating flow-based models of the state distribution, and since flows are presumably the topic least familiar to an IL researcher reading the paper, we now provide a brief but thorough review of RealNVP and MAF, along with some additional discussion.

Following the description in Section 2.2: For both training and generation to be efficient, f_θ must be easy to evaluate, easy to invert and the determinant of its Jacobian must be easy to compute. Moreover, for training to be plausible, f_θ must be expressive enough to capture complex high-dimensional distributions.

RealNVP (Dinh et al., 2016) uses coupling layers (distinct from our coupled approach) to achieve the above requirements. As described in the paper (Dinh et al., 2016), a coupling layer receives a D dimensional input vector x and outputs:

$$\begin{aligned} y_{1:d} &= x_{1:d} \\ y_{d+1:D} &= x_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d}) \end{aligned} \tag{11}$$

where $d < D$, s and t are functions from $R^d \to R^{D-d}$ and $\odot$ represents an element-wise product. Since coupling layers leave certain variables unchanged, they are composed in an alternating pattern to allow all variables to be transformed. Note that the Jacobian of this transformation is triangular, and its determinant is simply $exp(\sum_j s(x_{1:d})_j)$. Note further that the inverse of this transformation is also a simple computation:

$$\begin{aligned} x_{1:d} &= y_{1:d} \\ x_{d+1:D} &= (y_{d+1:D} - t(x_{1:d})) \odot \exp(-s(x_{1:d})) \end{aligned} \tag{12}$$

Finally, note that s and t have no requirement to be invertible, so they can be arbitrary neural networks.

Masked Autoregressive Flow (MAF) (Papamakarios et al., 2017) is a generalization of RealNVP which substitutes (11) with the autoregressive

$$y_i = x_i \exp(s_i(x_{1:i-1})) + t_i(x_{1:i-1}) \tag{13}$$

As stated in 2.2, this improves expressiveness but prevents single pass sampling. However, density evaluation—our main concern—can still be performed efficiently using masked autoencoders (Germain et al., 2015).

Generally, normalizing flows are sensitive objects, after all, density estimation is a difficult task, particularly with such limited data of such high dimension. Given the delicacy involved in their training, much deliberation was done over the potential influence of various things. In particular, the need for normalization, something common in other IL works (Kostrikov et al., 2018; 2019; Schroecker & Isbell, 2020). Although we did finally settle on the more robust approach of using no normalization, or any other special pre-processing, still, this could be useful and even necessary for other applications incorporating density models of the state distribution. Moreover, despite the existence of higher capacity flows (Kingma & Dhariwal, 2018; Huang et al., 2018; Durkan et al., 2019; Ho et al., 2019), when learning from such few demonstrations, seemingly more powerful flows may be far more prone to overfitting then their less expressive counterparts. We aim to inspire others to take advantage of modern powerful density estimators for explicit modeling of the state and state-action distributions, and apply them all around the reinforcement learning literature.

E. CFIL With More Trajectories

Figure 6. CFIL's performance with varying numbers of expert trajectories (1, 5, and 10), demonstrating its consistency. Note, the single trajectory run is identical to that of Figure 2.

F. BC Analysis Extras

F.1. RegularNet's BC Graph

Figure 7 shows the BC graph of RegularNet (see ablation) for the HalfCheetah-v2 environment. This serves as an example estimator which fails in RL despite a good BC graph (its failure in RL is illustrated in Table 1). As described in Section 4, a BC graph's utility is mainly one way. That is, when poor it is indicative of failure in RL, but the reverse is not necessarily the case. Beyond that however, it also provides insight into what is captured by an estimator as well as being valuable as a quick sanity test of an estimator's quality (can be generated quickly on saved BC trajectories).

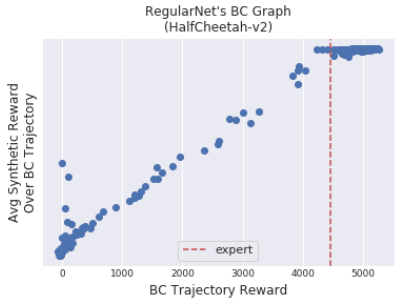

Figure 7. The BC graph of RegularNet (see ablation) for the HalfCheetah-v2 environment.

