Markov Balance Satisfaction Improves Performance in Strictly Batch Offline Imitation Learning

Rishabh Agrawal[1], **Nathan Dahlin**[2], **Rahul Jain**[1], **Ashutosh Nayyar**[1]
[1] University of Southern California
[2] University at Albany, SUNY
rishabha@usc.edu, ndahlin@albany.edu, {rahul.jain, ashutosn}@usc.edu

Abstract

Imitation learning (IL) is notably effective for robotic tasks where directly programming behaviors or defining optimal control costs is challenging. In this work, we address a scenario where the imitator relies solely on observed behavior and cannot make environmental interactions during learning. It does not have additional supplementary datasets beyond the expert's dataset nor any information about the transition dynamics. Unlike state-of-the-art (SOTA) IL methods, this approach tackles the limitations of conventional IL by operating in a more constrained and realistic setting. Our method uses the Markov balance equation and introduces a novel conditional density estimation-based imitation learning framework. It employs conditional normalizing flows for transition dynamics estimation and aims at satisfying a balance equation for the environment. Through a series of numerical experiments on Classic Control and MuJoCo environments, we demonstrate consistently superior empirical performance compared to many SOTA IL algorithms.

Introduction

Reinforcement Learning (RL) has provided us with some very notable achievements over the past decade from mastering complex games (Mnih et al. 2015; Silver et al. 2016; Vinyals et al. 2019) to advancing protein structure prediction systems (Jumper et al. 2021), and now proficiency at coding and high school-level mathematics. And yet, all of these achievements are built on a fundamental hypothesis that the "reward is enough" (Silver et al. 2021). But in the real world, when only data is available, there is no natural reward model available. Countless hours are spent on reward engineering. In fact, lately, there has been tremendous progress in building reward models for language and multi-modal generative AI systems. And yet, they have led to their own set of challenges, namely reward hacking and over-optimization, lack of diversity and robustness in learned policies.

Classically, in imitation learning, this was sought to be addressed via inverse RL methods (Abbeel and Ng 2004; Ng and Russell 2000), e.g., the MaxEntropy-IRL algorithm (Ziebart et al. 2008) that first estimates a reward function from demonstration data and then uses it with RL algorithms to find near-optimal policies. Unfortunately, this has two limitations: First, the reward function estimation problem is ill-posed (Baheri 2023) and any additional criterion introduce estimation errors. Second, the demonstrator (e.g., a human subject) may not be optimizing with respect to any reward function at all! Additionally, IRL algorithms are computationally intensive, particularly in high-dimensional state spaces, limiting their scalability (Barnes et al. 2024). Thus, there is a need to develop Imitation Learning (IL) algorithms that don't depend on reward estimation as a first step.

Behavioral Cloning (BC) is one such classical IL method (Pomerleau 1988). It is inspired by supervised learning and learns a map from states to actions from trajectory data. Unfortunately, it does not account for the fact that such trajectory data often satisfies a Markov balance equation (MBE). In fact, this MBE is the only mathematical structure that ties the trajectory data together. And if we don't use it, we can expect we will not do as well as possible. Thus, it is not unexpected that the BC method is known to be vulnerable to error propagation and covariate shift issues, thus constraining its generalization capabilities (Ross and Bagnell 2010). This is typically sought to be addressed by allowing for additional online interactions with the environment or the demonstrator, or using supplementary data (Ross, Gordon, and Bagnell 2011; Piot, Geist, and Pietquin 2016). Unfortunately, in many practical situations, this is simply not a possibility such as in autonomous vehicles and healthcare, where engaging directly with the environment is often infeasible due to safety concerns or high costs. Therefore, we must learn from the offline data we already have. A premise of ours is that just like in training of LLMs, a combination of supervised fine-tuning and RL with human feedback improves LLM performance, in the same way, an imitation learning method that combines behavior cloning policy with accounting for Markovian dynamics will perform better as well.

At the core of imitation learning is distribution matching, which treats state-action pairs from expert demonstrations as samples drawn from a target distribution. The objective is to develop a policy that minimizes the divergence between this target distribution and the distribution induced by the imitator's policy. Adversarial Imitation Learning (AIL) methods use this principle (Ho and Ermon 2016; Fu, Luo, and Levine 2018; Ke et al. 2021), but they face a significant limitation: estimating density ratios requires on-policy samples from the environment. This need for continuous interaction

with the environment makes AIL methods impractical when only offline data is available.

To address the availability of limited expert data, (Yue et al. 2023a; Xu et al. 2022) propose a supplementary data framework in imitation learning. This method uses additional data, cheaply gathered by executing suboptimal policies, to augment the expert dataset. However, such supplementary data may not be available, and in fact may even cause distribution shifts due to out-of-expert-distribution trajectories, which may degrade model performance.

In this paper, we focus on the strictly batch imitation learning problem (i.e., no further interaction or supplementary data) and propose a novel approach to imitation learning that does not do *reward model estimation*, or *distribution matching* between occupation measures via on-policy samples. Instead, it learns a policy that minimizes a loss function related to satisfaction of the Markov balance equation, which is the one and only fundamental relationship we know about the trajectory data. This Markov balance equation involves two conditional state-action transition density functions which need to be learnt from data. Density estimation, and especially conditional density estimation is rather tricky, especially for continuous state and action space settings. Fortunately, recently developed normalizing flows methods (Dinh, Sohl-Dickstein, and Bengio 2017) have proven remarkably well-suited for this problem. Furthermore, we regularize the MBE loss around a behavioral cloning policy, thus combining supervised and RL methods for imitation learning in a natural manner. This *novel* combination enables excellent numerical performance compared to state-of-the-art IL/IRL/AIL methods in the considered settings across a range of Classic Control and MuJoCo tasks.

Related Work. ***Offline IRL.*** To circumvent costly online environmental interactions in classic IRL, offline IRL aims to infer a reward function and recover the expert policy solely from a static dataset without accessing the environment. (Klein, Geist, and Pietquin 2011) introduced *LSTD-*μ, extending classic apprenticeship learning (Abbeel and Ng 2004) to batch and off-policy cases for computing feature expectations, while (Klein et al. 2012) developed a score function-based classification algorithm to output the reward function. (Lee, Srinivasan, and Doshi-Velez 2019a) propose *DSFN*, using a transition-regularized imitation network for an initial policy close to expert behavior. However, these methods assume complete knowledge of reward features, which is often impractical for complex problems due to the problem-dependent nature of feature selection (Arora and Doshi 2021). *RCAL* (Piot, Geist, and Pietquin 2014) employs a boosting method to minimize a large margin objective with a regularization term, avoiding feature selection steps. (Chan and van der Schaar 2021a) introduced *AVRIL*, jointly learning an approximate posterior distribution over reward and policy. (Garg et al. 2021a) proposed *IQ-Learn*, using a learned soft Q-function to represent both reward and policy implicitly. Despite these advances, they struggle with covariate shift and reward extrapolation errors, impacting performance in novel environments (Yue et al. 2023b). *CLARE* (Yue et al. 2023b) incorporates conservatism in reward estimation but performs poorly with low-quality transition samples (Zeng et al. 2023) and requires additional diverse datasets unavailable in our setting.

