Applied Intelligence
https://doi.org/10.1007/s10489-023-05062-w

Imitation learning by state-only distribution matching

Damian Boborzi[1] · Christoph-Nikolas Straehle[2] · Jens S. Buchner[3] · Lars Mikelsons[1]

Accepted: 28 September 2023

Abstract
Imitation Learning from observation describes policy learning in a similar way to human learning. An agent's policy is trained by observing an expert performing a task. Although many state-only imitation learning approaches are based on adversarial imitation learning, one main drawback is that adversarial training is often unstable and lacks a reliable convergence estimator. If the true environment reward is unknown and cannot be used to select the best-performing model, this can result in bad real-world policy performance. We propose a non-adversarial learning-from-observations approach, together with an interpretable convergence and performance metric. Our training objective minimizes the Kulback-Leibler divergence (KLD) between the policy and expert state transition trajectories which can be optimized in a non-adversarial fashion. Such methods demonstrate improved robustness when learned density models guide the optimization. We further improve the sample efficiency by rewriting the KLD minimization as the Soft Actor Critic objective based on a modified reward using additional density models that estimate the environment's forward and backward dynamics. Finally, we evaluate the effectiveness of our approach on well-known continuous control environments and show state-of-the-art performance while having a reliable performance estimator compared to several recent learning-from-observation methods.

Keywords Imitation learning · State-only · Normalizing flows · Reinforcement learning · Learning from observations

1 Introduction

Imitation learning (IL) describes methods that learn optimal behavior that is represented by a collection of expert demonstrations. In standard reinforcement learning (RL), the agent is trained on environment feedback using a reward signal. IL can alleviate the problem of designing effective reward functions using demonstrations. This is particularly useful for tasks where demonstrations are more accessible than designing a reward function. One popular example is to train traffic agents in a simulation to mimic real-world road users [1].

Learning-from-demonstrations (LfD) describes IL approaches that require state-action pairs from expert demonstrations [2]. Although actions can guide policy learning, it might be very costly or even impossible to collect actions alongside state demonstrations in many real-world setups. For example, when expert demonstrations are available as video recordings without additional sensor signals. One example of such a setup is training traffic agents in a simulation, where the expert data contains traffic recordings in bird's eye view [1]. No direct information on the vehicle physics, throttle, and steering angle is available. Another example is teaching a robot to pick, move, and place objects based on human demonstrations [3]. In such scenarios, actions have to be estimated based on sometimes incomplete information to train an agent to imitate the observed behavior.

Alternatively, learning-from-observations (LfO) performs state-only IL and trains an agent without actions being available in the expert dataset [4]. Although LfO is a more challenging task than LfD, it can be more practical in case of incomplete data sources. Like LfD, distribution matching based on an adversarial setup is commonly used in LfO [5]. In adversarial imitation learning (AIL), a policy is trained using an adversarial discriminator, which is used to estimate a reward that guides policy training. AIL methods obtain better-performing agents than supervised methods like behavioral cloning (BC) using less data. However, adversar-

Damian Boborzi and Christoph-Nikolas Straehle have contributed equally to this work

✉ Damian Boborzi
damian.boborzi@uni-a.de

1 Augsburg University, Augsburg, Germany

2 Bosch Center for Artificial Intelligence, Renningen, Germany

3 Bosch GmbH, Stuttgart, Germany

ial training often has stability issues [6] and, under some conditions, is not guaranteed to converge [7]. One possibility to improve the stability of generative adversarial networks is to use normalization layers like spectral normalization [6]. Jin et al. [7] give further insights into improving the stability of minimax optimization which is used in adversarial training. Additionally, estimating the performance of a trained policy without access to the environment reward can be very challenging. Although the duality gap [8, 9] is a convergence metric suited for GAN based methods, it is difficult to use in the AIL setup since it relies on the gradient of the generator for an optimization process. In the AIL setup, the generator consists of the policy and the environment and therefore the gradient is difficult to estimate with black box environments. As an alternative for AIL setups, the predicted reward (discriminator output) or the policy loss can be used to estimate the performance. We evaluate this approach empirically in Section 5.

To address the limitations of AIL, we propose a state-only distribution matching method that learns a policy in a non-adversarial way. We optimize the matching between the actionless policy and expert trajectories by minimizing the Kulback-Leibler divergence (KLD) of the conditional state transition distribution of the policy and the expert for all time steps. We estimate the expert state transition distribution using normalizing flows, which can be trained offline using the expert dataset. Thus, stability issues arising from the min-max adversarial optimization in AIL methods can be avoided. This objective is similar to FORM [10], which was shown to be more stable in the presence of task-irrelevant features.

Although maximum entropy RL methods [11] can improve policy training by increasing the exploration of the agent, they also add a bias if being used to minimize the proposed KLD. To match the transition distributions of the policy and the expert exactly, the state-next-state distribution of the policy is expanded into policy entropy, forward dynamics and the inverse action model of the environment. It has been shown that such dynamic models can improve the convergence [12] and are well suited to infer actions not available in the dataset [13]. We model these distributions using normalizing flow models which have been demonstrated to perform very well in learning complex probability distributions [14]. Combining all estimates results in an interpretable reward that can be used together with standard maximum entropy RL methods [15]. The optimization based on the KLD provides a reliable convergence metric of the training and a good estimator for policy performance.

As contributions, we derive SOIL-TDM (State Only Imitation Learning by Trajectory Distribution Matching), a non-adversarial LfO method which minimizes the KLD between the conditional state transition distributions of the policy and the expert using maximum entropy RL. We show the convergence of the proposed method using off-policy samples from a replay buffer. We develop a practical algorithm based on the SOIL-TDM objective and demonstrate its effectiveness in measuring its convergence compared to several other state-of-the-art methods. Empirically we compared our method to the recent state-of-the-art IL approaches OPOLO [12], F-IRL [16], and FORM [10] in complex continuous control environments. We demonstrate that our method is superior especially if the selection of the best policy cannot be based on the true environment reward signal. This is a setting that more closely resembles real-world applications in autonomous driving or robotics where it is difficult to define a reward function [3].

2 Background

In this work, we want to train a stochastic policy function $\pi_\theta(a_i|s_i)$ in continuous action spaces with parameters θ in a sequential decision making task considering finite-horizon environments[1].

The problem is modeled as a Markov Decision Process (MDP), which is described by the tuple (S, A, p, r) with the continuous state spaces S and action spaces A. The transition probability is described by $p(s_{i+1}|s_i, a_i)$ and the bounded reward function by $r(s_i, a_i)$. At every time step i the agent interacts with its environment by observing a state s_i and taking action a_i. This results in a new state s_{i+1} and a reward signal r_{i+1} based on the transition probability and reward function. We will use $\mu^{\pi_\theta}(s, a)$ and $\mu^{\pi_\theta}(s', s)$ to denote the state-action and state-next-state marginals of the trajectory distribution induced by the policy π_θ. These marginal distributions describe the state-action and state-next-state frequency over all time steps of a given policy π_θ.

2.1 Maximum entropy reinforcement learning and soft actor critic

The standard objective in RL is the expected sum of undiscounted rewards $\sum_{i=0}^{T} \mathbb{E}_{(s_i,a_i)\sim\mu^{\pi_\theta}}[r(s_i, a_i)]$. The goal of the agent is to learn a policy $\pi_\theta(a_i|s_i)$ which maximises this objective. The maximum entropy objective

$$J_\pi(\theta) = \sum_{i=0}^{T} \mathbb{E}_{(s_i,a_i)\sim\mu^{\pi_\theta}}[r(s_i, a_i) + \alpha\mathcal{H}(\pi_\theta(\cdot|s_i))] \tag{1}$$

introduces a modified goal for the RL agent, where the agent has to maximise the sum of the reward signal and its output entropy $\mathcal{H}(\pi_\theta(\cdot|s_i)) = \mathbb{E}_{a\sim\pi_\theta}[-\log\pi_\theta(\cdot|s_i)]$ [11]. The parameter α controls the stochasticity of the optimal policy

[1] An extension to infinite-horizons is given in Section 3

by determining the relative importance of the entropy term versus the reward.

Soft Actor-Critic (SAC) [15, 17] is an off-policy actor-critic algorithm based on the maximum entropy RL framework. Since we apply SAC in our imitation learning setup the main objectives will be briefly explained. SAC combines off-policy Q-Learning with a stable stochastic actor-critic formulation.

The state value function and Q-function are used to estimate how good an agent is in a specific state s_i or to perform a specific action a_i in a given state s_i, based on the expected return. The Q-function can be estimated using a function approximator $Q_\Psi(s_i, a_i)$. The soft Q-function parameters Ψ can be trained to minimize the soft Bellman residual

$$J_Q(\Psi) = \mathbb{E}_{(s_i,a_i)\sim D_{RB}}[\frac{1}{2}(Q_\Psi(s_i, a_i) - \hat{Q}_{\hat{\Psi}}(s_i, a_i))^2], \quad (2)$$

where state-action pairs are sampled from a replay buffer D_{RB} which contains state-action pairs from repeated policy-environment interactions. The target Q-function $\hat{Q}_{\hat{\Psi}}$ can be estimated by

$$\hat{Q}_{\hat{\Psi}}(s_i, a_i) = r(s_i, a_i) + \gamma \mathbb{E}_{s_{i+1}}[V_{\hat{\Psi}}(s_{i+1})], \quad (3)$$

using the soft state value function $V_{\hat{\Psi}}(s_i)$, the current reward $r(s_i, a_i)$ and the discount factor γ. There is no need to include a separate function approximator for the soft value function since the state value is related to the Q-function and the policy by

$$V_{\hat{\Psi}}(s_i) := \mathbb{E}_{a_i\sim\pi_\theta}[Q_{\hat{\Psi}}(s_i, a_i) - \alpha \log \pi_\theta(a_i|s_i)]. \quad (4)$$

To stabilize the training, the update uses a target network with parameters $\hat{\Psi}$ that are a moving average of the parameters Ψ. Lastly, the policy is optimized by minimizing the following objective:

$$J_\pi(\theta) = \mathbb{E}_{s_i\sim D_{RB}}[\mathbb{E}_{a_i\sim\pi_\theta}[\alpha \log \pi_\theta(a_i|s_i) - Q_\Psi(s_i, a_i)]], \quad (5)$$

where the states are sampled from the replay buffer D_{RB} and the actions a_i are sampled using the current policy π_θ.

2.2 Imitation learning

In the IL setup, the agent does not have access to the true environment reward function $r(s_i, a_i)$ and instead has to imitate expert trajectories performed by an expert policy π_E collected in a dataset $\mathcal{D}_E$.

In the typical **learning-from-demonstration** setup the expert demonstrations $\mathcal{D}_E^{LfD} := \{s_i^k, a_i^k, s_{i+1}^k\}_{k=1}^N$ are given by state-action-next-state transitions. Distribution matching has been a popular choice among different LfD approaches. The policy π_θ is learned by minimizing the discrepancy between the stationary state-action distribution induced by the expert $\mu^E(s, a)$ and the policy $\mu^{\pi_\theta}(s, a)$. An overview and comparison of different LfD objectives resulting from this discrepancy minimization were made by Ghasemipour et al. [18]. Often the backward KLD is used to measure this discrepancy [19]:

$$J_{LfD}(\pi_\theta) := \mathbb{D}_{KL}(\mu^{\pi_\theta}(s, a)||\mu^E(s, a)). \quad (6)$$

Learning-from-observation (LfO) considers a more challenging task where expert actions are not available. Hence, the demonstrations $\mathcal{D}_E^{LfO} := \{s_i^k, s_{i+1}^k\}_{k=1}^N$ consist of state-next-state transitions. The policy learns which actions to take based on interactions with the environment and the expert state transitions. Distribution matching based on state-transition distributions is a popular choice for state-only IL [5, 12]:

$$J_{LfO}(\pi_\theta) := \mathbb{D}_{KL}(\mu^{\pi_\theta}(s', s)||\mu^E(s', s)). \quad (7)$$

3 Method

In a finite horizon MDP setting, the joint state-only trajectory distributions are defined by the start state distribution $p(s_0)$ and the product of the conditional state transition distributions $p(s_{i+1}|s_i)$. For the policy distribution μ^{π_θ} and the expert distribution μ^E, this becomes

$$\mu^{\pi_\theta}(s_T, \ldots, s_0) = p(s_0) \prod_{i=0\ldots T-1} \mu^{\pi_\theta}(s_{i+1}|s_i),$$
$$\mu^E(s_T, \ldots, s_0) = p(s_0) \prod_{i=0\ldots T-1} \mu^E(s_{i+1}|s_i).$$

Our goal is to match the state-only trajectory distribution μ^{π_θ} induced by the policy with the state-only expert trajectory distribution μ^E by minimizing the Kulback-Leibler divergence (KLD) between them. This results in the SOIL-TDM objective

$$\begin{aligned} J_{SOIL-TDM} &= \mathbb{D}_{KL}(\mu^{\pi_\theta}||\mu^E) \\ &= \mathbb{E}_{(s_T,\ldots,s_0)\sim\mu^{\pi_\theta}}[\log \mu^{\pi_\theta} - \log \mu^E] \\ &= \sum_{i=0\ldots T-1} \mathbb{E}_{(s_{i+1},s_i)\sim\mu^{\pi_\theta}}[\log \mu^{\pi_\theta}(s_{i+1}|s_i) \\ &\quad - \log \mu^E(s_{i+1}|s_i)]. \end{aligned} \quad (8)$$

To estimate the policy-induced conditional state transition distribution, we define the following equation based on the