F.2. Two-Dimensional BC Analysis

What we have termed the BC graph is the diagonal slice of a broader two-dimensional picture. Recall the generation process described in Section 4: BC graphs show the i'th estimator's evaluation of the i'th BC trajectory against the true environment reward of that same trajectory. However, one may take interest in evaluating all N estimators on all N trajectories. To that end, out of the $N = 375$ estimator updates, we show every 25'th (along with the first), evaluated on all the BC trajectories, scattered against their true environment rewards. This is illustrated in Figures 8, 9 and 10 corresponding to coupled flows, uncoupled (independent) flows and RegularNet, respectively. The figures provide insight into what individual estimators captured as well as how they morphed over time. For example, for the coupled flows, we can see how the later estimators (synthetic rewards) incentivize the agent not to go beyond expert level, while the early ones do not. This is reasonable, since estimator i is only encountered by policy π_i (hence why the BC graph is only the diagonal cut). For this reason, evaluations of early estimators on much later trajectories aren't particularly insightful with regards to what an RL agent with the analogously constructed reward will face. Still, they're interesting in their own right, showing what has been captured by the estimator at that time.

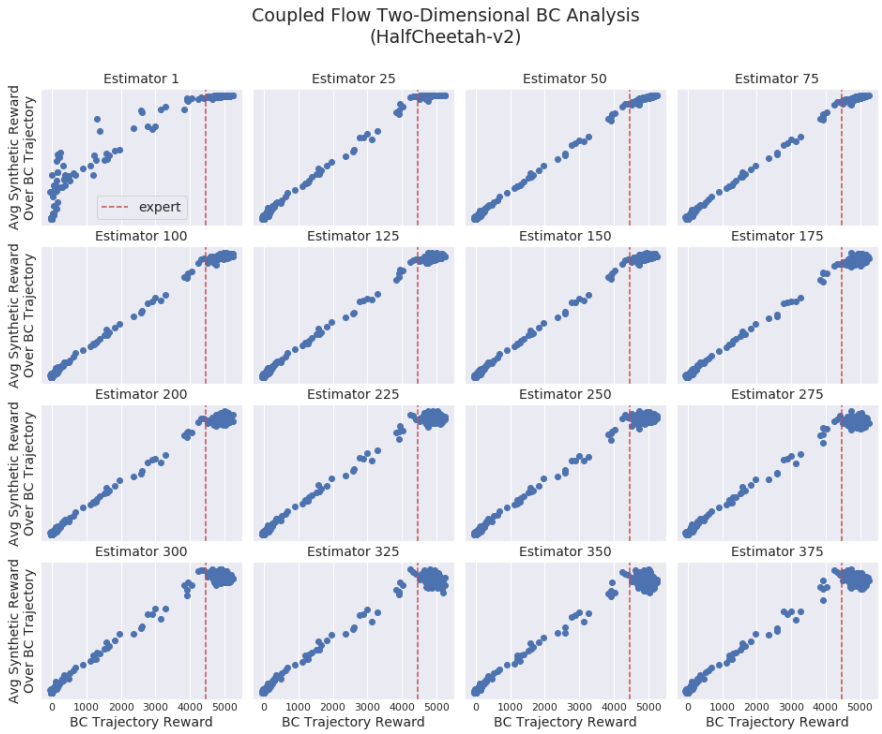

Figure 8. Two-dimensional BC analysis of coupled flows for the HalfCheetah-v2 environment. In contrast to the BC graph which is the i'th estimator's evaluation of the i'th BC trajectory vs the true environment reward of that same trajectory, here we see scatters for individual estimators along the way. That is, their evaluations of all BC trajectories scattered against the true environment rewards of the trajectories.