Offline IL. EDM (Jarrett, Bica, and van der Schaar 2020b) captures the expert's state occupancy measure by training an explicit energy-based model but faces significant limitations and is unsuitable for continuous action domains (Swamy et al. 2021). DICE (DIstribution Correction Estimation) methods perform stationary distribution matching of state-action pairs between the learner's policy and expert demonstrations. (Kostrikov, Nachum, and Tompson 2020a) introduces *ValueDICE*, which minimizes the Donsker-Varadhan representation of KL divergence between these stationary distributions but suffers from biased gradient estimates due to logarithmic terms applied to expectations. SoftDICE (Sun et al. 2021) addresses these limitations by using the Earth-Mover Distance (EMD) for distribution matching, eliminating problematic logarithmic and exponential terms. DemoDICE (Kim et al. 2022) and SMODICE (Ma et al. 2022) assume additional demonstration data of unknown degrees of optimality beyond expert's data to deal with the narrow support of the expert data distribution, but such information is unavailable in our setup. DICE methods involve two gradient terms for value function learning: the forward gradient (on the current state) and the backward gradient (on the next state). However, conflicting directions between these gradients can lead to performance degradation (Mao et al. 2024a). To address this, (Mao et al. 2024a) introduces ODICE, an orthogonal-gradient update method that projects the backward gradient onto the normal plane of the forward gradient, ensuring effective state-action-level constraints and better convergence.

Our work is clearly differentiated from prior work in not doing stationary distribution matching, but instead in first using normalizing flows in conditional state-action density estimation, and then using these in ensuring Markov balance equation satisfaction.

Preliminaries

The Imitation Learning Problem. An infinite horizon discounted Markov decision process (MDP) M is defined by the tuple (S, A, T, r, γ) with states $s \in S$, actions $a \in A$ and successor states $s' \in S$ drawn from the transition function $T(s'|s, a)$. The reward function $r : S \times A \to \mathbb{R}$ maps state-action pairs to scalar rewards, and γ is the discount factor. Policy π is a probability distribution over actions conditioned on state and is given by $\pi(a|s) = P_\pi(a_t = a|s_t = s)$, where $a_t \in A$, $s_t \in S$, $\forall t = 0, 1, 2, \cdots$. The induced occupancy measure of a policy is given as $\rho_\pi(s, a) := \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \mathbf{1}_{s_t=s, a_t=a}]$, where the expectation is taken over $a_t \sim \pi(\cdot|s_t)$, $s_{t+1} \sim T(\cdot|s_t, a_t)$ for all t, and the initial state s_0. The corresponding state-only occupancy measure is given as $\rho_\pi(s) = \sum_a \rho_\pi(s, a)$. In the offline imitation learning (IL) framework, the agent is provided with trajectories generated by a demonstration policy π_D, collected as $D = \{(s_0, a_0), (s_1, a_1), (s_2, a_2), ...\}$; and is not allowed any further interaction with the environment. The data D does *not* include any reward r_t at each time step. Indeed, rather than long-term reward maximization, the IL objective is to learn a policy $\pi^\star$ that is close to π_D in the

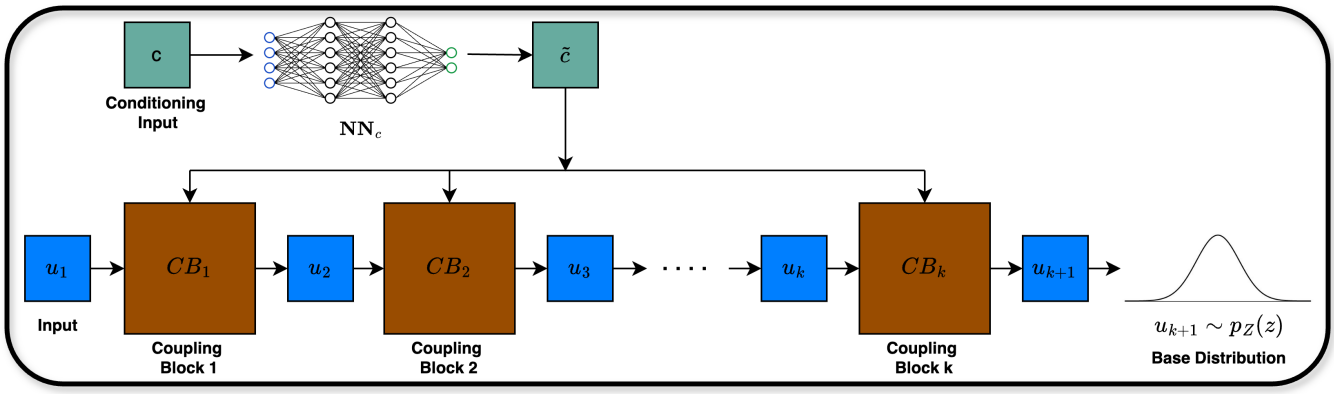


Figure 1: A high-level description of Conditional Normalizing Flow, where CB_i is the i^{th} coupling block.

following sense (Yue and Le 2018):

$$\pi^\star \in \arg\min_{\pi \in \Pi} \mathbb{E}_{s\sim\rho_\pi}[\mathcal{L}(\pi(\cdot|s), \pi_D(\cdot|s))], \tag{1}$$

where Π is the set of all randomized (Markovian) stationary policies, and $\mathcal{L}$ is a chosen loss function. In practice, (1) can only be solved approximately since π_D is unknown and only transitions are observed in the dataset D.

Normalizing Flows. The imitation learning approach we introduce depends on transition density estimation. Despite advancements in statistical theory, conditional density estimation is a difficult problem due to a lack of clarity on what parametric families of density functions are good candidates. We make use of normalizing flows (NFs) to address our density estimation problem. NFs belong to a class of generative models adept at modeling complex probability distributions. NFs employ a sequence of invertible and differentiable transformations $g_{\phi,k} = g_k \circ g_{k-1} \circ \cdots \circ g_1$, applied to a base distribution $p_Z(z)$, typically a simple distribution, e.g., standard Gaussian. These transformations progressively refine the distribution to accurately model the target data distribution $p_X(x)$ via the mapping $z = g_{\phi,k}(x)$. Consequently, the log-likelihood of the target distribution is formulated as:

$$\log p_X(x) = \log p_Z(z) + \sum_{i=1}^{k} \log \left| \det \frac{\partial g_i}{\partial g_{i-1}} \right|, \tag{2}$$

which facilitates the training of parameters ϕ via maximum likelihood. Once the flow is trained, density estimation can be performed by evaluating the right-hand side of (2).

Early contributions such as NICE (Dinh, Krueger, and Bengio 2014) streamline computations using non-linear additive coupling layers. Building on this, RealNVP (Dinh, Sohl-Dickstein, and Bengio 2017) utilizes affine coupling layers to achieve efficient and precise density estimation and sampling, offering improved expressiveness and flexibility compared to NICE. Masked Autoregressive Flow (MAF) (Papamakarios, Pavlakou, and Murray 2017) replaces these affine coupling layers with autoregressive ones, increasing expressiveness at the expense of single-pass sampling. More recently, Glow (Kingma and Dhariwal 2018) advances these concepts further by incorporating invertible 1x1 convolutions, offering greater flexibility within multi-scale architectures.

Markov Balance-based Imitation Learning

We next describe MBIL, our imitation learning algorithm. A key premise of our algorithm is that the demonstration trajectory data satisfies a balance equation involving the demonstration policy, the Markov decision process (MDP) transition density, and the induced Markov chain (MC). We can use this balance equation to guide the agent's learning. Using the balance equation requires estimating certain transition (conditional probability) density functions, which we obtain via the conditional normalizing flow method. We describe our approach in Algorithm 1 and present numerical evidence of its efficacy on several benchmark Classic Control and MuJoCo tasks.