Bayes theorem

$$\pi'_\theta(a_i|s_{i+1}, s_i) = \frac{p(s_{i+1}|a_i, s_i)\pi_\theta(a_i|s_i)}{\mu^{\pi_\theta}(s_{i+1}|s_i)}. \quad (9)$$

The posterior distribution is represented by the inverse action distribution density $\pi'_\theta(a_i|s_{i+1}, s_i)$, the likelihood distribution is represented by the environment model $p(s_{i+1}|a_i, s_i)$, the prior is represented by the policy distribution $\pi_\theta(a_i|s_i)$, and the marginal likelihood by the policy-induced conditional state transition distribution $\mu^{\pi_\theta}(s_{i+1}|s_i)$. By solving for the marginal likelihood, we can rewrite (9) to

$$\mu^{\pi_\theta}(s_{i+1}|s_i) = \frac{p(s_{i+1}|a_i, s_i)\pi_\theta(a_i|s_i)}{\pi'_\theta(a_i|s_{i+1}, s_i)}. \quad (10)$$

It holds for any a_i where $\pi' > 0$. Thus, one can extend the expectation over (s_{i+1}, s_i) by the action a_i and the KLD minimization $\min \mathbb{D}_{KL}(\mu^{\pi_\theta}||\mu^E)$ can be rewritten as

$$\min \sum_{i=0\ldots T-1} \mathbb{E}_{(s_i,a_i,s_{i+1})\sim\pi_\theta}[\log p(s_{i+1}|a_i, s_i) + \log \pi_\theta(a_i|s_i)$$
$$- \log \pi'_\theta(a_i|s_{i+1}, s_i) - \log \mu^E(s_{i+1}|s_i)]. \quad (11)$$

Now, we define a reward function (also see Appendix B)

$$r(a_i, s_i) := \mathbb{E}_{s_{i+1}\sim p(s_{i+1}|a_i,s_i)}[-\log p(s_{i+1}|a_i, s_i)$$
$$+ \log \pi'_\theta(a_i|s_{i+1}, s_i) + \log \mu^E(s_{i+1}|s_i)], \quad (12)$$

which depends on the expert state transition likelihood $\mu^E(s_{i+1}|s_i)$, on the environment model $p(s_{i+1}|a_i, s_i)$ and on the inverse action distribution density $\pi'_\theta(a_i|s_{i+1}, s_i)$. Using this reward function, the state-only trajectory distribution matching problem can be transformed into a max-entropy RL task

$$\min \mathbb{D}_{KL}(\mu^{\pi_\theta}||\mu^E) = \max \sum_{i=0\ldots-1} \mathbb{E}_{(a_i,s_i)\sim\pi_\theta}[-\log \pi_\theta(a_i|s_i) + r(a_i, s_i)]$$
$$= \max \sum_{i=0\ldots T-1} \mathbb{E}_{(a_i,s_i)\sim\pi_\theta}[r(a_i, s_i) + \mathcal{H}(\pi_\theta(\cdot|s_i)]. \quad (13)$$

In practice, the reward function $r(a_i, s_i)$ can be computed using Monte Carlo integration with a single sample from $p(s_{i+1}|a_i, s_i)$ using the replay buffer.

This max-entropy RL task can be optimized with standard max-entropy RL algorithms. In this work, we applied the SAC algorithm [15] as it is outlined in Section 2.1.

The extension to infinite horizon tasks can be done by introducing a discount factor γ as in the work by Haarnoja et al. [17]. In combination with our reward definition, one obtains the following infinite horizon maximum entropy objective

$$J_{ME-iH} = \sum_{i=0\ldots\inf} \mathbb{E}_{(a_i,s_i)\sim\pi_\theta}[\sum_{j=i\ldots\inf} \gamma^{j-i}\mathbb{E}_{(a_j,s_j)\sim\pi_\theta}[r(a_j, s_j)$$
$$+\mathcal{H}(\pi_\theta(\cdot|s_j)|s_i, a_i]]. \quad (14)$$

3.1 Algorithm

To evaluate the reward function, the environment model $p(s_{i+1}|a_i, s_i)$ and the inverse action distribution function $\pi'_\theta(a_i|s_{i+1}, s_i)$ have to be estimated. We model both distributions using conditional normalizing flows and train them with maximum likelihood based on expert demonstrations and rollout data from a replay buffer. The environment model $p(s_{i+1}|a_i, s_i)$ is modeled by $\mu_\phi(s_{i+1}|a_i, s_i)$ with parameter ϕ, while the inverse action distribution function $\pi'_\theta(a_i|s_{i+1}, s_i)$ is modeled by $\mu_\eta(a_i|s_{i+1}, s_i)$ with parameter η.

The whole training process according to Algorithm 1 is described in the following.[2] The expert state transition model $\mu^E(s_{i+1}|s_i)$ is trained offline using the expert dataset D_E which contains K expert state trajectories. We assume the expert distribution is correctly represented by the dataset. If the expert demonstrations are incomplete and important information is missing, the resulting policy might perform sub-optimally. For optimal performance, additional guidance like an additional reward might be a possible solution under such circumstances. We show that our method can learn meaningful expert state transition distributions even for incomplete expert trajectories (Appendix G). Since we use a normalizing flow to estimate the expert state transition distribution, this transition distribution can be complex and multimodal. Therefore learning the behavior from different experts is possible. We performed experiments using experts trained differently in the same environment by using different Reinforcement Learning algorithms. The results show that our method is able to train well-performing agents also in the case of multimodal demonstrations which come from different expert trajectories. Density modeling of the expert state transitions can still result in overfitting when only a few expert demonstrations are available (due to limit of the sample amount). We improve the expert training process by adding Gaussian noise to the state values. The standard deviation of the noise is reduced during training so that the model has a correct estimate of the density without overfitting to the

[2] Code will be available at https://github.com/FeMa42/soil-tdm

explicit demonstrations. With this approach, we are able to successfully train expert state transition models on at least one expert trajectory. The influence of this improved routine is studied in Appendix F.

After this initial step, the following process is repeated until convergence in each episode. The policy $\pi_\theta(\hat{a}_i|s_i)$ interacts with the environment for T steps by generating an action $\hat{a}_i$ based on the current state s_i. The environment generates a next-state s_{i+1} based on its current state and the received action at each time step. The collected state-action-next-state information is saved in the replay buffer D_{RB}. The conditional normalizing flows for the environment model $\mu_\phi(s_{i+1}|a_i, s_i)$ (policy independent) and the inverse action distribution model $\mu_\eta(a_i|s_{i+1}, s_i)$ (policy dependent) are optimized using samples from the replay buffer D_{RB} for N steps. In Appendix C we show that this reduces the KLD (11) in each step.

To train the Q-function, we compute a one-sample Monte-Carlo approximation of the reward using the learned models together with the samples from the reply buffer. The policy $\pi_\theta(a_t|s_t)$ is updated based on $J_\pi(\theta)$ using SAC as described in Section 2.1. The entropy weight α in (1) is set to constant 1 during the optimization without automatic entropy weight tuning (proposed in [17]).

The SAC-based Q-function training and policy optimization also minimize the KLD in (11) (see (13)) in each step. During the SAC optimization, the reward function is fixed. Alternately, the subcomponents of it (μ_ϕ and μ_η) are trained separately to the policy optimization. However, since after each SAC policy learning step we use data from new rollouts to adapt the inverse action distribution approximation (Algorithm 1), π'_θ and μ_η do not diverge. Theoretically, without the limit of sample amount, and under the assumption of well-behaved function approximators, the inverse action distribution (π'_θ) can be approximated without bias. The SAC based policy optimization together with the inverse action policy learning leads to a converging algorithm since all steps reduce the KLD, which is bounded from below by 0 (see Appendix C).

It is worth noting that the overall algorithm is non-adversarial, the inverse action policy optimization and the policy optimization using SAC both reduce the overall objective - the KLD. On the contrary, the AIL algorithms (like OPOLO) are based on an adversarial nested min-max optimization. Additionally, we can estimate the similarity of state transitions from our policy to the expert during each optimization step, since we model all densities in the rewritten KLD from (11). As a result, we have a reliable performance estimate enabling us to select the best-performing policy based on the lowest KLD between policy and expert state transition trajectories.

3.2 Relation to learning from observations

The LfO objective of previous approaches like OPOLO minimizes the divergence between the joint policy state transition distribution and the joint expert state transition distribution

$$J_{LfO}(\pi_\theta) = \mathbb{D}_{KL}(\mu^{\pi_\theta}(s', s)||\mu^E(s', s)), \tag{15}$$

which can be rewritten as (see Appendix A)

$$\begin{aligned} J_{LfO}(\pi_\theta) = \; & \mathbb{D}_{KL}(\mu^{\pi_\theta}(s_T, \ldots, s_0)||\mu^E(s_T, \ldots, s_0)) \\ & + \sum_{i=1\ldots T-1} \mathbb{D}_{KL}(\mu^{\pi_\theta}(s_i)||\mu^E(s_i)). \end{aligned} \tag{16}$$

Thus, the LfO objective minimizes both KLD of the joint distributions and the KLDs of the marginal distributions of all time steps. The SOIL-TDM objective in comparison minimizes purely the KLD of the joint distributions. In the case of perfect distribution matching - a zero KLD between the joint distributions - the KLDs of the marginals also vanish so both objectives have the same optimum. Minimizing purely the KLD of the joint distributions can contribute to the robustness of the learning algorithm, as it was demonstrated by Jaegle et al. [10]. Methods based on conditional state probabilities are less sensitive to erroneously penalizing features that may not be in the demonstrator data but lead to correct transitions. Hence, such methods may be less prone to overfit to irrelevant differences between the learner and the expert data. This, as well as the relation to the work by Jaegle et al. [10], is discussed further in Section 4.1.

4 Related work

Many recent IL approaches are based on inverse RL (IRL) [20]. In IRL, the goal is to learn a reward signal for which the expert policy is optimal. AIL algorithms are popular methods to perform IL in a RL setup [2, 19, 21]. In AIL, a discriminator gets trained to distinguish between expert states and states coming from policy rollouts. The goal of the policy is to fool the discriminator. The policy gets optimized to match the state action distribution of the expert using this two-player game. Based on this idea more general approaches have been derived based on f-divergences. Ni et al. [16] derived an analytic gradient of any f-divergence between the agent and expert state distribution w.r.t. reward parameters. Based on this gradient they presented the algorithm F-IRL that recovers a stationary reward function from the expert density by gradient descent. Ghasemipour et al. [18] identified that IRL's state-marginal matching objective contributes most to its superior performance and applied this

Algorithm 1 State-Only Imitation Learning by Trajectory Distribution Matching (SOIL-TDM).

```
1: Input:
2: Expert Dataset $D_E : \{s_0, s_1, \dots s_T\}_{k=0}^{K}$
3: Randomly initialized $\mu^E$, $\mu_\eta$, $\mu_\phi$, $Q_\psi$, $Q_{\hat{\Psi}}$, and $\pi_\theta$
4: procedure SOIL- TDM($D_E$)
5:     Train $\mu^E(s_{i+1}|s_i)$ with $D_E$          ▷ expert model optimization
6:     for episodes do
7:         for range(T) do                              ▷ generate data
8:             $\hat{a}_i \leftarrow$ sample($\pi_\theta(\hat{a}_i|s_i)$)
9:             $s_{i+1} \leftarrow p_{sim}(s_{i+1}|s_i, \hat{a}_i)$          ▷ apply action
10:            Store $(s_i, \hat{a}_i, s_{i+1})$ in $D_{RB}$
11:        end for
12:        for range(N) do                              ▷ update dynamics models
13:            $\{(s_i, \hat{a}_i, s_{i+1})\}_{i=1}^{B} \sim D_{RB}$          ▷ sample batch
14:            train $\mu_\eta(\hat{a}_i|s_{i+1}, s_i)$ and $\mu_\phi(s_{i+1}|\hat{a}_i, s_i)$
15:        end for
16:        for range(N) do                              ▷ SAC optimization
17:            $\{(s_i, \hat{a}_i, s_{i+1})\}_{i=1}^{B} \sim D_{RB}$          ▷ sample batch
18:            $a_i \leftarrow$ sample($\pi_\theta(a_i|s_i)$)
19:            optimize $\pi_\theta(a_i|s_i)$ with $J_\pi$ from (5)
20:                                                     ▷ estimate reward
21:
               $r(s_i, \hat{a}_i) \leftarrow -\log \mu_\phi(s_{i+1}|\hat{a}_i, s_i) + \log \mu_\eta(\hat{a}_i|s_{i+1}, s_i)$
                   $+ \log \mu^E(s_{i+1}|s_i)$
22:            optimize $Q_\psi(\hat{a}_i, s_i)$ with $J_Q$ from (2)
23:        end for
24:    end for
25: end procedure
26: Output: Trained Normalizing Flow models $\mu^E$, $\mu_\eta$, $\mu_\phi$, and policy $\pi_\theta$
```

understanding to teach agents a diverse range of behaviors using simply hand-specified state distributions.

A key problem with AIL for LfD and LfO is optimization instability [6]. Wang et al. [22] avoided the instabilities resulting from adversarial optimization by estimating the support of the expert policy to compute a fixed reward function. Similarly, Brantley et al. [23] used a fixed reward function by estimating the variance of an ensemble of policies. Both methods rely on additional behavioral cloning steps to reach expert-level performance. Liu et al. [24] proposed Energy-Based Imitation Learning (EBIL) which recovers fixed and interpretative reward signals by directly estimating the expert's energy. Neural Density Imitation (NDI) [25] uses density models to perform distribution matching. Deterministic and Discriminative Imitation (D2-Imitation) [26] requires no adversarial training by partitioning samples into two replay buffers and then learning a deterministic policy via off-policy reinforcement learning. Inverse soft-Q learning (IQ-Learn) [27] avoids adversarial training by learning a single Q-function to represent both reward and policy implicitly. The implicitly learned rewards from IQ-Learn show a high positive correlation with the ground-truth rewards.