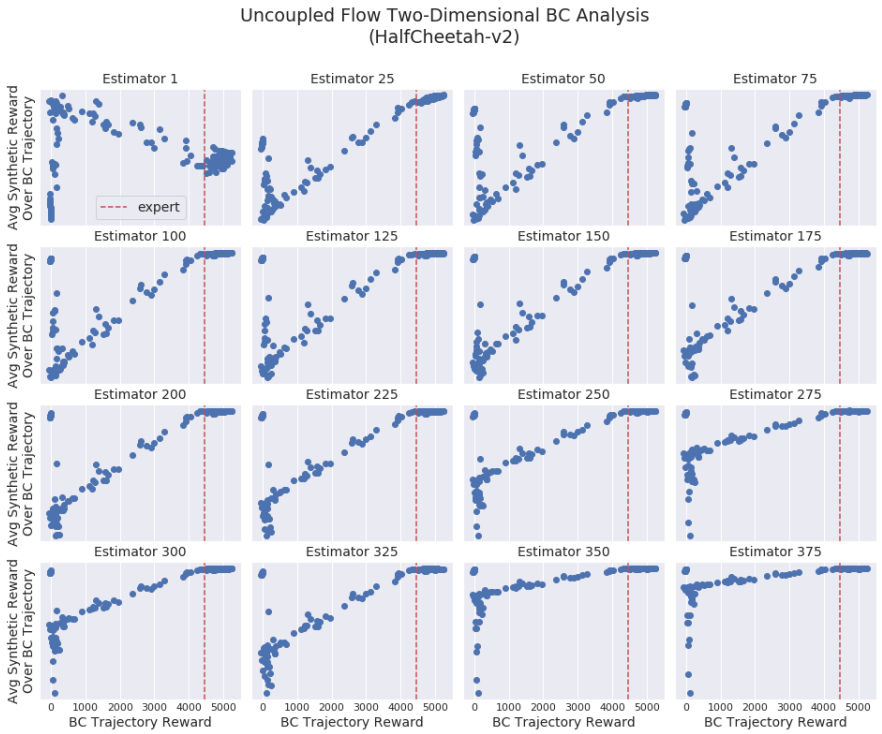

Figure 9. Two-dimensional BC analysis of uncoupled flows for the HalfCheetah-v2 environment. See Figure 8's caption.

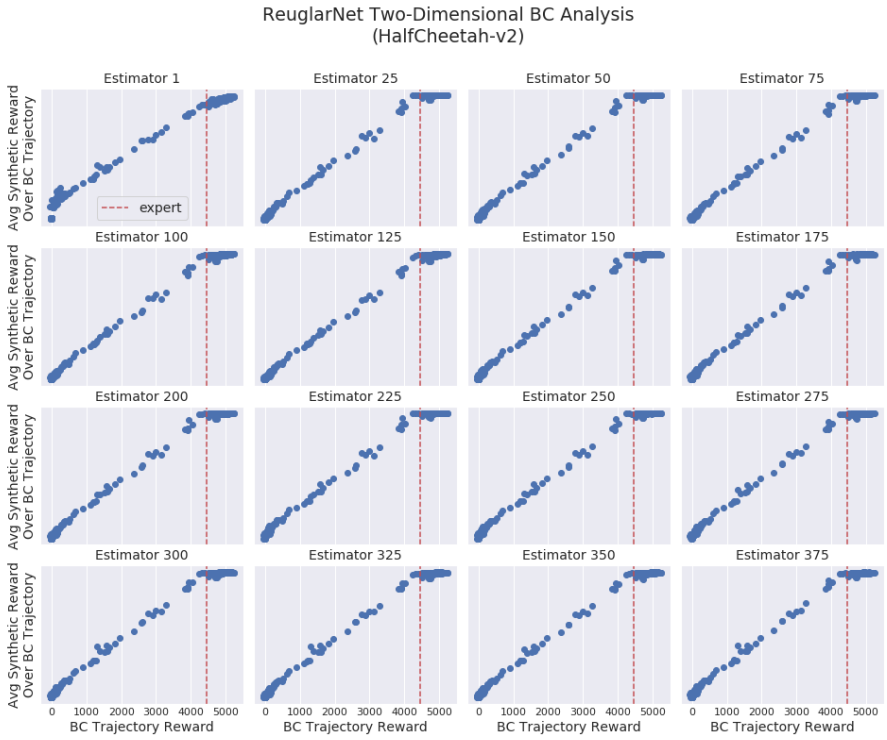

Figure 10. Two-dimensional BC analysis of RegularNet for the HalfCheetah-v2 environment. See Figure 8's caption.