The Markov Balance Equation. Consider a demonstration policy π_D that is used to take actions starting from an initial state s_0. Let $T(s'|s,a)$ denote the transition density function of the MDP. Note that π_D is a randomized stationary Markovian policy and $\pi_D(\cdot|s)$ is the probability distribution of actions at state s. Using policy π_D on the underlying MDP induces a Markov chain on the state space S whose transition density is denoted by $P(s'|s)$. This transition density satisfies the following equation which we refer to as the Markov balance equation: $P(s'|s) = \sum_a \pi_D(a|s)T(s'|s,a)$. Unfortunately, this approach involves a summation, which becomes problematic in continuous action domains, where the sum translates to an integral over actions. Therefore, we use the following alternative balance equation which involves the transition density of the induced Markov chain on the *state-action* space,

$$P_{\pi_D}(s', a'|s, a) = \pi_D(a'|s')T(s'|s, a). \tag{3}$$

The above balance equation is the basis of our IL approach. If we can estimate P_{π_D} and T in (3) (estimates denoted by $\hat{P}$ and $\hat{T}$ respectively), we can use the balance equation to guide agent's learning. In our approach, we consider a combination of a *policy-based* loss function that measures the discrepancy between the demonstrator's and

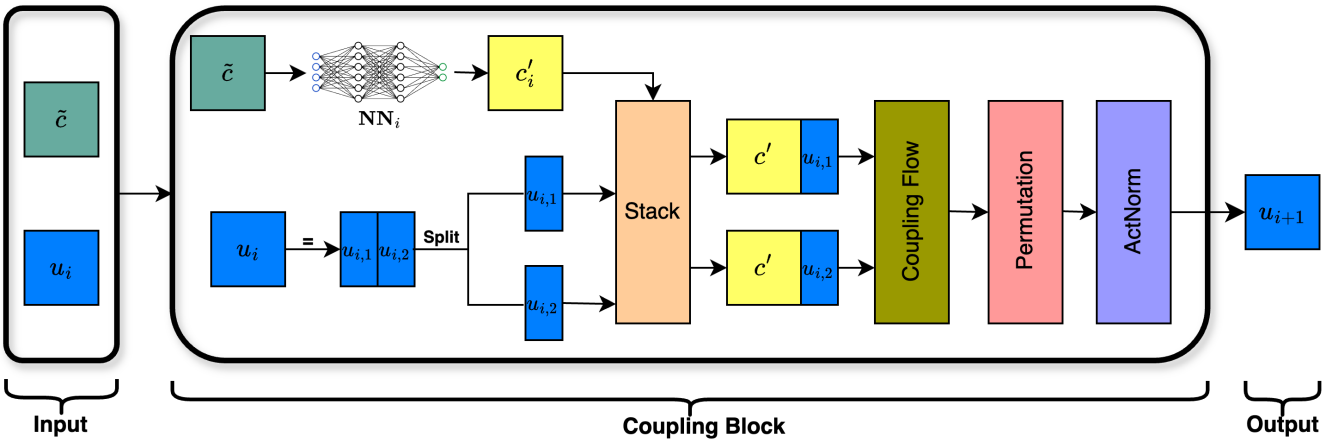


Figure 2: A high-level description of coupling block CB_i in conditional density estimation setup.

learner's policies and a *dynamics-based* loss function that quantifies the discrepancy between the two sides of the balance equation under the learner's policy.

We consider a class of policies parametrized by θ and formulate the following optimization problem:

$$\min_{\theta \in \Theta} J_{\mathrm{MBIL}}(\pi_\theta) := \alpha \underbrace{\mathbb{E}_{s,a,s',a' \sim D}[\mathrm{BALANCE}(s', a', s, a)]}_{\text{Dynamics Loss}} + \beta \underbrace{\mathbb{E}_{s,a \sim D}\left[\mathcal{L}\left(\pi_D(\cdot \mid s), \pi_\theta(\cdot \mid s)\right)\right]}_{\text{Policy Loss}} \tag{4}$$

Examples of policy loss function $\mathcal{L}(\cdot)$ include KL divergence between the policies (which amounts to maximizing the log-likelihood of the expert's actions in the states observed in the demonstration data) or the mean-squared error between the agent's and the expert's actions, among other possibilities. The $BALANCE$ function introduces a dynamics-based loss term designed to enforce the Markov Balance constraint and is defined as follows:

$$\mathrm{BALANCE}(s', a', s, a) = \left[\log \hat{P}(s', a'|s, a) - \log \pi_\theta(a'|s') - \log \hat{T}(s'|s, a)\right]^2.$$

Θ is a given parameter set. The parameters could be weights of a neural network, for example. α and β are parameters that represent the relative priorities of the dynamics-based term and the policy-based term, respectively.

Remarks. In Offline IL-compatible DICE methods like ValueDICE, the derivation of the off-policy objective assumes that the Markov chain induced by the learned policy is always ergodic and that the state-action distribution is stationary. However, these assumptions often fail during training due to early terminations from environment resets triggered by adverse states or fatal actions leading to termination bias (Sun et al. 2021). Algorithms such as IQ-Learn and SoftDICE address this by adding an indicator for absorbing states and modifying the Bellman operator, but this can introduce reward bias, as the value of absorbing states should be learned rather than assumed to be zero (Kostrikov et al. 2018). LS-IQ (Al-Hafez et al. 2023) improves upon IQ-Learn by correcting reward bias with a modified inverse Bellman operator but performs poorly in pure offline settings, especially on MuJoCo tasks. In contrast, our method avoids the off-policy derivation framework and employs the Markov Balance Equation instead. The rationale is straightforward: *every* transition captured in the dataset, including those leading to the terminal state, must adhere to this equation. As a result, we eliminate the biases that stationary distribution matching algorithms often face. This also eliminates the need for a discount factor γ (typically required in DICE methods and may influence their performance), and simplifies training by avoiding a complex alternating max-min optimization (typical in DICE methods) with a single optimization procedure.

Transition Density Estimation We now introduce methodology for estimation of the the two conditional densities P_{π_D} and T.

Estimating transition densities in continuous setting is challenging because each visited state would only appear once, and most states may never be visited in the dataset. This necessitates the use of more sophisticated conditional density estimation methods. These include non-parametric methods like Gaussian process conditional density estimation (Dutordoir et al. 2018) and Conditional Kernel Density Estimation (CKDE) (Li and Racine 2006); semi-parametric methods like least squares conditional density estimation (Sugiyama et al. 2010); and parametric approaches such as mixture density networks (MDN) (Bishop 1994) and normalizing flows (Dinh, Sohl-Dickstein, and Bengio 2017).

We use Normalizing Flows (NFs) because they outperform density estimation techniques such as CKDE and MDN. Unlike CKDE and MDN, which can struggle with high-dimensional data and require careful tuning of bandwidths or mixture components, NFs leverage a series of invertible transformations to model complex distributions with high flexibility and precision. This allows for exact likelihood computation, making NFs ideal for tasks requiring

Algorithm 1: Markov Balance-based Imitation Learning (MBIL)

Input: Expert dataset of trajectories $D = \{(s_i, a_i)\}_{i=1}^{n}$.
Output: θ^*.