LfO can be divided into model-free and model-based approaches. GAILfO [5] is a model-free approach that uses the GAIL principle with the discriminator input being state-only. Yang et al. [28] analyzed the gap between the LfD and LfO objectives and proved that it lies in the disagreement of inverse dynamics models between the imitator and expert. Their proposed method Inverse-Dynamics-Disagreement-Minimization (IDDM) is based on an upper bound of this gap in a model-free way. OPOLO [12] is a sample-efficient LfO approach also based on AIL, which enables off-policy optimization. The policy update is also regulated with an inverse action model that assists distribution matching in a mode-covering perspective.

Other model-based approaches either apply forward dynamics models or inverse action models. Sun et al. [29] proposed a solution based on forward dynamics models to learn time dependent policies. Although being provably efficient, it is not suited for infinite horizon tasks. Alternatively, behavior cloning from observations (BCO) [13] learns an inverse action model based on simulator interactions to infer actions based on the expert state demonstrations. GPRIL [30] uses normalizing flows as generative models to learn backward dynamics models to estimate predecessor transitions and augment the expert data set with further trajectories, which lead to expert states. Jiang et al. [31] investigated IL using few expert demonstrations and a simulator with misspecified dynamics. A detailed overview of LfO was done by Torabi et al. [4].

4.1 Method discussion and relation to FORM

While our proposed method SOIL-TDM was independently developed, it is most similar to the state-only approach FORM [10]. In FORM the policy training is guided by a conditional density estimation of the expert's observed state transitions. In addition a state transition model $\mu_\Phi^{\pi_\theta}(s_{i+1}|s_i)$ of the current policy is learned. The policy reward is estimated by: $r_i = log\mu^E(s_{i+1}|s_i) - log\mu_\Phi^{\pi_\theta}(s_{i+1}|s_i)$. The approach matches conditional state transition probabilities of expert and policy in comparison to the joint state-action (like GAIL) or joint state-next-state (like OPOLO or GAILfO) densities. The authors of FORM argue that this conditional state matching contributes to the robustness of their approach. Namely, methods based on conditional state probabilities are less sensitive to erroneously penalizing features that may not be in the demonstrator data but lead to correct transitions. Hence, such methods may be less prone to overfit to irrelevant differences. Jaegle et al. [10] demonstrate the benefit of such a conditional density matching approach.

In contrast to FORM we show in (10) that the policies next-state conditional density $\mu^{\pi_\theta}(s_{i+1}|s_i)$ can be separated into the policies action density and the forward- and the backward-dynamics densities. Using this decomposition, we show that the KLD minimization is equivalent to a maximum entropy RL objective (see (13)) with a special reward (see (12)). Here the entropy of the policy stemming from the decomposition of the conditional state-next-state density leads to the maximum entropy RL objective. Jaegle et al. [10] mention that the second term $log \mu_{\Phi}^{\pi_\theta}(s_{i+1}|s_i)$ in their reward objective can be viewed as an entropy-like expression. Hence, if this reward is optimized using a RL algorithm that includes some form of policy entropy regularization like SAC this entropy is basically weighted twice. In the experiments, we show that this double accounting of the policy entropy negatively affects the sample efficiency of the algorithm in comparison to our method.

5 Experiments

We evaluate our proposed method described in Section 3 in a variety of different IL tasks and compare it against the baseline methods OPOLO, F-IRL, and FORM. For all methods, we use the complex and high-dimensional continuous control environments AntBulletEnv-v0, HalfCheetahBulletEnv-v0, HopperBulletEnv-v0, Walker2DBulletEnv-v0, Humanoid BulletEnv-v0 of the Pybullet physics simulation [32]. To evaluate the performance of all methods, the cumulative rewards of the trained policies are compared to cumulative rewards from the expert policy. The expert data generation as well as the used baseline implementations are described in Appendix E.

Since we assume no environment reward is available as an early stopping criterion, we use other convergence estimates available during training to select the best policy for each method. In adversarial training, the duality gap [8, 9] is an established method to estimate the convergence of the training process. In the IL setup, the duality gap can be very difficult to estimate since it requires the gradient of the policy and the environment (i.e. the gradient of the generator) for the optimization process it relies on. We, therefore, use two alternatives for model selection for OPOLO. The first approach selects the model with the lowest policy loss and the second approach selects the model based on the highest estimated cumulative reward over ten consecutive epochs. For F-IRL we selected the model with the lowest estimated Jensen-Shannon divergence over ten epochs. To estimate the convergence of SOIL-TDM the policy loss based on the KLD from (11) is used. It can be estimated using the same models used for training the policy. Similarly, we used the effect models of FORM to estimate the convergence based on the reward.

Many IL approaches show asymptotic performance. Although it is a reasonable comparison, we also argue that early stopping based on estimated performance gives valuable insights into how well the policy performance can be estimated without relying on external signals like an environment reward. It especially shows that the performance of adversarial methods can be estimated less reliably. Estimating the convergence asymptotically for unknown hyperparameter setups without any external signal is therefore less reliable. However, for complete transparency, we also included results based on the best environment reward and also added additional results in Appendix H which show the time to convergence as well as the asymptotic performance of all methods.

The evaluation is done by running 3 training runs with ten test episodes (in total 30 rollouts) for each trained policy and calculating the respective mean and confidence interval for all runs. The methods are compared based on different amounts of expert trajectories. This limited amount of expert demonstrations can cause a suboptimal learned representation of expert behavior which can lead to a deviation to expert performance. We plot the cumulative reward normalized so that 1 corresponds to expert performance. Values above 1 mean that the agent has achieved a higher cumulative reward than the mean of the expert. The expert in the proposed experiments is a policy trained on the true environment reward using SAC. The expert is not necessarily optimal w.r.t the environment reward (episode rewards of the expert are reported in Appendix E). Hence, achieving the most similar behavior is more desirable than surpassing the reward of the expert. Additionally, smaller confidence intervals are also desired since they indicate higher training stability and more reliable results. Implementation details of our method are described in Appendix D.

The evaluation results of the discussed methods on a suite of continuous control tasks with unknown true environment reward and the previously described selection criteria are shown in Fig. 1.

Although the true environment reward is unknown the results show that SOIL-TDM achieves or surpasses the performance of the baseline methods on all tested environments (except for two and four expert trajectories in the HumanoidBulletEnv-v0 environment and two trajectories in the Walker2DBulletEnv-v0 environment). In general the adversarial-based methods OPOLO and F-IRL exhibit a high variance of the achieved rewards using the proposed selection criteria. Although loss and reward are well suited for selecting the best model in usual setups, the results demonstrate that they might be less expressive for estimating the convergence in adversarial training due to the min-max game of the discriminator and the policy. The stability of the SOIL-TDM training method is evident from the small confidence band of the results which gets smaller for more expert demonstrations. Although the selection of FORM is more stable than the adversarial methods it generally achieves lower rewards in the sample efficient regime of one and two expert trajectories.

Figure 2 shows the benchmark results of OPOLO, F-IRL, FORM, and SOIL-TDM if the true environment reward is

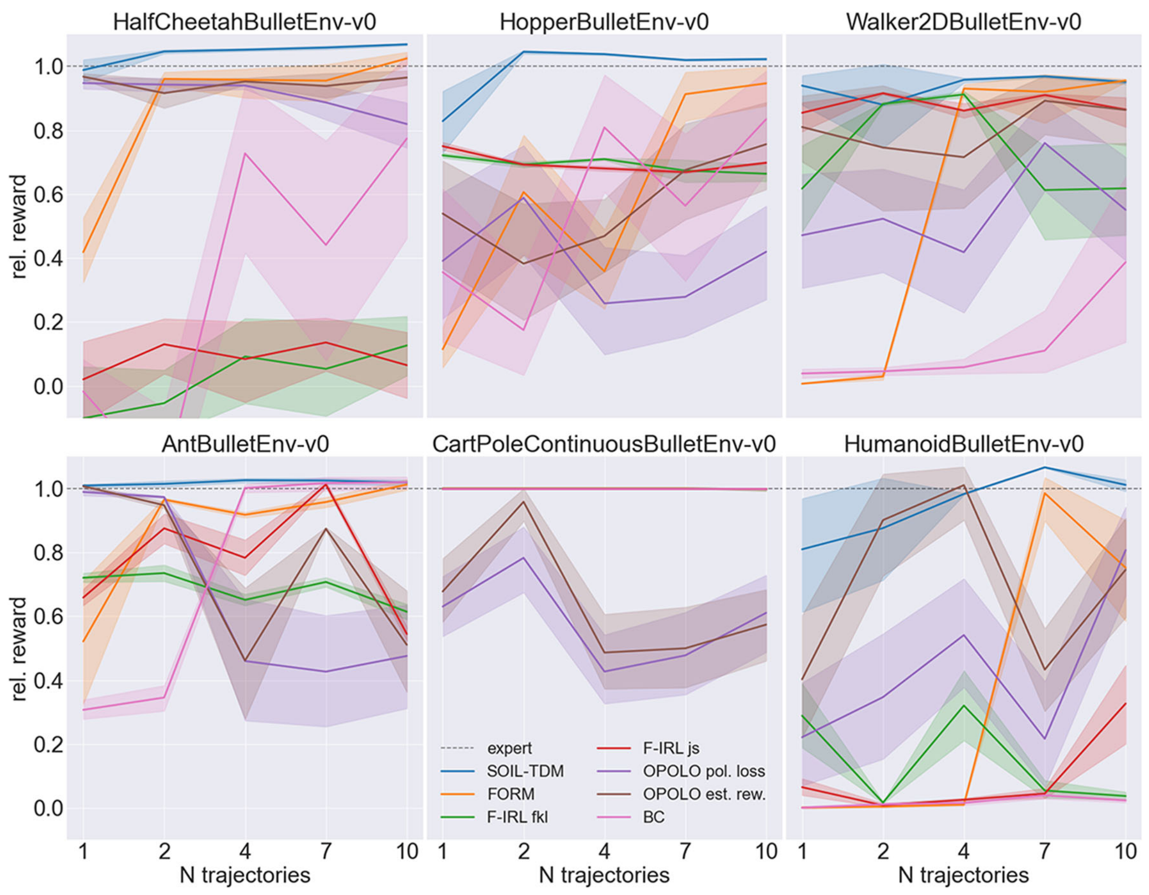


Fig. 1 Unkown true environment reward selection criteria: Relative cumulative reward for a different amount of expert trajectories on continuous control environments. The best policies based on estimated convergence values were selected. The value 1 corresponds to expert policy performance. The confidence intervals are plotted using lighter colors

used as an early stopping criterion. In this setup, our method still achieves competitive performance or surpasses OPOLO, F-IRL, and FORM. Compared to the results from Fig. 1 the baseline methods often achieve better results using the true environment reward as a selection criterion. In contrast, our proposed method has more similar results for both selection criteria, which underlines the reliability of our proposed performance estimation.

We argue that the reduced performance of the baseline methods OPOLO and F-IRL are due to missing reliable and tractable convergence estimators for adversarial-based approaches if the best policy is selected based on estimated performance. The results based on the best true environment reward selection criterion are more similar in performance among the different IL approaches. The optimum can only be reached if enough expert demonstrations are available and for enough trajectories, all methods result in expert-level performance. Although the statement that SOIL-TDM and OPOLO have the same optimum under the condition of a zero KLD between the joint distributions (which might be violated with conditional Gaussian policies and limited data) is not contradicted by the experiments, we can show that SOIL-TDM is efficient regarding expert demonstrations. Additional figures for comparing the training performance and efficiency can be found in Appendix H. An ablation study for our method is in Appendix F.

6 Conclusion

In this work, we propose a non-adversarial state-only imitation learning approach based on minimising the Kulback-Leibler divergence between the policy and the expert state trajectory distribution. This objective leads to a maximum entropy reinforcement learning problem with a reward function depending on the expert state transition distribution and the forward and backward dynamics of the environment which can be modelled using conditional normalizing flows. The proposed approach is compared to several state-of-the-art learning from observation methods in a scenario with unknown environment rewards and achieves state-of-the-art performance.