1: Initialize policy parameters θ, expert MC parameters η, transition MDP parameters ψ.
2: Transform dataset D into (s, a, s', a') tuples, then store them in buffer B.
3: Train $P_\eta(s', a'|s, a)$ and $T_\psi(s'|s, a)$ by log-likelihood maximization, using the estimates provided by (6).
4: **for** $iter = 0, 1, \ldots$ **do**
5: Sample a batch b_{iter} of (s, a, s', a') tuples from B.
6: Obtain $\hat{P}_\eta, \hat{T}_\psi$ on b_{iter} using (6) on P_η and T_ψ respectively.
7: Calculate empirical estimate of the objective function in (4) using all $(s, a, s', a') \in b_{iter}$ as:

$$\sum_{(s,a,s',a')\in b_{iter}} \alpha\big[\log \hat{P}_\eta(s', a'|s, a) - \log \pi_\theta(a'|s') - \log \hat{T}_\psi(s'|s, a)\big]^2 + \sum_{(s,a)\in b_{iter}} \beta[\mathcal{L}\left(\pi_D(a \mid s), \pi_\theta(a \mid s)\right)] \quad (5)$$

8: Update the policy parameter θ using gradient update to minimize the calculated empirical estimate of the objective function
9: **end for**
10: **return** θ^*

precise probability evaluation, as in our setup. Additionally, NFs can capture complex, multimodal distributions without predefined parametric forms, offering a more expressive framework (Papamakarios et al. 2021). Furthermore, NFs with affine coupling flows are universal distribution approximators under suitable conditions (Draxler et al. 2024).

Given a conditioning input $c \in C$, the conditional density estimation of $x \in \mathcal{X}$, $p(x|c)$ using a Conditional Normalizing Flow (CNF) is given by (Ardizzone et al. 2019):

$$p_{X|C}(x|c) = p_Z\left(g_{\phi,k}(x, c)\right) \left| \frac{\partial g_{\phi,k}(x, c)}{\partial x} \right|,$$

where $\left| \frac{\partial g_{\phi,k}(x,c)}{\partial x} \right|$ is the Jacobian determinant of $g_{\phi,k}$ evaluated at x for the conditioning input c. Finally, the log-likelihood of x given c can be computed as:

$$\log p_{X|C}(x|c) = \log p_Z\left(g_{\phi,k}(x, c)\right) + \sum_{i=1}^{k} \log \left| \det \frac{\partial g_i}{\partial g_{i-1}} \right| \quad (6)$$

Figure 1 illustrates our conditional normalizing flow setup. The neural network NN_c processes the conditioning input c, mapping it to a latent space variable $\tilde{c}$, which is then fed into each coupling block of our NF configuration. The specifics of the i^{th} coupling block, for $i \in \{1, 2, \ldots, k\}$, are shown in Figure 2. Coupling block i includes a neural network NN_i that further transforms $\tilde{c}$ into c'_i before passing it to the coupling flow.

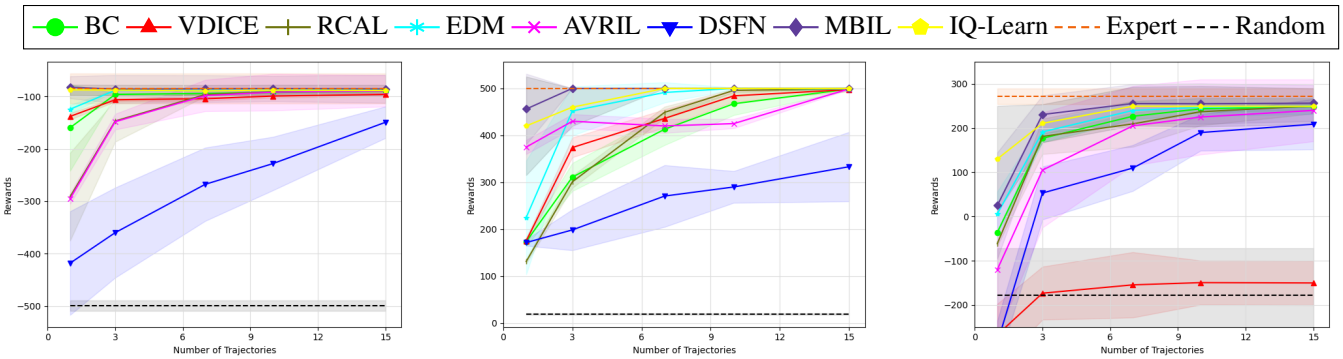
Experimental Results

Setup. We evaluate the empirical performance of our MBIL algorithm using the MuJoCo locomotion suite (Todorov, Erez, and Tassa 2012) and the classic control suite from OpenAI Gym (Brockman et al. 2016). MuJoCo tasks, with their varying difficulty levels, are a popular benchmark for IL in continuous action domains, while classic control environments like LunarLander are used to assess IL algorithms in discrete action domains. For constructing the demonstration dataset D, we use data from (Kostrikov, Nachum, and Tompson 2020b), where the Generative Adversarial Imitation Learning algorithm (Ho and Ermon 2016) was applied for MuJoCo tasks. For classic control tasks, we utilize pretrained and hyperparameter-optimized agents from the RL Baselines Zoo (Raffin 2020), employing a PPO agent for LunarLander-v2, a DQN agent for CartPole-v1, and an A2C agent for Acrobot-v1.

Implementation. A neural network (NN) with two hidden layers, each using the ReLU activation function, is used for representing the policy. For tasks with discrete actions (e.g., Classic Control tasks), the output layer has a dimension equal to the number of action dimensions and uses a softmax function to generate a probability distribution over actions given a state. In contrast, for MuJoCo tasks with continuous actions, the output consists of two separate layers: one for the mean and another for the standard deviations, each with a size equal to the action dimension. The policy is then modeled as a Gaussian distribution, with these parameters generated by the neural network. Training is performed with the Adam optimizer (Kingma and Ba 2015). Detailed implementation, including hyperparameters for MBIL and benchmark algorithms, are provided in the Appendix.

The transition dynamics models P_η and T_ψ are implemented using RealNVPs (Dinh, Sohl-Dickstein, and Bengio 2017). We employ the publicly available framework version 0.2 (Ardizzone et al. 2018-2022), utilizing their GLOWCouplingBlocks implementation. Detailed parameters for these models are provided in Table 1. A central aspect of our approach is performing imitation learning in the low data regime. To facilitate this, we introduce Gaussian noise as a regularizer for training the expert MC and transition MDP, which enhances training stability with limited data.

Results. When given sufficient demonstration data, all benchmarks can achieve performance comparable to optimized demonstration agents. Therefore, we test the algorithms' ability to handle limited data. Several recent offline



(a) Acrobot (b) CartPole (c) LunarLander

Figure 3: Average rewards achieved by benchmark IRL/IL/AIL and MBIL policies during real-time deployment plotted against the number of trajectories included in demonstration dataset D (higher values indicate better performance).