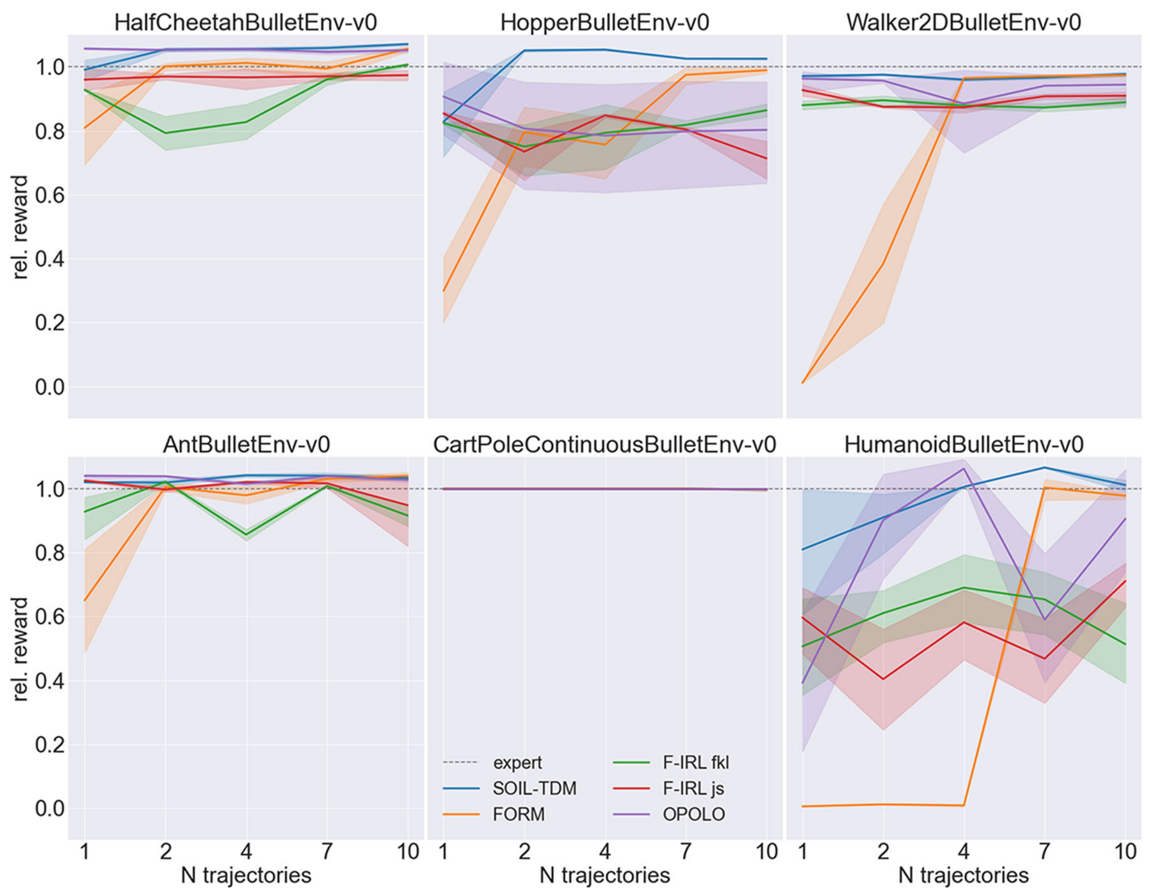


Fig. 2 Best true environment reward selection criterion: Relative cumulative reward for a different amount of expert trajectories on continuous control environments. The best policies based on the cumulative reward were selected. The value 1 corresponds to expert policy performance. The confidence intervals are plotted using lighter colors

Appendix A: Relation to LfO

The learning from observations (LfO) objective minimizes the divergence between the joint policy state transition distribution and the joint expert state transition distribution:

$$\min J_{LfO}(\pi_\theta) = \min \mathbb{D}_{KL}(\mu^{\pi_\theta}(s', s)||\mu^E(s', s)) \tag{A1}$$

where s' is a successor state of s given a stationary policy and stationary s', s marginals. This can be rewritten as

$$\begin{aligned}
J_{LfO}(\pi_\theta) &= \sum_{i=0\ldots T-1} \mathbb{D}_{KL}(\mu^{\pi_\theta}(s_{i+1}, s_i)||\mu^E(s_{i+1}, s_i)) \\
&= \sum_{i=0\ldots T-1} \int \mu^{\pi_\theta}(s_{i+1}, s_i)(\log \mu^{\pi_\theta}(s_{i+1}, s_i) \\
&\quad - \log \mu^E(s_{i+1}, s_i)) \\
&= \sum_{i=0\ldots T-1} \int \mu^{\pi_\theta}(s_T, \ldots, s_0)(\log \mu^{\pi_\theta}(s_{i+1}, s_i) \\
&\quad - \log \mu^E(s_{i+1}, s_i)) \\
&= \int \mu^{\pi_\theta}(s_T, \ldots, s_0) \sum_{i=0\ldots T-1} (\log \mu^{\pi_\theta}(s_{i+1}, s_i) \\
&\quad - \log \mu^E(s_{i+1}, s_i)) \\
&= \int \mu^{\pi_\theta}(s_T, \ldots, s_0) \sum_{i=0\ldots T-1} (\log \mu^{\pi_\theta}(s_{i+1}|s_i) \\
&\quad + \log \mu^{\pi_\theta}(s_i) - \log \mu^E(s_{i+1}|s_i)) - \log \mu^E(s_i) \\
&= \mathbb{E}_{(s_T,\ldots,s_0)\sim\mu^{\pi_\theta}}[\log \frac{\mu^{\pi_\theta}(s_T, \ldots, s_0)}{\mu^E(s_T, \ldots, s_0)} \\
&\quad + \log \prod_{i=1\ldots T-1} \frac{\mu^{\pi_\theta}(s_i)}{\mu^E(s_i)}] \\
&= \mathbb{D}_{KL}(\mu^{\pi_\theta}(s_T, \ldots, s_0)||\mu^E(s_T, \ldots, s_0)) \\
&\quad + \sum_{i=1\ldots T-1} \mathbb{E}_{(s_T,\ldots,s_0)\sim\mu^{\pi_\theta}}[\log \frac{\mu^{\pi_\theta}(s_i)}{\mu^E(s_i)}] \\
&= \mathbb{D}_{KL}(\mu^{\pi_\theta}(s_T, \ldots, s_0)||\mu^E(s_T, \ldots, s_0)) \\
&\quad + \sum_{i=1\ldots T-1} \mathbb{D}_{KL}(\mu^{\pi_\theta}(s_i)||\mu^E(s_i))
\end{aligned} \tag{A2}$$

Appendix B: Bounded rewards

Since we use the SAC algorithm as a subroutine all rewards must be bounded. This is true if all subterms of our reward function

$$r(a_i, s_i) = \mathbb{E}_{s_{i+1} \sim \mu^{\pi_\theta}(s_{i+1}|s_i)}[-\log p(s_{i+1}|a_i, s_i) + \log \pi'_\theta(a_i|s_{i+1}, s_i) + \log \mu^E(s_{i+1}|s_i)] \quad \text{(B3)}$$

are bounded which holds if

$$\epsilon \leq \pi'_\theta(a_i|s_{i+1}, s_i), \quad p(s_{i+1}|a_i, s_i), \quad \mu^E(s_{i+1}|s_i) \leq H \qquad \forall a_i, s_i, s_{i+1} \quad \text{(B4)}$$

for some ϵ and H which is a rather strong assumption that requires compact action and state spaces and a non-zero probability of reaching every state s_{i+1} given any action a_i from a predecessor state s_i. Since this is in general not the case in practice, we clip the logarithms of $\pi'_\theta(a_i|s_{i+1}, s_i)$, $p(s_{i+1}|a_i, s_i)$, $\mu^E(s_{i+1}|s_i)$ to $[-15, 1e9]$. It should be noted that clipping the logarithms to a maximum negative value still provides a reward signal which guides the imitation learning to policies that achieve higher rewards. The clip interval was selected such that the trainings on all tested environments converged. A wide range of intervals converges and just very large absolute values should be clipped.

Appendix C: Correctness of using replay buffer

Here we argue that Algorithm 1 leads to a local optimum of the KLD objective from (11) under the conditions: a) In the large sample limit per iteration b) appropriate density estimators and optimizers are used c) (11) is minimized by optimizing the policy using a maximum entropy RL algorithm d) the inverse action policy $\pi'(a|s', s)$ is trained using maximum likelihood from a replay buffer in an alternating fashion with the policy optimization.

In Section 3 we show that (11) is a maximum entropy RL objective. Thus when optimizing the policy $\pi_{\theta(e)}$ in episode e in Algorithm 1 using the maximum entropy RL algorithm SAC [15] keeping the parameters of the conditional normalizing flows $\mu^E(s'|s)$, $\mu_\phi(s'|s, a)$, and $\mu_\eta(a|s', s)$ (to stay consistent with our method section, we use the distribution definitions here instead of the model definition) fixed which define the reward implies

$$\sum_{i=0...T-1} \mathbb{E}_{(s_i, a_i, s_{i+1}) \sim \pi_{\theta(e)}}[\log p(s_{i+1}|a_i, s_i) + \log \pi_{\theta(e)}(a_i|s_i) - \log \pi'_{\theta(e)}(a_i|s_{i+1}, s_i) - \log \mu^E(s_{i+1}|s_i)]$$
$$\leq \sum_{i=0...T-1} \mathbb{E}_{(s_i, a_i, s_{i+1}) \sim \pi_{\theta(e-1)}}[\log p(s_{i+1}|a_i, s_i) + \log \pi_{\theta(e-1)}(a_i|s_i) - \log \pi'_{\theta(e)}(a_i|s_{i+1}, s_i) - \log \mu^E(s_{i+1}|s_i)] \quad \text{(C5)}$$

Now, in the next episode $e+1$ the first part of Algorithm 1, i.e. optimizing the model of the inverse action policy $\pi'_{\theta(e+1)}$ with a maximum likelihood objective using the new replay buffer data $(s_i, a_i, s_{i+1}) \sim \pi_{\theta(e)}$ obtained from rollouts in episode $e+1$ with the new policy $\pi_{\theta(e)}$ trained in episode e leads to

$$\sum_{i=0...T-1} \mathbb{E}_{(s_i, a_i, s_{i+1}) \sim \pi_{\theta(e)}}[-\log \pi'_{\theta(e+1)}(a_i|s_{i+1}, s_i)]$$
$$\leq \sum_{i=0...T-1} \mathbb{E}_{(s_i, a_i, s_{i+1}) \sim \pi_{\theta(e)}}[-\log \pi'_{\theta(e)}(a_i|s_{i+1}, s_i)] \quad \text{(C6)}$$

due to the maximum likelihood objective for the inverse action policy.

Using this inequality one obtains

$$\sum_{i=0...T-1} \mathbb{E}_{(s_i, a_i, s_{i+1}) \sim \pi_{\theta(e)}}[\log p(s_{i+1}|a_i, s_i) + \log \pi_{\theta(e)}(a_i|s_i) - \log \pi'_{\theta(e+1)}(a_i|s_{i+1}, s_i) - \log \mu^E(s_{i+1}|s_i)]$$
$$\leq \sum_{i=0...T-1} \mathbb{E}_{(s_i, a_i, s_{i+1}) \sim \pi_{\theta(e-1)}}[\log p(s_{i+1}|a_i, s_i) + \log \pi_{\theta(e-1)}(a_i|s_i) - \log \pi'_{\theta(e)}(a_i|s_{i+1}, s_i) - \log \mu^E(s_{i+1}|s_i)] \quad \text{(C7)}$$

Thus Algorithm 1 optimizes (11) also in the "update dynamics models" part when training $\mu_\eta(a|s', s)$ using maximum likelihood from a replay buffer. Thus, optimizing the policy π using SAC and training the model of the inverse action policy π' using the replay buffer and maximum likelihood are non-competing and non adversarial objectives, they alternately minimize the same objective in each part of Algorithm 1 and decrease the Kulback-Leibler Divergence in each step, ending in a minimum at convergence since the KLD is bounded by 0 from below.

The inequality from (C6) is based on the maximum likelihood objective and a "clean" replay buffer that contains only samples from the current policy. But it can be shown that it also holds for a replay buffer that contains a mixture of samples from the current and old policies: the inequality holds for the mixture distribution which contains a fraction α of the replay buffer which stems from the new policy and a fraction $1 - \alpha$ which stems from the old policies

(α depends on the size of the replay buffer and the number of new samples obtained in the current rollout). I.e. $p(RB(e+1)) = \alpha\pi_{\theta(e+1)} + (1-\alpha)p(RB(e))$. Since the old inverse action policy $\pi'_{\theta(e)}$ is the argmax of the maximum likelihood objective of the $1-\alpha$ fraction of RB(e+1) which is RB(e) it is better or equal than any other inverse action policy with regard to that previous replay buffer RB(e). Thus

$$\begin{aligned}&\sum_{i=0\ldots T-1} \mathbb{E}_{(s_i,a_i,s_{i+1})\sim RB(e)}[-\log \pi'_{\theta(e+1)}(a_i|s_{i+1},s_i)]\\ \geq &\sum_{i=0\ldots T-1} \mathbb{E}_{(s_i,a_i,s_{i+1})\sim RB(e)}[-\log \pi'_{\theta(e)}(a_i|s_{i+1},s_i)]\end{aligned} \tag{C8}$$

due to the maximum likelihood objective for $\pi'_{\theta(e+1)}$ on the data RB(e+1)) the following inequality follows:

$$\begin{aligned}&\sum_{i=0\ldots T-1} \mathbb{E}_{(s_i,a_i,s_{i+1})\sim RB(e+1)}[-\log \pi'_{\theta(e+1)}(a_i|s_{i+1},s_i)]\\ \leq &\sum_{i=0\ldots T-1} \mathbb{E}_{(s_i,a_i,s_{i+1})\sim RB(e+1)}[-\log \pi'_{\theta(e)}(a_i|s_{i+1},s_i)]\end{aligned} \tag{C9}$$

Also, by using the mixture definition of RB(e+1) one can rewrite an Expectation over RB(e+1) as follows:

$$\begin{aligned}\mathbb{E}_{(s_i,a_i,s_{i+1})\sim RB(e+1)}[f] = \;&\alpha\mathbb{E}_{(s_i,a_i,s_{i+1})\sim\pi_{\theta(e)}}[f]\\ &+(1-\alpha)\mathbb{E}_{(s_i,a_i,s_{i+1})\sim RB(e)}[f]\end{aligned} \tag{C10}$$