	$P_\eta(s', a'\|s, a)$	$T_\psi(s'\|s, a)$
k Flow Blocks	16	16
Exponent Clamping	4	4
NN_c hidden layers	2	2
NN_c hidden nodes	$64, 64$	$64, 64$
NN_c activation function	LeakyReLU	LeakyReLU
NN_i layers	1	1
NN_i nodes	32	32
CB_i hidden layers	2	2
CB_i hidden nodes	$64, 64$	$64, 64$
CB_i activation function	LeakyReLU	LeakyReLU

Table 1: Normalizing Flow Parameters

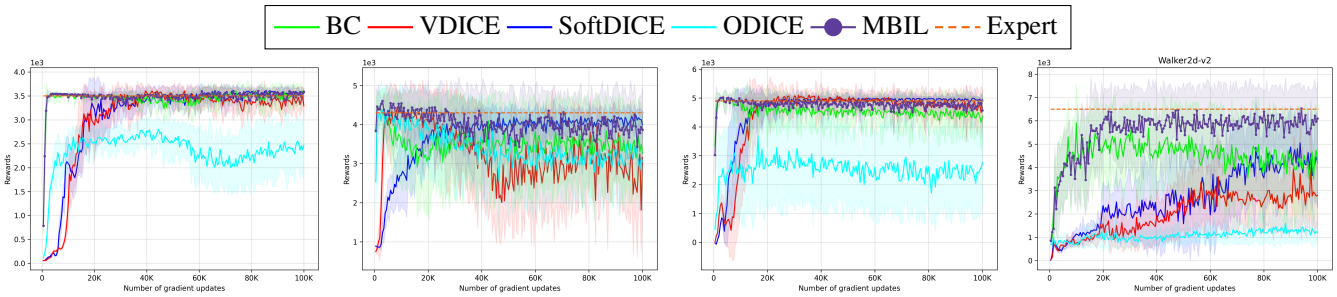
IRL/IL methods, such as CLARE (Yue et al. 2023b), DemoDICE (Kim et al. 2022), and MILO (Chang et al. 2021), depend on additional diverse data, while others, like OPOLO (Zhu et al. 2020), allow for interactions with the environment within their off-policy frameworks. Given that our setting strictly considers only offline expert data without access to additional interactions or supplementary datasets, we exclude methods that rely on such resources from our comparisons to ensure a fair evaluation.

Classic Control Tasks. Inspired by (Jarrett, Bica, and van der Schaar 2020b), we trained algorithms until convergence on datasets of 1, 3, 7, 10, or 15 trajectories sampled from a pool of 1000 expert trajectories and recorded the average scores over 300 episodes for each algorithm, repeating this process 10 times with varied initializations and trajectories. We compare our MBIL algorithm (Algorithm 1) against various offline IRL/IL/AIL baselines, including BC, ValueDICE (VDICE), RCAL, EDM, AVRIL, DSFN, and the state-of-the-art model-free offline IRL algorithm IQ-Learn. We use log-likelihood maximization based BC as $\mathcal{L}(\cdot)$ in (5) in our Classic Control experiments.

Figure 3 illustrates the average rewards obtained by each algorithm as the demonstration dataset size increases in the Acrobot, CartPole, and LunarLander environments. The results highlight MBIL's ability to learn effective policies, consistently outperforming the baseline algorithms, particularly when data is limited. MBIL achieves near-expert-level performance in these environments with at most three trajectories and, remarkably, can reach near-expert performance in the CartPole environment with just a single trajectory. Notably, MBIL generally outperforms all baseline algorithms across these environments, with IQ-Learn showing performance closest to MBIL. Additionally, off-policy adaptations of online algorithms, such as VDICE and DSFN, do not maintain the same level of consistent performance as their inherently online counterparts. This highlights the need for more than just adopting online algorithms in offline scenarios. Moreover, the difficulty in estimating the expectation of an exponential distribution may contribute to VDICE's relative underperformance compared to MBIL and other methods.

MuJoCo Tasks. Following the methodology outlined in (Kostrikov, Nachum, and Tompson 2020a; Sun et al. 2021), we use a single demonstration trajectory, validate performance every 500 training iterations across 10 episodes, and report means and standard deviations from 5 random seeds. We compare MBIL (Algorithm 1) against strong baselines for locomotion tasks, including BC, ValueDICE, SoftDICE, and ODICE. IQ-Learn is excluded due to its poor performance in high-dimensional MuJoCo tasks with continuous action spaces in the strictly offline setting, as it suffers from exploding Q-functions (Al-Hafez et al. 2023). We also exclude EDM and AVRIL because they are incompatible with continuous action spaces. Notably, while baselines like SoftDICE and ValueDICE, which use log-likelihood maximization for BC and show its poor performance, MSE-based BC, as demonstrated by (Li et al. 2022), performs well in MuJoCo tasks and is therefore used for BC in our comparison. Furthermore, we employ this mean-squared error (MSE) BC loss as $\mathcal{L}(\cdot)$ in (5) in our MuJoCo experiments.

Figure 4 compares the performance of MBIL with strong baselines on MuJoCo tasks in the offline IL literature. MBIL consistently achieves near-expert performance on all tasks with just a single trajectory in the dataset. Among the base-



(a) Hopper (b) Ant (c) HalfCheetah (d) Walker2d

Figure 4: Average rewards achieved by benchmark and MBIL policies against the number of gradient updates on MuJoCo tasks with 1 expert trajectory (higher values indicate better performance).

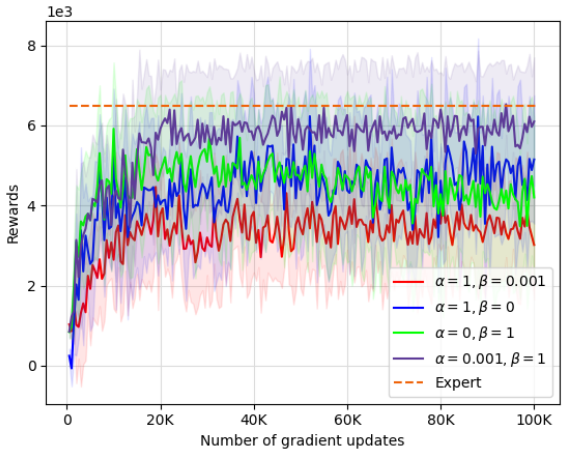


Figure 5: Ablation study: Average rewards achieved by policies trained using different combinations of α and β values in the MBIL objective against the number of gradient updates on MuJoCo tasks with 1 expert trajectory on Walker2d environment (higher values indicate better performance).

lines, BC and SoftDICE show strong performance. However, BC's performance deteriorates in the Ant and Walker2d environments as the number of gradient updates increases, despite using orthogonal regularization (Brock, Donahue, and Simonyan 2018), a trend also noted in (Li et al. 2022). In contrast, MBIL maintains robust performance throughout the learning process by integrating dynamics loss with the policy loss. SoftDICE encounters difficulties in the Walker2d environment, where absorbing states are more common than Ant and HalfCheetah. Its assumption of a value of 0 for these states can introduce reward bias, hindering performance since the values of these states also need to be learned (Al-Hafez et al. 2023). The performance decline of ValueDICE is likely due to biased gradient updates in its learning process (Sun et al. 2021). ODICE attempts to address the issue of conflicting gradients in DICE methods by incorporating orthogonal gradient updates, but this approach may not have been effective for these tasks. Overall, the results highlight our framework's effectiveness, which combines policy loss and dynamics loss to facilitate effective learning.

Ablation study. In our framework, the objective function comprises two components: 1) the policy loss and 2) the dynamics loss. We investigate how each component affects the performance of our algorithm on the Walker2d task by varying the α and β coefficients in (4), with results shown in Figure 5. Specifically, we evaluate the following configurations: $\alpha = 1, \beta = 0$ (dynamics loss only), $\alpha = 0, \beta = 1$ (policy loss only, representing BC in this experiment), $\alpha = 1, \beta = 0.001$ (higher relative weight on dynamics loss), and $\alpha = 0.001, \beta = 1$ (higher relative weight on policy loss). Our findings reveal that while both policy loss alone and dynamics loss alone perform similarly, with dynamics loss performing slightly better as gradient updates progress, it is the combination of policy loss and dynamics loss, with a higher weight assigned to the policy loss, that enables us to achieve near-expert performance using only one expert trajectory.