Using the expanded Expectation Inequality (C9) can be rewritten as follows:

$$\begin{aligned}&\sum_{i=0\ldots T-1} [\alpha\mathbb{E}_{(s_i,a_i,s_{i+1})\sim\pi_{\theta(e)}} - \log \pi'_{\theta(e+1)}(a_i|s_{i+1},s_i)\\ &\quad+(1-\alpha)\mathbb{E}_{(s_i,a_i,s_{i+1})\sim RB(e)} - \log \pi'_{\theta(e+1)}(a_i|s_{i+1},s_i)]\\ \leq &\sum_{i=0\ldots T-1} [\alpha\mathbb{E}_{(s_i,a_i,s_{i+1})\sim\pi_{\theta(e)}} - \log \pi'_{\theta(e)}(a_i|s_{i+1},s_i)\\ &\quad+(1-\alpha)\mathbb{E}_{(s_i,a_i,s_{i+1})\sim RB(e)} - \log \pi'_{\theta(e)}(a_i|s_{i+1},s_i)]\end{aligned} \tag{C11}$$

By using Inequality (C8) it can be rewritten to

$$\begin{aligned}&\sum_{i=0\ldots T-1} [\alpha\mathbb{E}_{(s_i,a_i,s_{i+1})\sim\pi_{\theta(e)}} - \log \pi'_{\theta(e+1)}(a_i|s_{i+1},s_i)\\ &\quad+(1-\alpha)\mathbb{E}_{(s_i,a_i,s_{i+1})\sim RB(e)} - \log \pi'_{\theta(e+1)}(a_i|s_{i+1},s_i)]\\ \leq &\sum_{i=0\ldots T-1} [\alpha\mathbb{E}_{(s_i,a_i,s_{i+1})\sim\pi_{\theta(e)}} - \log \pi'_{\theta(e)}(a_i|s_{i+1},s_i)\\ &\quad+(1-\alpha)\mathbb{E}_{(s_i,a_i,s_{i+1})\sim RB(e)} - \log \pi'_{\theta(e+1)}(a_i|s_{i+1},s_i)]\end{aligned} \tag{C12}$$

which implies (by subtracting the common $1-\alpha$ term from both sides)

$$\begin{aligned}&\sum_{i=0\ldots T-1} [\alpha\mathbb{E}_{(s_i,a_i,s_{i+1})\sim\pi_{\theta(e)}} - \log \pi'_{\theta(e+1)}(a_i|s_{i+1},s_i)\\ \leq &\sum_{i=0\ldots T-1} [\alpha\mathbb{E}_{(s_i,a_i,s_{i+1})\sim\pi_{\theta(e)}} - \log \pi'_{\theta(e)}(a_i|s_{i+1},s_i)\end{aligned} \tag{C13}$$

which is Inequality (C6) multiplied by α and thus also implies (together with Inequality (C5)) Inequality (C7). From this follows the convergence of Algorithm 1 to a minimum when using a mixed replay buffer.

Appendix D: Implementation details

We use the same policy implementation for all our SOIL-TDM experiments. The stochastic policies $\pi_\theta(a_i|s_i)$ are modeled as a diagonal Gaussian to estimate continuous actions with two hidden layers (512, 512) with ReLU nonlinearities.

To train a policy using SAC as the RL algorithm, we also need to model a Q-function. Our implementation of SAC is based on the original implementation from Haarnoja et al. [17] and the used hyperparameters are described in Table 1. In this implementation, they use two identical Q-Functions with different initialization to stabilize the training process. These Q-Functions are also modeled with an MLP having two hidden layers (512, 512) and Leaky ReLU. We kept the entropy parameter α fixed to 1 and did not apply automatic entropy tuning as described by Haarnoja et al. [17].

We implement all our models for SOIL-TDM using the PyTorch framework version 1.9.0.[3] To estimate the imitation reward in SOIL-TDM a model for the expert transitions $\mu_E(s'|s)$ as well as a forward $\mu_\phi(s'|s,a)$ and backward dynamics model $\mu_\eta(a|s',s)$ has to be learned. All three density models are based on RealNVPs [34] consisting of several flow blocks where MLPs preprocess the conditions to a smaller condition size. The RealNVP transformation parameters are also calculated using MLPs, which process a concatenation of the input and the condition features. After each flow block, we added activation normalization layers like [35]. To implement these models, we use the publicly available VLL-FrEIA[4] framework version 0.2 [36] using their implementation of GLOWCouplingBlocks with exponent clamping activated. The setup for each model is layed out in Table 2. We add Gaussian noise to the state vector as a regularizer for the training of the expert transition model $\mu_E(s'|s)$ which increased training stability for a low amount

[3] https://github.com/pytorch/pytorch

[4] https://github.com/VLL-HD/FrEIA (Open Source MIT License)

Table 1 Training Hyperparameter

SAC Parameter	Value
Optimizer	Adam
learning rate policy	$1 \cdot 10^{-4}$
learning rate Q-function	$3 \cdot 10^{-4}$
discount γ (Halfcheetah, Walker2D)	0.7
discount γ	0.9
mini batch size	2048
replay buffer size	$1 \cdot 10^5$
target update interval	1
number of environments	16
max number of environment steps	$4.0 \cdot 10^6$
SOIL-TDM Parameter	Value
expert transition model training steps	$10^3 - 10^4$
learning rate expert transition model	$1 \cdot 10^{-4}$
learning rate forward dynamics model	$1 \cdot 10^{-4}$
learning rate backward dynamics model	$1 \cdot 10^{-4}$
update interval dynamics models	1

of expert trajectories. We implement a linear decrease of the standard deviation from 0.05 to 0.005 during the training of the expert model.

We train and test all algorithms on a computer with 8 CPU cores, 64 GB of working memory, and an RTX2080 Ti Graphics card. The computing time for the SOIL-TDM method depends on the time to convergence and is from 4h to 14h.

Appendix E: Expert data generation and baseline methods

The expert data is generated by training an expert policy based on conditional normalizing flows and the SAC algorithm on the environment reward. A conditional normalizing flow policy has been chosen for the expert to make the distribution matching problem for the baseline methods and

Table 2 Normalizing Flow Setup

	$\mu^E(s'\|s)$	$\mu_\phi(s'\|s,a)$	$\mu_\eta(a\|s',s)$
N flow blocks	16	16	16
Condition hidden neurons	64	48	256
Condition hidden layer	2	2	2
Condition feature size	32	32	32
Flow block hidden neurons	64	48	256
Flow block hidden layer	2	2	2
Exponent clamping	6	1	1

Table 3 Episode Reward of Expert Policy

Environment	Average Expert Episode Reward
AntBulletEnv-v0	2583
HalfCheetahBulletEnv-v0	2272
HopperBulletEnv-v0	2357
Walker2DBulletEnv-v0	1805
HumanoidBulletEnv-v0	2750

SOIL-TDM - which employ a conditional Gaussian policy - more challenging and more similar to real-world IL settings. The idea is that real-world demonstrations might be more complex and experiments using the same policy setup for the expert and the imitator might not translate well to real-world tasks.

The stochastic flow policy $\pi_\theta(a_i|s_i)$ is based on Real-NVPs [34] which have the same setup as the normalizing flow implementations used for $\mu^E(s'|s)$, $\mu_\phi(s'|s,a)$, and $\mu_\eta(a|s',s)$ also using N=16 GLOWCouplingBlocks. The state is processed as a condition with one MLP having a hidden size of 128. Each flow block has an additional fully connected layer to further process the condition to a small feature size of 8. Every flow block has 128 hidden neurons. Finally, each action is passed through a tanh layer as it is described in the SAC implementation. The log probability calculation was adapted accordingly [17]. The final episode reward of the trained expert policy is in Table 3.

The expert trajectories are generated using the trained policy and saved as done by [2].[5] For the OPOLO[6] and F-IRL[7] baseline, the original implementations with the official default parameters for each environment are used. Only the loading of the expert data was changed to use the demonstrations of the previously trained normalizing flow policy. Since no official code for FORM was publicly available, the FORM baseline was implemented based on our method. We changed the reward to use the state prediction effect model $\mu_\Phi^{\pi_\theta}(s'|s)$ as proposed by Jaegle et al. [10]. Both effect models were implemented using the same implementation as for our expert transition model $\mu^E(s'|s)$. Training of the expert effect model $\mu_\omega^E(s'|s)$ was performed offline with the same hyperparameter setup used for our expert transition model $\mu^E(s'|s)$ (see Tables 1 and 2). The policy optimization was done using the same SAC setup as for our method since FORM does not depend on a specific RL algorithm [10]. We tested different setups for the entropy parameter α and found that automatic entropy tuning as described by Haarnoja et al. [17] worked best.

[5] https://github.com/openai/imitation (Open Source MIT License)

[6] https://github.com/illidanlab/opolo-code (Open Source MIT License)

[7] https://github.com/twni2016/f-IRL (Open Source MIT License)

It should be noted that training the expert policy using SAC might introduce a bias since FORM and our method SOIL-TDM is also trained with SAC. But the results in Appendix 6 show that our method still performs reliably if the expert is trained with another Reinforcement Learning algorithm, namely PPO. We, therefore, argue that using SAC for training the expert did not influence the results in favor of our proposed method. In addition, using a stochastic flow policy for the SAC expert data generation makes the task more difficult for all tested imitation learning methods since all methods (including FORM and our methods SOIL-TDM) used conditional Gaussian policies.

For our experiment with an unknown true environment reward, the following selection criteria are used. For "OPOLO est. reward" the estimated return based on the reward $r(s, s') = -log(1 - D(s, s'))$ with the state s, next state s' and the discriminator output $D(s, s')$ is used. For "OPOLO pol. loss" the original OPOLO policy loss is used:

$$\begin{aligned} J(\pi_\theta, Q) = (1-\gamma)\ & \mathbb{E}_{s\sim S_0}[Q(s, \pi_\theta(s))] + \\ & \mathbb{E}_{(s,a,s')\sim R}[f(r(s, s') + \\ & \gamma Q(s', \pi_\theta(s')) - Q(s, a))] \qquad \text{(C14)} \end{aligned}$$

With the Q-Function Q and the f-divergence function. For both estimates, the original OPOLO implementation was used. For FORM the convergence was estimated with the estimated return based on the reward: $r_t = log\mu_\omega^E(s'|s) - log\mu_\Phi^{\pi_\theta}(s'|s)$ using the normalizing flow effect models. For F-IRL we use the implementation of Ni et al. [16] for the estimate of the Jensen-Shannon divergence between state marginals of the expert and policy

$$\frac{1}{2}\int p(x)log\frac{2p(x)}{p(x)+q(x)} + q(x)log\frac{2q(x)}{p(x)+q(x)}dx.$$

Appendix F: Ablation study

In this section, we want to investigate the influence of different components on our proposed imitation learning setup. First, we want to answer whether learning additional backward and forward dynamics models to estimate the state transition KLD improves policy performance. We compare our proposed method SOIL-TDM to an approach where we train a policy only based on the log-likelihood of the expert

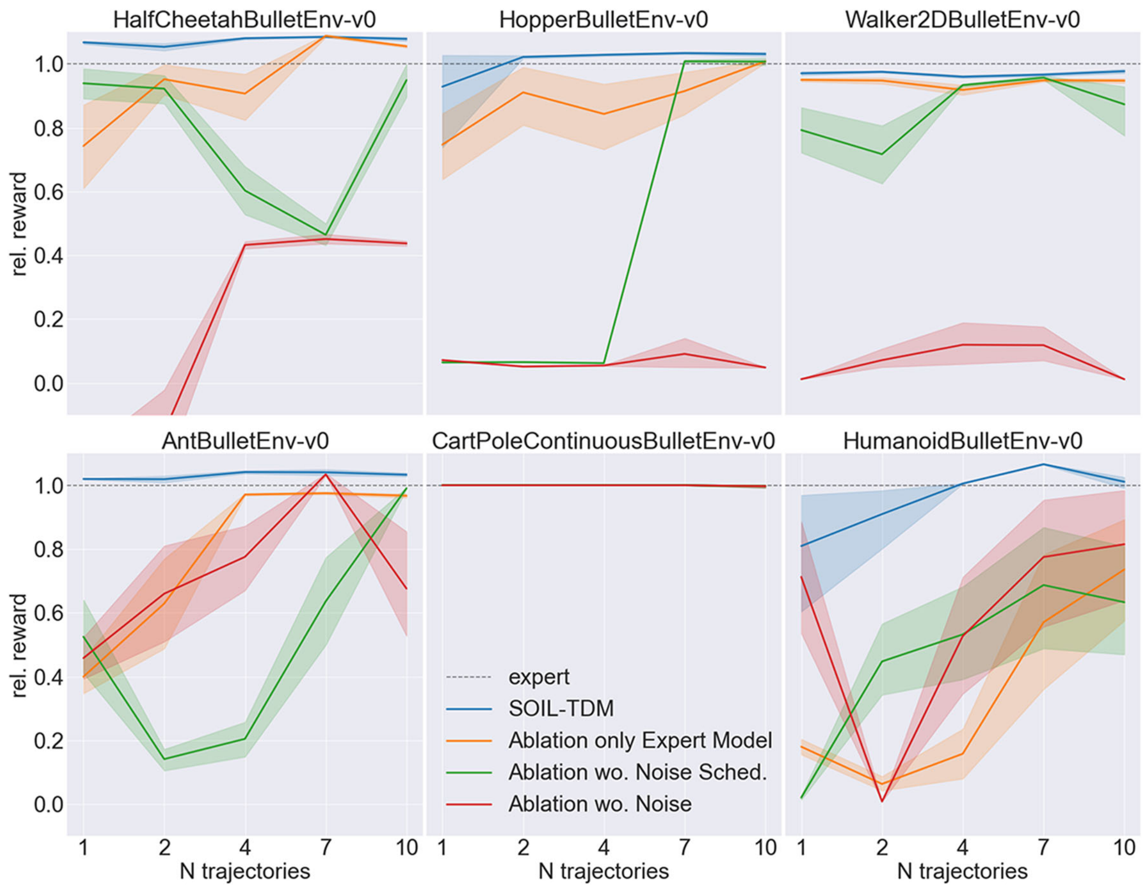


Fig. 3 Best environment reward for ablation experiments. The relative reward for a different amount of expert trajectories. The value 1 corresponds to expert policy performance