Conclusion

In this paper, we address the strictly batch imitation learning problem in a continuous state and action space setting, i.e., we do not assume access to any further interactions or supplementary data. We present a Markov balance-based imitation learning algorithm that combines it with supervised learning-based behavior cloning while using conditional normalizing flows for density estimation. Our method does not rely on reward model estimation and avoids stationary distribution matching which suffer from termination and reward bias. From numerical experiments, we see that our proposed algorithm does as well or much better than any state-of-the-art algorithm across a variety of Classic Control and MuJoCo environments. In fact, as can be seen in further results presented in the Appendix, it handles the distribution shift issue more effectively than other strictly batch imitation learning algorithms. Future research could benefit from deriving both asymptotic and non-asymptotic sample complexity bounds, which are scarce in current literature, and from exploring extensions to handle suboptimal data cases.

References

Abbeel, P.; and Ng, A. Y. 2004. Apprenticeship Learning via Inverse Reinforcement Learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*.

Al-Hafez, F.; Tateo, D.; Arenz, O.; Zhao, G.; and Peters, J. 2023. LS-IQ: Implicit Reward Regularization for Inverse Reinforcement Learning. In *The Eleventh International Conference on Learning Representations*.

Ardizzone, L.; Bungert, T.; Draxler, F.; Köthe, U.; Kruse, J.; Schmier, R.; and Sorrenson, P. 2018-2022. Framework for Easily Invertible Architectures (FrEIA).

Ardizzone, L.; Lüth, C.; Kruse, J.; Rother, C.; and Köthe, U. 2019. Guided image generation with conditional invertible neural networks. *arXiv preprint arXiv:1907.02392*.

Arora, S.; and Doshi, P. 2021. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297: 103500.

Baheri, A. 2023. Understanding Reward Ambiguity Through Optimal Transport Theory in Inverse Reinforcement Learning. In *NeurIPS 2023 Workshop Optimal Transport and Machine Learning*.

Barnes, M.; Abueg, M.; Lange, O. F.; Deeds, M.; Trader, J.; Molitor, D.; Wulfmeier, M.; and O'Banion, S. 2024. Massively Scalable Inverse Reinforcement Learning in Google Maps. In *The Twelfth International Conference on Learning Representations*.

Bishop, C. 1994. Mixture density networks. Technical report, Aston University.

Brock, A.; Donahue, J.; and Simonyan, K. 2018. Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Chan, A. J.; and van der Schaar, M. 2021a. Scalable Bayesian Inverse Reinforcement Learning. In *International Conference on Learning Representations*.

Chan, A. J.; and van der Schaar, M. 2021b. Scalable Bayesian Inverse Reinforcement Learning. https://github.com/XanderJC/scalable-birl.

Chang, J. D.; Uehara, M.; Sreenivas, D.; Kidambi, R.; and Sun, W. 2021. Mitigating covariate shift in imitation learning via offline data without great coverage. *arXiv preprint arXiv:2106.03207*.

Dinh, L.; Krueger, D.; and Bengio, Y. 2014. Nice: Nonlinear independent components estimation. *arXiv preprint arXiv:1410.8516*.

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S. 2017. Density estimation using Real NVP. In *International Conference on Learning Representations*.

Draxler, F.; Wahl, S.; Schnoerr, C.; and Koethe, U. 2024. On the Universality of Volume-Preserving and Coupling-Based Normalizing Flows. In Salakhutdinov, R.; Kolter, Z.; Heller, K.; Weller, A.; Oliver, N.; Scarlett, J.; and Berkenkamp, F., eds., *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 11613–11641. PMLR.

Dutordoir, V.; Salimbeni, H.; Hensman, J.; and Deisenroth, M. 2018. Gaussian process conditional density estimation. *Advances in neural information processing systems*, 31.

Fu, J.; Luo, K.; and Levine, S. 2018. Learning Robust Rewards with Adverserial Inverse Reinforcement Learning. In *International Conference on Learning Representations*.

Garg, D.; Chakraborty, S.; Cundy, C.; Song, J.; and Ermon, S. 2021a. Iq-learn: Inverse soft-q learning for imitation. *Advances in Neural Information Processing Systems*, 34: 4028–4039.

Garg, D.; Chakraborty, S.; Cundy, C.; Song, J.; and Ermon, S. 2021b. Iq-learn: Inverse soft-q learning for imitation. https://github.com/Div99/IQ-Learn.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30.

Ho, J.; and Ermon, S. 2016. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29.

Jarrett, D.; Bica, I.; and van der Schaar, M. 2020a. Strictly Batch imitation learning by Batch Imitation Learning by Energy-based Distribution Matching. https://github.com/vanderschaarlab/mlforhealthlabpub/tree/main/alg/edm.

Jarrett, D.; Bica, I.; and van der Schaar, M. 2020b. Strictly batch imitation learning by energy-based distribution matching. *Advances in Neural Information Processing Systems*, 33: 7354–7365.

Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Ronneberger, O.; Tunyasuvunakool, K.; Bates, R.; Žídek, A.; Potapenko, A.; et al. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873): 583–589.

Ke, L.; Choudhury, S.; Barnes, M.; Sun, W.; Lee, G.; and Srinivasa, S. 2021. Imitation learning as f-divergence minimization. In *Algorithmic Foundations of Robotics XIV: Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics 14*, 313–329. Springer.

Kim, G.-H.; Seo, S.; Lee, J.; Jeon, W.; Hwang, H.; Yang, H.; and Kim, K.-E. 2022. Demodice: Offline imitation learning with supplementary imperfect demonstrations. In *International Conference on Learning Representations*.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations*.

Kingma, D. P.; and Dhariwal, P. 2018. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31.

Klein, E.; Geist, M.; and Pietquin, O. 2011. Batch, off-policy and model-free apprenticeship learning. In *European Workshop on Reinforcement Learning*, 285–296. Springer.

Klein, E.; Geist, M.; Piot, B.; and Pietquin, O. 2012. Inverse reinforcement learning through structured classification. *Advances in neural information processing systems*, 25.

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

Kostrikov, I.; Nachum, O.; and Tompson, J. 2020a. Imitation Learning via Off-Policy Distribution Matching. In *International Conference on Learning Representations*.

Kostrikov, I.; Nachum, O.; and Tompson, J. 2020b. Imitation Learning via Off-Policy Distribution Matching. https://github.com/google-research/google-research/tree/master/value_dice.

Lee, D.; Srinivasan, S.; and Doshi-Velez, F. 2019a. Truly Batch Apprenticeship Learning with Deep Successor Features. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 5909–5915.

Lee, D.; Srinivasan, S.; and Doshi-Velez, F. 2019b. Truly Batch Apprenticeship Learning with Deep Successor Features. https://github.com/dtak/batch-apprenticeship-learning.

Li, Q.; and Racine, J. S. 2006. *Nonparametric Econometrics: Theory and Practice*. Number 8355 in Economics Books. Princeton University Press.

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

Liu, M.; He, T.; Xu, M.; and Zhang, W. 2020. Energy-based imitation learning. *arXiv preprint arXiv:2004.09395*.

Ma, Y.; Shen, A.; Jayaraman, D.; and Bastani, O. 2022. Versatile offline imitation from observations and examples via regularized state-occupancy matching. In *International Conference on Machine Learning*, 14639–14663. PMLR.

Mao, L.; Xu, H.; Zhang, W.; and Zhan, X. 2024a. Odice: Revealing the mystery of distribution correction estimation via orthogonal-gradient update. *arXiv preprint arXiv:2402.00348*.

Mao, L.; Xu, H.; Zhang, W.; and Zhan, X. 2024b. ODICE: Revealing the Mystery of Distribution Correction Estimation via Orthogonal-gradient Update. https://github.com/maoliyuan/ODICE-Pytorch.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Ng, A. Y.; and Russell, S. J. 2000. Algorithms for Inverse Reinforcement Learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 663–670.

Papamakarios, G.; Nalisnick, E.; Rezende, D. J.; Mohamed, S.; and Lakshminarayanan, B. 2021. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57): 1–64.

Papamakarios, G.; Pavlakou, T.; and Murray, I. 2017. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30.

Piot, B.; Geist, M.; and Pietquin, O. 2014. Boosted and Reward-Regularized Classification for Apprenticeship Learning. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, 1249–1256.

Piot, B.; Geist, M.; and Pietquin, O. 2016. Bridging the gap between imitation learning and inverse reinforcement learning. *IEEE transactions on neural networks and learning systems*, 28(8): 1814–1826.

Pomerleau, D. A. 1988. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1: 305–313.

Raffin, A. 2020. RL Baselines3 Zoo. https://github.com/DLR-RM/rl-baselines3-zoo.

Reddy, S.; Dragan, A. D.; and Levine, S. 2019. SQIL: imitation learning via regularized behavioral cloning. *arXiv preprint arXiv:1905.11108*, 2(5).

Ross, S.; and Bagnell, D. 2010. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 661–668. JMLR Workshop and Conference Proceedings.

Ross, S.; Gordon, G.; and Bagnell, D. 2011. 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*, volume 15, 627–635.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587): 484–489.

Silver, D.; Singh, S.; Precup, D.; and Sutton, R. S. 2021. Reward is enough. *Artificial Intelligence*, 299: 103535.

Sugiyama, M.; Takeuchi, I.; Suzuki, T.; Kanamori, T.; Hachiya, H.; and Okanohara, D. 2010. Least-squares conditional density estimation. *IEICE Transactions on Information and Systems*, 93(3): 583–594.

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

Swamy, G.; Choudhury, S.; Bagnell, J. A.; and Wu, Z. S. 2021. A Critique of Strictly Batch Imitation Learning. *arXiv preprint arXiv:2110.02063*.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, 5026–5033. IEEE.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Wang, R.; Ciliberto, C.; Amadori, P. V.; and Demiris, Y. 2019. Random expert distillation: Imitation learning via expert policy support estimation. In *International Conference on Machine Learning*, 6536–6544. PMLR.

Xu, H.; Zhan, X.; Yin, H.; and Qin, H. 2022. Discriminator-weighted offline imitation learning from suboptimal demonstrations. In *International Conference on Machine Learning*, 24725–24742. PMLR.

Yue, S.; Wang, G.; Shao, W.; Zhang, Z.; Lin, S.; Ren, J.; and Zhang, J. 2023a. CLARE: Conservative Model-Based Reward Learning for Offline Inverse Reinforcement Learning. In *The Eleventh International Conference on Learning Representations*.

Yue, S.; Wang, G.; Shao, W.; Zhang, Z.; Lin, S.; Ren, J.; and Zhang, J. 2023b. CLARE: Conservative Model-Based Reward Learning for Offline Inverse Reinforcement Learning. In *The Eleventh International Conference on Learning Representations*.

Yue, Y.; and Le, H. M. 2018. Imitation learning (Tutorial). *International Conference on Machine Learning (ICML)*.

Zeng, S.; Li, C.; Garcia, A.; and Hong, M. 2023. Understanding Expertise through Demonstrations: A Maximum Likelihood Framework for Offline Inverse Reinforcement Learning. *arXiv preprint arXiv:2302.07457*.

Zhu, Z.; Lin, K.; Dai, B.; and Zhou, J. 2020. Off-policy imitation learning from observations. *Advances in neural information processing systems*, 33: 12402–12413.

Ziebart, B. D.; Maas, A.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum Entropy Inverse Reinforcement Learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, 1433–1438. AAAI Press.

Appendix

Hyperparameters for MBIL

In the Classic Control tasks, where actions are discrete, the policy is represented by a neural network with two layers, each containing 256 nodes and using ReLU() activation. A softmax function is applied to the output layer to produce a distribution over actions for a given state. In contrast, for MuJoCo environments with continuous actions, the policy is modeled as a Gaussian distribution, with the mean and covariance parameters determined by a neural network. This network features two hidden layers, each with 256 nodes and ReLU() activation functions. The output consists of two separate layers: one with a size equal to the action dimension for the means, and another with the same size for the standard deviations of each action dimension. The diagonal covariance matrix is then constructed using these standard deviations.

We use a learning rate of $5e-4$ for policy training and a batch size of 512. Additionally, operating in a low-data regime, we employ orthogonal regularization and weight decay with a coefficient of $1e-4$ to mitigate overfitting. Our experiments in Figures 3 and 4 uses $\alpha = 1$ and $\beta = 0.001$.

Benchmark Algorithms and Hyperparameters

We evaluate the performance of our MBIL algorithm (Algorithm 1) by comparing it with a range of offline IRL, IL, and AIL baselines, including several recent state-of-the-art methods. For the Classic Control tasks, our comparison includes Behavioral Cloning (BC), a method inherently suited for offline learning; ValueDICE (VDICE), a sample-efficient approach designed for offline scenarios by eliminating replay regularization; Reward-Regularized Classification (RCAL), which introduces a sparsity-based penalty on inferred rewards to leverage dynamics information; Energy-Based Distribution Matching (EDM), which captures the expert's state occupancy patterns through an energy-based model; AVRIL, a recent model-free offline IRL technique using a variational approach to learn an approximate posterior distribution over rewards and policies; and Deep Successor Feature Network (DSFN), which adapts max-margin IRL with deep network architecture and least-squares temporal-difference learning for reward and policy outputs. Additionally, we compare MBIL to IQ-Learn, a state-of-the-art model-free offline IRL algorithm.

For the MuJoCo tasks, we evaluate baselines designed for continuous action domains capable of working in strictly offline imitation learning settings. We assess MBIL against Behavioral Cloning (BC), VDICE, SoftDICE (a stationary distribution matching method using the Earth Mover Distance metric), and ODICE (a stationary distribution matching algorithm utilizing orthogonal gradient updates).

Implementation Details. For VDICE, we utilized the open-sourced code available at (Kostrikov, Nachum, and Tompson 2020b), noting that offline learning was achieved by setting the "replay regularization" coefficient to zero. We implemented EDM using the source code provided at (Jarrett, Bica, and van der Schaar 2020a). The primary distinction between BC and EDM arises from the incorporation of L_ρ, an occupancy loss introduced in the EDM work, while deriving the RCAL loss involves a straightforward inversion of the Bellman equation. The source codes for AVRIL and DSFN were obtained from (Chan and van der Schaar 2021b) and (Lee, Srinivasan, and Doshi-Velez 2019b), respectively. IQ-Learn was run using the code available at (Garg et al. 2021b). For SoftDICE, we sourced code from the authors of that paper, and for ODICE, we utilized the code from (Mao et al. 2024b). All simulations are run on a linux machine with a maximum capacity of 124 GB of RAM. Code for MBIL and ODICE uses pytorch 2.4.0 whereas for ValueDICE and SoftDICE, it uses Tensorflow 2.13.1. Other packages related details including their version are available in the code folder provided with the supplementary file.