Table 4 Test log-likelihood values of expert transition models $\mu^E(s'|s)$ for 1, 2, 4, 7, and 10 training trajectories using 20 unknown test trajectories

Environment	Log-Likelihood for $\mu^E(s'|s)$
Ant	38.5, 43.5, 54.2, 59.9, 62.7
HalfCheetah	45.5, 47.8, 70.2, 71.1, 71.3
Hopper	14.4, 13.8, 15.9, 40.7, 36.9
Walker	38.9, 47.4, 62.5, 65.2, 67.4
Humanoid	39.5, 50.9, 62.9, 77.7, 77.4

Table 6 Test log-likelihood values of expert transition models $\mu^E_{abl2}(s'|s)$ for 1, 2, 4, 7, and 10 training trajectories using 20 unknown test trajectories

Environment	Log-Likelihood for $\mu^E_{abl2}(s'|s)$ (no noise)
Ant	28.7, 31.9, 37.3, 37.4, 38.9
HalfCheetah	$-inf$, $-inf$, −270.2, −194.2, −187.8
Hopper	−171.3, −629.0, −379.1, −87.1, −70.9
Walker	$-inf$, $-inf$, −389.0, −182.4, −180.1
Humanoid	25.9, 36.7, 46.4, 50.1, 52.1

defined by:

$$r_{abl}(s, s') = \log \mu^E(s'|s) \tag{C15}$$

Using this reward the policy is optimized using SAC as described earlier. We call this simplified reward design approach "Ablation only Expert Model". By comparing the performance of this method to our approach, we can show that learning additional density models to estimate forward and backward dynamics leads to improved policy performance. The resulting rewards are plotted in Fig. 3. The relative reward using this ablation method is much lower compared to SOIL-TDM. Only for a high amount of trajectories does this method reach expert-level performance.

We furthermore want to evaluate how the quality of the learned normalizing flows affects the overall algorithm performance. We, therefore, report the estimated test log-likelihood of the trained expert models $\mu^E(s'|s)$ for a different amount of expert trajectories using a separate test dataset with 20 unseen expert trajectories in Table 4. The influence of the expert model quality on the overall algorithm performance can be seen by comparing the test log-likelihood to the reward of the trained policy (in Fig. 3 "SOIL-TDM" for our method and "Ablation only Expert Model" for the simplified reward using the same expert model $\mu^E(s'|s)$). SOIL-TDM is less sensitive to expert model performance compared to "Ablation only Expert Model".

Table 5 Test log-likelihood values of expert transition models $\mu^E_{abl1}(s'|s)$ for 1, 2, 4, 7, and 10 training trajectories using 20 unknown test trajectories

Environment	Log-Likelihood for $\mu^E_{abl1}(s'|s)$ (constant noise)
Ant	30.9, 30.8, 41.1, 41.1, 59.0
HalfCheetah	26.3, 25.1, 78.3, 77.9, 79.2
Hopper	−31.6, −26.9, −24.1, 41.6, 40.8
Walker	42.1, 46.1, 55.8, 56.8, 58.1
Humanoid	26.4, 38.0, 51.5, 54.6, 57.4

Lastly, we want to investigate the expert model training. In the improved expert model training Routine, we regularize the optimization by adding Gaussian noise to the expert state values and linearly decrease its standard deviation down to 0.005 during training. As an additional ablation, we tested 2 different training setups for the expert transition models $\mu^E(s'|s)$. For the first model ($\mu^E_{abl1}(s'|s)$) we omit the noise decay and only use the final constant Gaussian noise during training. For the second model ($\mu^E_{abl2}(s'|s)$) no noise is added during the training of the expert model. We also evaluated the test log-likelihood of the trained models using the test dataset with 20 unseen expert trajectories. The resulting test log-likelihoods are in Tables 5–6. The results show that constant Gaussian noise already improves the performance of the expert model and our applied noise scheduling routine results in further performance improvements. We also used the trained expert models for policy training based on the ablation reward from (C15). These ablation methods are called "Ablation wo. Noise Sched.", "Ablation wo. Noise" respectively. The final rewards of these methods are also plotted in Fig. 3. The results indicate that adding noise to the states during the offline training of the expert transition model also improves final policy performance.

Appendix G: Incomplete and mixed expert demonstrations

In this section, we investigate the influence of using different experts to generate demonstrations. In the following experiments, the expert data comes from two differently trained experts. The first expert is the same conditional normalizing flow policy $\pi_\theta(a_i|s_i)$ trained with SAC as in the original experiments (See Section 6). The second expert is

Table 7 Average Episode Reward of Expert Policies

Environment	Average Expert Episode Reward
HopperBulletEnv-v0	2546
HumanoidBulletEnv-v0	2682

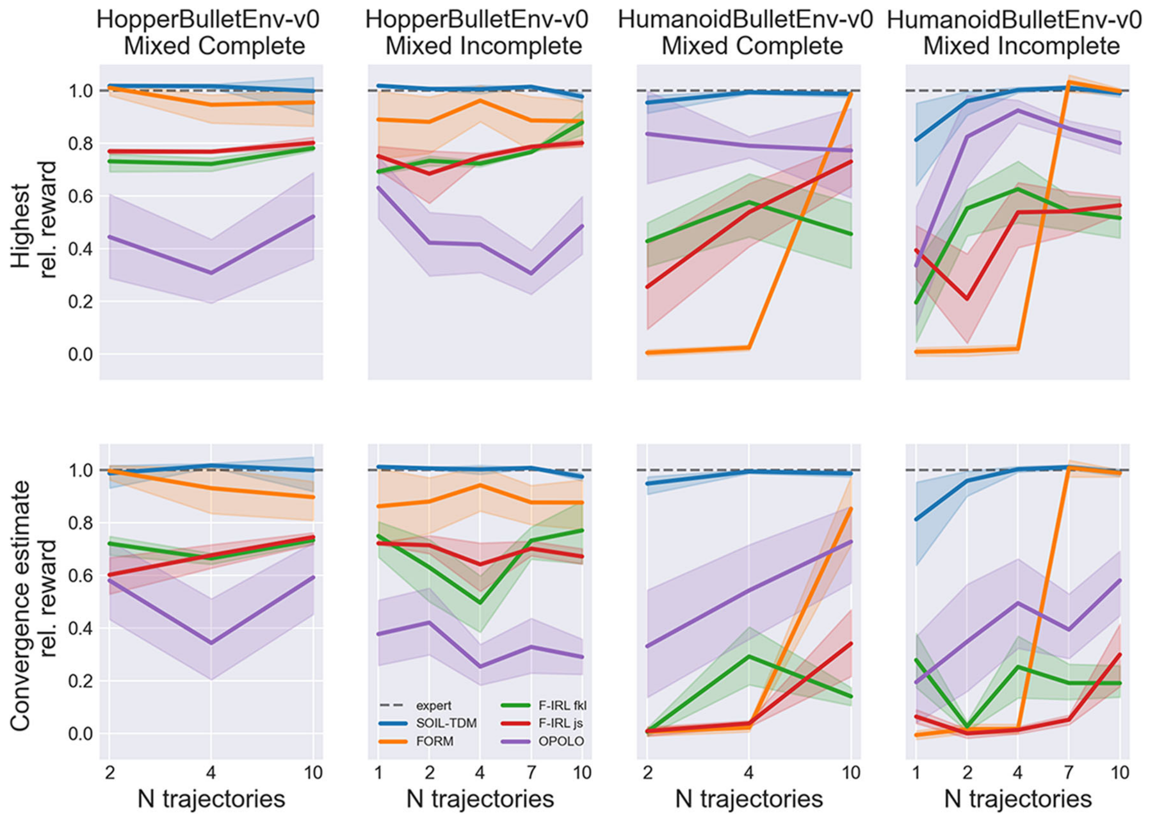


Fig. 4 Comparison of the different imitation learning methods using demonstrations from two different expert policy action distributions for each environment. The figure shows the relative reward for a different amount of expert trajectories for the HopperBulletEnv-v0 (left) and HumanoidBulletEnv-v0 (right) environment. The value 1 corresponds to expert policy performance

a conditional Gaussian policy $\pi_{\theta_2}(a_i|s_i)$ trained with Proximal Policy Optimization (PPO)[8] on the environment reward. The mean and standard deviation of the Gaussian distribution used to sample the action is conditioned on the state. The state is processed using a MLP having two hidden layers with a hidden size of 256.

This results in two different expert policy action distributions for one environment. We then combined the expert demonstrations in two different ways. In the first approach ("Mixed Complete"), we combined the complete trajectories of both policies (resulting in 2, 4, 10 trajectories) and in the second approach ("Mixed Incomplete"), we removed 50% of every trajectory from both experts and concatenated the trajectories of both (resulting in 1, 2, 4, 7, 10 trajectories). Hence, in the second approach, we have the same amount of demonstrations (1000 for each amount of selected trajectories) where the data is incomplete and comes from different expert policy distributions. The Experiments were performed on the HopperBulletEnv-v0 and HumanoidBulletEnv-v0 environments for OPOLO, F-IRL, FORM, and SOIL-TDM (our method).

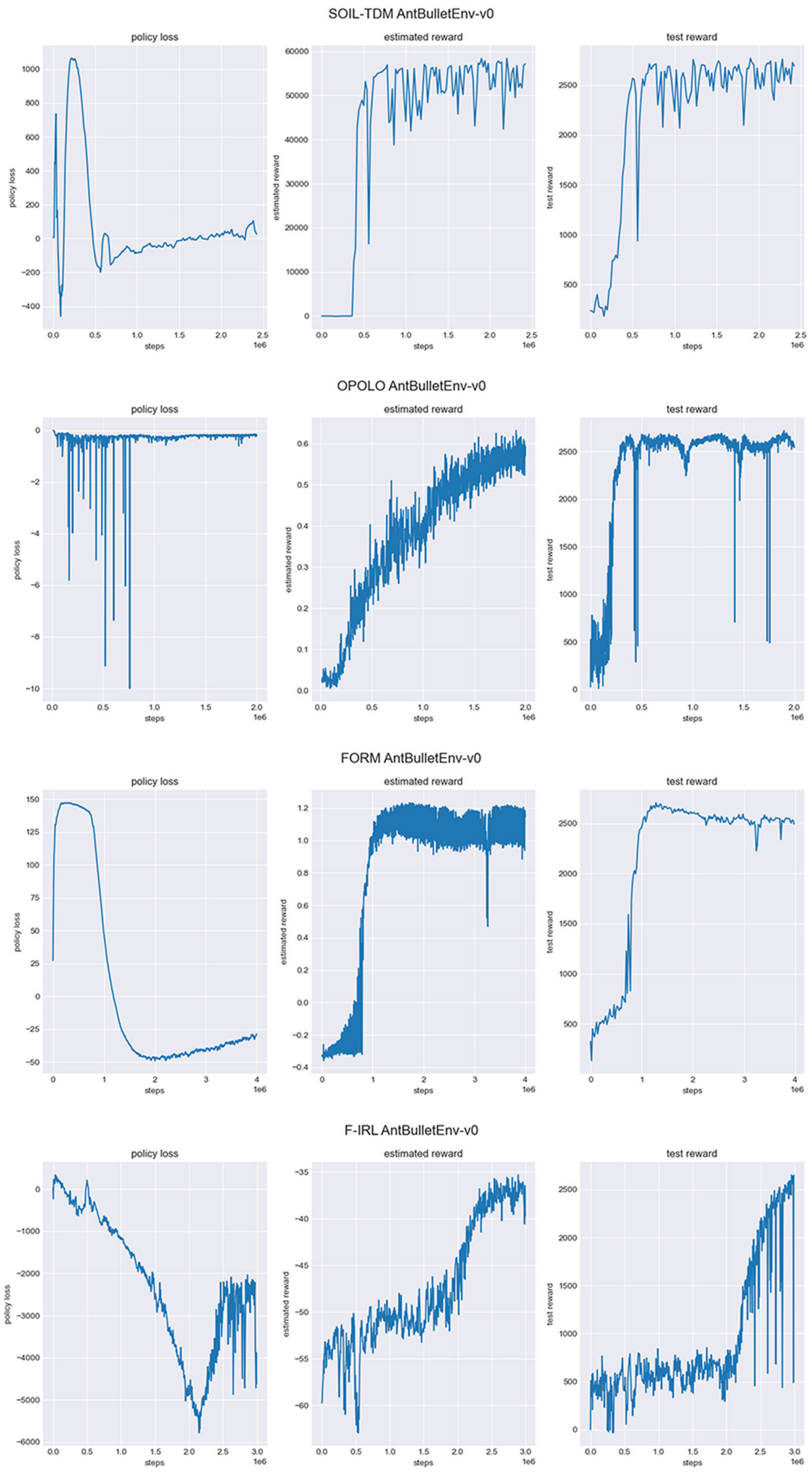
The resulting rewards for the "Mixed Complete" and "Mixed Incomplete" datasets are plotted in Fig. 4. It shows the highest relative reward and the relative reward based on policies selected using the previously introduced convergence estimates (the different performance estimation methods are described in Appendix E). The learning problem should be harder for both datasets since there is no consistent expert behavior to observe. However, the performance of FORM improved compared to the original experiments. SOIL-TDM (our method) performs similarly well to the experiments using one expert policy for each environment and is still better or competitive with the baseline methods. In the HopperBulletEnv-v0 environment, OPOLO performs worse, while the resulting rewards are higher for F-IRL in this setup. In Summary, methods based on conditional state probabilities (FORM and SOIL-TDM) showed consistent or improved performance in this experimental setup. This suggests that these methods might be well suited for tasks where the demonstrations come from multiple experts.