RCAL This introduces an expansion of the policy loss by incorporating an extra sparsity-driven loss concerning the inferred rewards $\hat{R}(s,a)$, defined as $f_\theta(s)[a] - \gamma softmax_{a'} f_\theta(s')[a']$, acquired through the inversion of the Bellman equation. The policy network employed is the fully-connected type detailed previously. The coefficient for sparsity-based regularization is designated as 1e-2.

	Algorithm		
Trajectories (τ)	EDM	IQ-Learn	MBIL
$\tau = 1$	-406.37 ± 340.03	5.12 ± 190.12	-0.81 ± 169.56
$\tau = 3$	-299.21 ± 184.60	159.726 ± 120.18	201.12 ± 99.15
$\tau = 7$	-240.38 ± 202.53	205.96 ± 99.68	235.11 ± 77.34
$\tau = 10$	-38.36 ± 130.54	232.28 ± 87.66	244.26 ± 55.82
$\tau = 15$	89.32 ± 101.45	242.32 ± 77.59	247.29 ± 58.11

Table 2: Performance of EDM, IQ-Learn, and MBIL in the presence of initial distribution shift in LunarLander environment (higher values indicate better performance)

VDICE We employ the publicly accessible source code from https://github.com/google-research/google-research/tree/master/value_dice. To accommodate discrete action spaces, we incorporate a Gumbel-softmax parameterization for the final layer of the actor network. Both the actor and discriminator architecture encompass two fully connected hidden layers, each composed of 64 units activated by ReLU functions. Consistent with the original framework, the output is merged with the action and propagated through two additional hidden layers, each containing 64 units. In addition, we set the "replay regularization" coefficient at zero for strict batch learning. Furthermore, the actor network is subjected to "orthogonal regularization" with a coefficient of 1e-4. The actor network's learning rate is set at 1e-5, while the discriminator operates with a learning rate of 1e-3.

EDM We utilize the code accessible at https://github.com/vanderschaarlab/mlforhealthlabpub/tree/main/alg/edm. Particularly for EDM, the hyperparameters for joint Energy-Based Model (EBM) training are adopted from https://github.com/wgrathwohl/JEM. These parameters include a noise coefficient of $\sigma = 0.01$, a buffer size of $\kappa = 10000$, a length of $\iota = 20$, and a reinitialization value of $\delta = 0.05$. These predefined configurations align effectively with the SGLD (Stochastic Gradient Langevin Dynamics) step size of $\alpha = 0.01$.

BC The sole distinction between Behavior Cloning (BC) and EDM lies in the inclusion of L_ρ, which is omitted in the implementation of BC. The policy network remains consistent with the description provided earlier.

AVRIL We use the code available at https://github.com/XanderJC/scalable-birl. The policy network remains consistent with the description provided earlier. γ, used while computing the TD error, equals 1. We used the default parameters provided in their GitHub repository.

DSFN We adopt the source code accessible at https://github.com/dtak/batch-apprenticeship-learning. We utilize a "warm-start" policy network consisting of two shared layers with dimensions 128 and 64, employing tanh activation. The hidden layer with a size of 64 serves as the feature map within the IRL algorithm. Each multitask head within the warm-start policy network features a hidden layer comprising 128 units and is activated by tanh. The Deep Q-Network (DQN), utilized for learning the optimal policy based on a set of reward weights, comprises two fully-connected layers, each containing 64 units. Similarly, the DSFN, employed for estimating feature expectations, comprises two hidden fully-connected layers, each containing 64 units. Across all environments, the warm-start policy network undergoes training for $50,000$ steps, employing the Adam optimizer with a learning rate of 3e-4 and a batch size of 64. The DQN network is trained for $30,000$ steps, using a learning rate of 3e-4 and a batch size of 64 (with the Adam optimizer). Lastly, the DSFN network is trained for $50,000$ iterations, utilizing a learning rate of 3e-4 and a batch size of 32 (with the Adam optimizer).

IQ-LEARN We use the code available at https://github.com/Div99/IQ-Learn. The policy network remains consistent with the description provided earlier. As highlighted in their implementation, we use a batch size of 32 and Q-network learning rate of 1e-4 with entropy coefficient of 0.01.

SoftDICE We obtained the code directly from the authors of this work. The policy network (actor) is structured as an MLP with two hidden layers, each containing 256 units, and employs orthogonal initialization. We use the Adam optimizer with a 10^{-3} learning rate for the Lipschitz continuous function and 10^{-5} for the policy π. To enforce Lipschitz continuity, gradient penalty is applied as outlined in (Gulrajani et al. 2017) and regularize the actor-network using orthogonal regularization (Brock, Donahue, and Simonyan 2018). The entropy coefficient is set to 0.01 for optimal performance across most tasks, except for Ant, where it is set to 0.2.

ODICE We use the code available at https://github.com/maoliyuan/ODICE-Pytorch and use the orthogonal true gradient updates. We adopt the default parameter values from their work: η, which controls the strength of the projected backward gradient against the forward gradient, is set to 1.0, and λ, which balances the current value of the state with the Bellman residual, is set to 0.4. The network is a $3-$layer MLP with 256 hidden units, and we use the Adam optimizer with a learning rate of 10^{-4} for both the value function V and the policy π across all tasks. Additionally, we employ a target network with a soft update weight of 5×10^{-3} for V.

Discussion on Distribution Shift

In this work, we have discussed the task of imitation learning in a strictly batch setting. Specifically, we assumed that

only expert data was available, with no possibility for further interaction with the environment. Another line of research in imitation learning aims to incentivize the imitating policy to remain within the distribution of states encountered in expert demonstrations. This research typically follows two approaches. The first approach assumes access to additional data from a behavioral policy (which may be sub-optimal) along with the expert data. This additional data is used to provide coverage, as expert data is generally narrow. Examples of this approach include methods like CLARE (Yue et al. 2023b) and MILO (Chang et al. 2021). The second approach involves techniques such as assigning a unit reward to all demonstrated actions in demonstrated states and zero otherwise (Reddy, Dragan, and Levine 2019), such as random expert distillation (Wang et al. 2019). Generally, these methods follow a "two-step" formula: first, a surrogate reward function is derived or defined; second, this reward function is optimized through environment interactions, making these techniques inherently online, rendering it inapplicable in our strictly batch setting (Liu et al. 2020).

Nevertheless, we investigated the effects of an initial distribution shift in the LunarLander-v2 environment, drawing inspiration from the approach in (Garg et al. 2021a). Typically, the agent starts in a small area at the center-top of the screen. However, we modified the environment so that the agent begins near the top-left corner instead. Using expert data from the standard, unmodified environment, we aimed to determine if the agent could still successfully learn to land the lunar module despite the shift in its initial conditions during testing. For comparisons, we consider EDM and IQ-Learn algorithms as these methods don't rely on additional data or further interactions with the environment, thus utilizing the same setup as ours. The findings are reported in Table 2. As anticipated, all algorithms perform poorly when the available data is very scarce. However, as the amount of data gradually increases, the MBIL agent demonstrates the ability to effectively land the lunar lander despite the initial distribution shift, even with limited training data (approximately 10 trajectories). Additionally, MBIL outperforms baseline algorithms like EDM and IQ-Learn under these conditions. Our experimental results suggest that our algorithm handles the distribution shift problem more effectively than the other baseline algorithms in the same setup.