Appendix H: Additional Results

The following figures (Figs. 5– 9) show the policy loss and the estimated reward together with the environment reward during the training on different pybullet environments for

[8] based on https://github.com/openai/baselines

Fig. 5 The policy loss, estimated reward and the environment test loss during training in the pybullet Ant environment using our proposed SOIL-TDM and the OPOLO, F-IRL, and FORM implementations with 4 expert trajectories



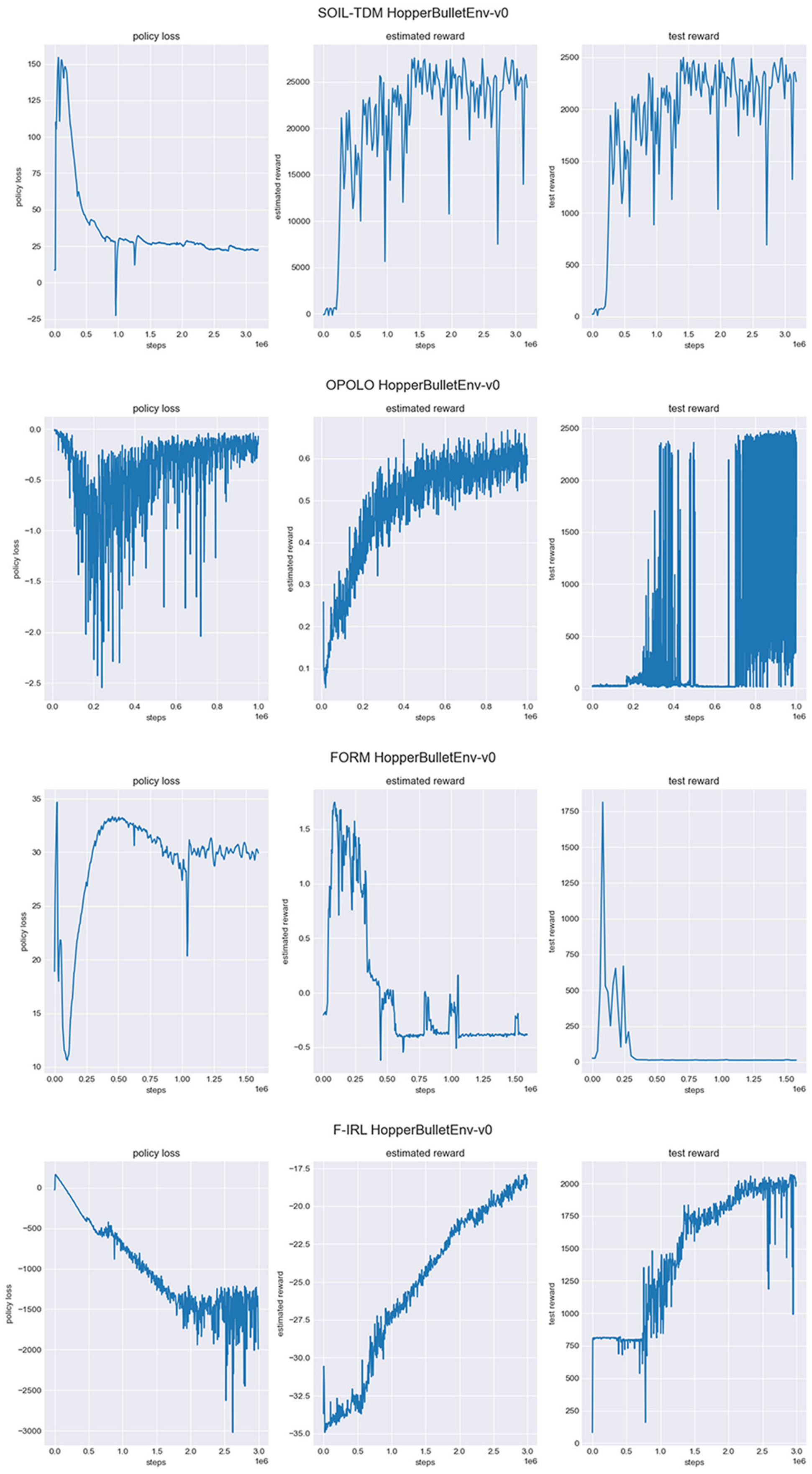


Fig. 6 The policy loss, estimated reward and the environment test reward during training in the pybullet Hopper environment using our proposed SOIL-TDM and the OPOLO, F-IRL, and FORM implementations with 4 expert trajectories

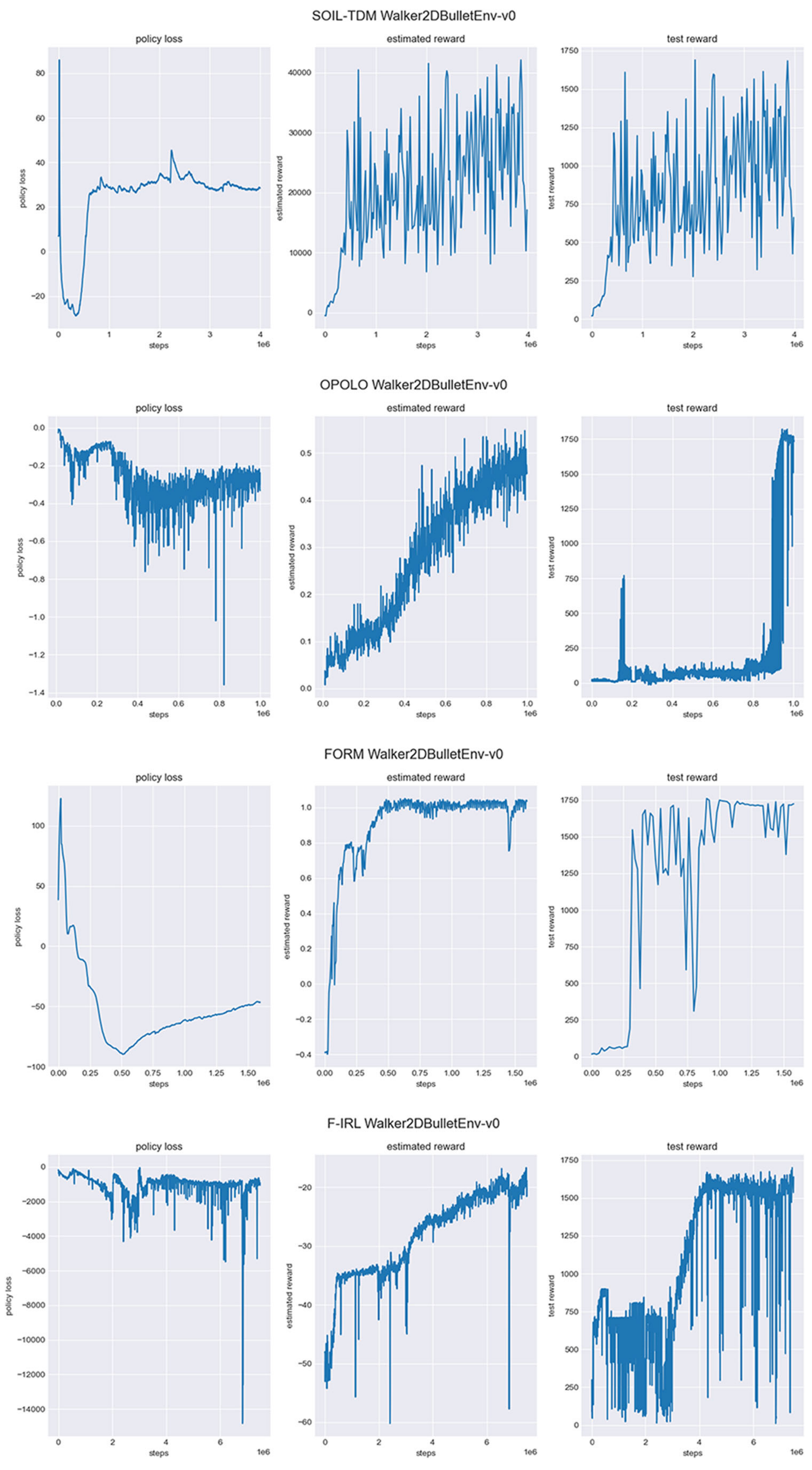


Fig. 7 The policy loss, estimated reward, and the environment test reward during training in the pybullet Walker2D environment using our proposed SOIL-TDM and the OPOLO, F-IRL, and FORM implementations with 4 expert trajectories

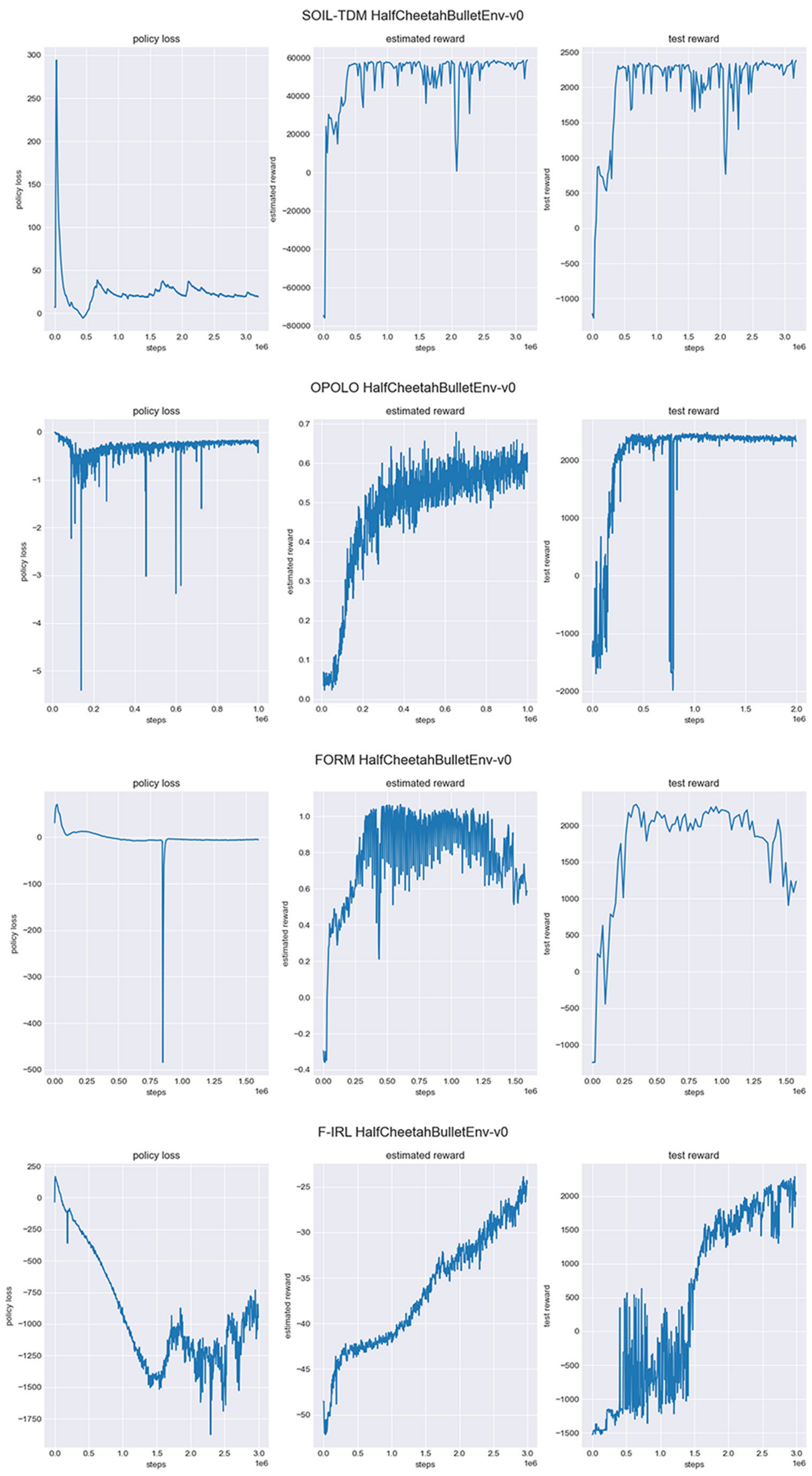
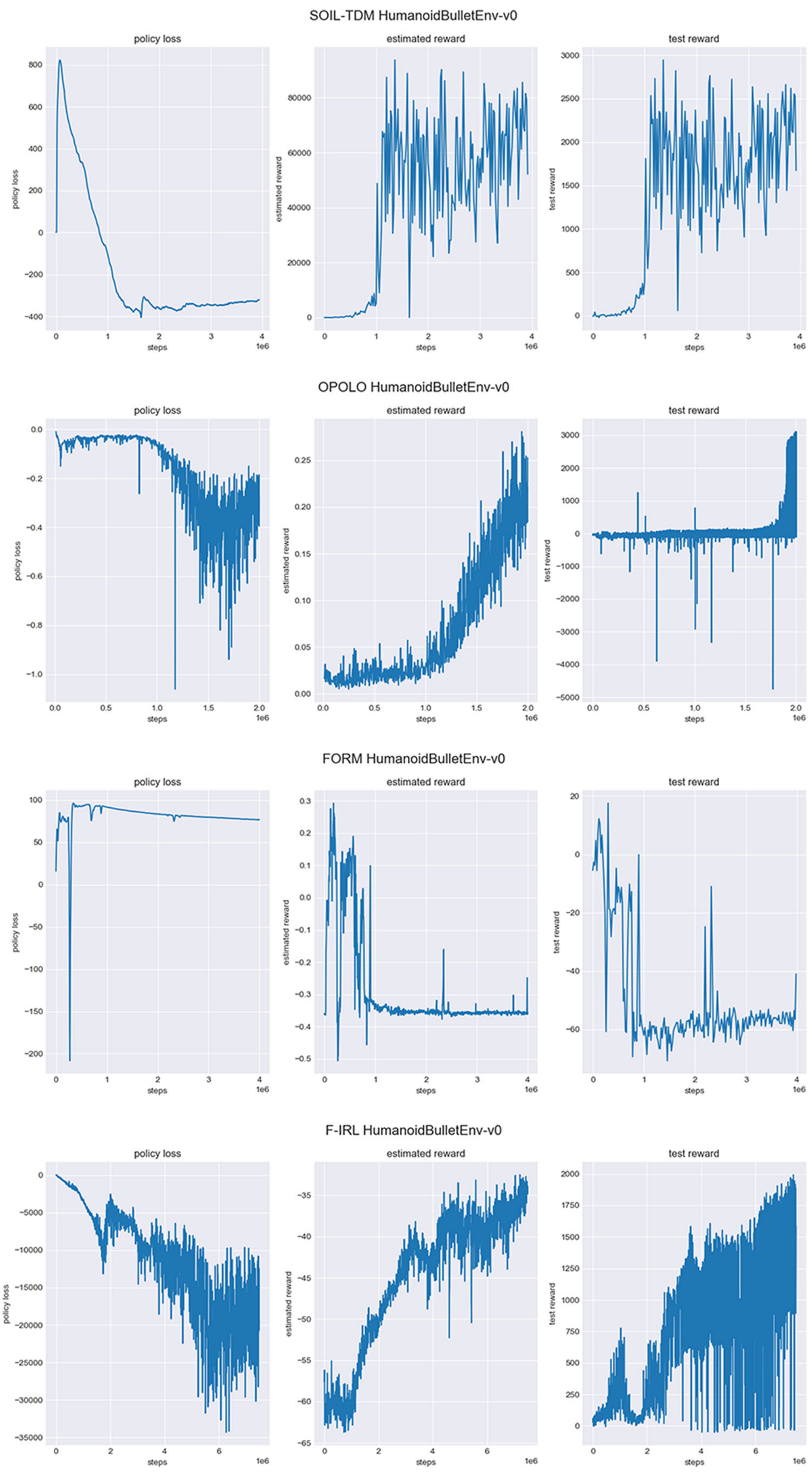


Fig. 8 The policy loss, estimated reward, and the environment test reward during training in the pybullet HalfCheetah environment using our proposed SOIL-TDM and the OPOLO, F-IRL, and FORM implementations with 4 expert trajectories

Fig. 9 The policy loss, estimated reward, and the environment test reward during training in the pybullet Humanoid environment using our proposed SOIL-TDM and the OPOLO, F-IRL, and FORM implementations with 4 expert trajectories



OPOLO, F-IRL, FORM, and SOIL-TDM (our method). All plots have been generated from training runs with 4 expert trajectories and 10 test rollouts. It can be seen that the estimated reward and policy loss from SOIL-TDM correlates well with the true environment reward. The policy loss of SOIL-TDM may be lower than 0 since it is not based on the true distributions. Instead, it is based on learned and inferred estimates of expert state conditional distribution, policy state conditional distribution, policy inverse action distribution and q-function with relatively large absolute values ($\sim 50-100$) each. These estimation errors accumulate in each time step due to sum and subtraction and due to the Q-function also over (on average) 500 timesteps which can lead to relatively large negative values.

Data Availibility Statement The datasets generated during the current study are available from the corresponding author upon reasonable request. We describe the data generation process using publicly available resources in Appendix E.

Declarations

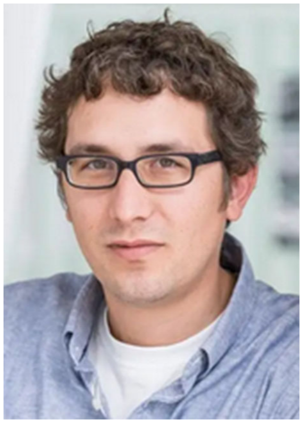
Funding and Competing Interests Statement Open Access funding enabled and organized by Projekt DEAL. The authors did not receive additional support from any organization for the submitted work. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Kuefler A, Morton J, Wheeler T, Kochenderfer M (2017) Imitating driver behavior with generative adversarial networks. In: 2017 IEEE intelligent vehicles symposium (IV)
2. Ho J, Ermon S (2016) Generative adversarial imitation learning. In: Advances in neural information processing systems
3. Osa T, Pajarinen J, Neumann G, Bagnell JA, Abbeel P, Peters J (2018) An algorithmic perspective on imitation learning. In: Foundations and trends in robotics
4. Torabi F, Warnell G, Stone P (2019)Recent advances in imitation learning from observation. In: Proceedings of the 28th international joint conference on artificial intelligence
5. Torabi F, Warnell G, Stone P (2018) Generative adversarial imitation from observation. In: International conference on machine learning workshop on imitation, intent, and interaction (I3)
6. Miyato T, Kataoka T, Koyama M, Yoshida Y (2018) Spectral normalization for generative adversarial networks. In: International conference on learning representations, ICLR
7. Jin C, Netrapalli P, Jordan M (2020) What is local optimality in nonconvex-nonconcave minimax optimization?. In: Proceedings of the 37th international conference on machine learning
8. Grnarova P, Levy KY, Lucchi A, Perraudin N, Goodfellow I, Hofmann T, Krause A (2019) A domain agnostic measure for monitoring and evaluating gans. In: Advances in neural information processing systems
9. Sidheekh S, Aimen A, Madan V, Krishnan NC (2021) On duality gap as a measure for monitoring gan training. In: 2021 international joint conference on neural networks (IJCNN)
10. Jaegle A, Sulsky Y, Ahuja A, Bruce J, Fergus R, Wayne G (2021) Imitation by predicting observations. In: Proceedings of the 38th international conference on machine learning
11. Ziebart BD (2010) Modeling purposeful adaptive behavior with the principle of maximum causal entropy. PhD thesis
12. Zhu Z, Lin K, Dai B, Zhou J (2020) Off-policy imitation learning from observations. In: Advances in neural information processing systems
13. Torabi F, Warnell G, Stone P (2018) Behavioral cloning from observation. In: Proceedings of the 27th international joint conference on artificial intelligence
14. Papamakarios G, Nalisnick ET, Rezende DJ, Mohamed S, Lakshminarayanan B (2019) Normalizing flows for probabilistic modeling and inference. J Mach Learn Res
15. Haarnoja T, Zhou A, Abbeel P, Levine S (2018) Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proceedings of the 35th international conference on machine learning
16. Ni T, Sikchi H, Wang Y, Gupta T, Lee L, Eysenbach B (2020) f-irl: inverse reinforcement learning via state marginal matching. In: Conference on robot learning
17. Haarnoja T, Zhou A, Hartikainen K, Tucker G, Ha S, Tan J, Kumar V, Zhu H, Gupta A, Abbeel P, Levine S (2018) Soft actor-critic algorithms and applications. Technical report
18. Ghasemipour SKS, Zemel RS, Gu S (2019) A divergence minimization perspective on imitation learning methods. In: 3rd annual conference on robot learning, CoRL. Proceedings of machine learning research
19. Fu J, Luo K, Levine S (2017) Learning robust rewards with adversarial inverse reinforcement learning. In: International conference on learning representations, ICLR
20. Ng A, Russell S (2000) Algorithms for inverse reinforcement learning. In: International conference on machine learning
21. Kostrikov I, Nachum O, Tompson J (2020) Imitation learning via off-policy distribution matching. In: International conference on learning representations, ICLR
22. Wang R, Ciliberto C, Amadori PV, Demiris Y (2019) Random expert distillation: Imitation learning via expert policy support estimation. In: Proceedings of the 36th International conference on machine learning
23. Brantley K, Sun W, Henaff M (2020) Disagreement-regularized imitation learning. In: International conference on learning representations, ICLR
24. Liu M, He T, Xu M, Zhang W (2020) Energy-based imitation learning. Autonomous agents and multi-agent systems
25. Kim K, Jindal A, Song Y, Song J, Sui Y, Ermon S (2021) Imitation with neural density models. In: Advances in neural information processing systems
26. Sun M, Devlin S, Hofmann K, Whiteson S (2021) Deterministic and discriminative imitation (d2-imitation): revisiting adversarial imitation for sample efficiency. In: Association for the advancement of artificial intelligence

27. Garg D, Chakraborty S, Cundy C, Song J, Ermon S (2021) IQ-learn: Inverse soft-q learning for imitation. In: 35th conference on neural information processing systems
28. Yang C, Ma X, Huang W, Sun F, Liu H, Huang J, Gan C (2019) Imitation learning from observations by minimizing inverse dynamics disagreement. In: Advances in neural information processing systems
29. Sun W, Vemula A, Boots B, Bagnell D (2019) Provably efficient imitation learning from observation alone. In: Proceedings of the 36th International conference on machine learning
30. Schroecker Y, Vecerík M, Scholz J (2019) Generative predecessor models for sample-efficient imitation learning. In: International conference on learning representations, ICLR
31. Jiang S, Pang J, Yu Y (2020) Offline imitation learning with a misspecified simulator. In: Advances in neural information processing systems
32. Coumans E, Bai Y (2016) PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org
33. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: International conference on learning representations, ICLR
34. Dinh L, Sohl-Dickstein J, Bengio S (2017) Density estimation using real nvp. In: International conference on learning representations
35. Kingma DP, Dhariwal P (2018) Glow: Generative flow with invertible 1x1 convolutions. In: Advances in neural information processing systems
36. Ardizzone L, Kruse J, Rother C, Köthe U (2019) Analyzing inverse problems with invertible neural networks. In: International conference on learning representations, ICLR. https://github.com/VLL-HD/FrEIA

Damian Boborzi received the B.S. degree in Electrical Engineering and Automation Technology from DHBW Stuttgart, in collaboration with Eisenmann SE, a plant manufacturer and automotive industry supplier, Stuttgart, Germany, in 2015. He completed his Master's degree in Electrical Engineering and Information Technology at the University of Stuttgart, Germany, in 2018. Since 2021, he has been pursuing his Ph.D. in computer science, supervised by the Chair of Mechatronics at the University of Augsburg, Augsburg, Germany. From 2018 to 2021, he was a participant in the Bosch Doctoral Program at the Bosch subsidiary, ETAS GmbH. Concurrently, since 2021, he serves as a Research Associate at the Chair of Mechatronics, University of Augsburg. His current research interests are at the intersection of machine learning methods and their real-world applications. Specifically, he has focused on the generation of virtual traffic participants from traffic measurement data using machine learning. He is also deeply involved in the exploration of generative models, notably Normalizing Flows, and their integration with Reinforcement Learning and Imitation Learning paradigms.



Christoph-Nikolas Straehle received a Diploma degree in Physics in 2011 and a PhD in natural sciences in 2015 from the University of Heidelberg. Since then he worked as a research scientist in Bosch Corporate Research and at the Bosch Center for Artificial intelligence. Current research interests include reinforcement learning, probabilistic models and uncertainty quantification in the area of autonomous driving and computer vision.

Jens S. Buchner received a diploma degree in Physics in 2009 and a PhD in Physic and Computer Science in 2012 from Heidelberg University. After holding various positions at ETAS GmbH (within the Bosch Group) he joined the Machine Learning Team at ETAS in 2018 as an Expert Machine Learning. After acting as a Senior Expert Machine Learning for Embedded from 2020, he accepted a position at the beginning of 2022 as a Senior Manager and Product Owner for Machine Learning Operations in the field of Automated Driving at Bosch. He currently focusses his interests on the methods and tools in the field of computer vision and other artificial intelligence-based methods in the automotive stack.

Lars Mikelsons holds a diploma in mathematics from the University of Duisburg-Essen, earned in 2007. He furthered his academic pursuits, securing a Ph.D. in Mechatronics in 2011. His professional journey includes a significant period at Bosch, where he served as a researcher and research project leader from 2011 to 2018. During this time, he made substantial contributions to the field of mechatronics, gaining recognition for his expertise and innovative approach. In 2018, Mikelsons transitioned to academia, assuming the role of Head of the Chair for Mechatronics at the University of Augsburg. His current research interest is at the forefront of model-based validation and development, with a specialized focus on the fields of Scientific Machine Learning, Neural Ordinary Differential Equations (NeuralODEs), and Reinforcement Learning.