STATE-ONLY IMITATION LEARNING BY TRAJECTORY DISTRIBUTION MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

The best performing state-only imitation learning approaches are based on adversarial imitation learning. The main drawback, however, is that adversarial training is often unstable and lacks a reliable convergence estimator. When 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, with an interpretable convergence and performance metric.

Our training objective minimizes the Kulback-Leibler divergence between the policy and expert state transition trajectories which can be optimized in a non-adversarial fashion. For this, additional density models estimate the expert state transition distribution and the environment's forward and backward dynamics. We demonstrate the effectiveness of our approach on well-known continuous control environments, where our method can generalize to expert performance. We demonstrate that our method and loss are better suited to select the best-performing policy compared to objectives from adversarial methods by being competitive to or outperforming the state-of-the-art learning-from-observation approach in these environments.

1 INTRODUCTION

Imitation learning (IL) describes methods that learn optimal behavior that is represented by a collection of expert demonstrations. While in standard reinforcement learning (RL), the agent is trained on environment feedback, IL can alleviate the problem of designing effective reward functions. Thus especially useful for tasks where demonstrations are easier to access compared to designing a reward function. One popular example is to train traffic agents in a simulation to mimic real-world road users (Kuefler et al., 2017).

Learning from demonstrations (LfD) describes IL approaches that require state-action pairs from expert demonstrations. Many promising approaches have been proposed which match state-action distributions of the expert and the learner using an adversarial training setup (Ho & Ermon, 2016; Fu et al., 2017; Kostrikov et al., 2019; 2020).

While 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 recordings of traffic in bird's eye view (Kuefler et al., 2017). 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 (Osa et al., 2018). 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 observation (LfO) performs state-only imitation learning and trains an agent without actions being available in the expert dataset (Torabi et al., 2019). While LfO is a more challenging task than LfD, it can be more practical for tasks with incomplete data sources. To learn a policy from observations learning an additional environment model may help to infer actions based on expert observations and the learned environment dynamics (Torabi et al., 2018a). Similarly to LfD distribution matching based on an adversarial setup is also used in LfO (Torabi et al., 2018b;

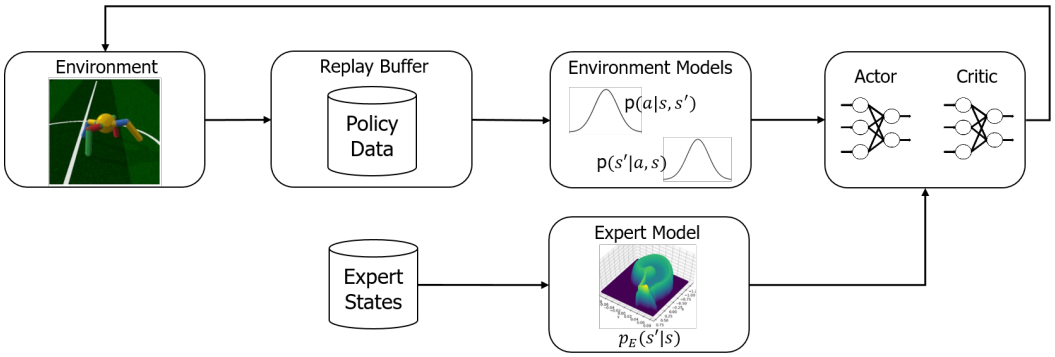

Figure 1: Overview of the proposed method. This setup is using off-policy data to train the policy and the environment models.

Yang et al., 2019; Zhu et al., 2020). In adversarial imitation learning (AIL) a policy is trained using an adversarial discriminator and a reinforcement learning method. The discriminator is used to estimate a reward which guides policy training. While AIL methods obtain better performing agents than supervised methods like behavioral cloning (BC) using less data, adversarial training often has stability issues (Miyato et al., 2018; Goodfellow, 2017) and under some conditions is not guaranteed to converge (Jin et al., 2020). Additionally, estimating the performance of a trained policy without access to the environment reward can be very challenging. While the duality gap (Grnarova et al., 2019; Sidheekh et al., 2021) 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.

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 Kulback Leibler divergence (KLD) between the actionless policy and expert trajectories by minimizing the KLDs of the conditional state transition distributions 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.

Additionally, to match the transition distributions of the policy and the expert, a forward dynamics and inverse action model of the environment is learned using normalizing flows. Normalizing flow models have been demonstrated to perform very well on learning complex probability distributions (Papamakarios et al., 2019). Combining all estimates results in an interpretable reward that can be used together with standard maximum entropy reinforcement learning methods (Ziebart, 2010; Haarnoja et al., 2018a). The optimization based on the KLD provides a reliable convergence metric of the training and a good estimator of the policy performance. An overview of our proposed method is given in Figure 1.

While Kim et al. (2021) also proposed a method that performs imitation learning using density estimates, their optimization is based on the learning from demonstrations objective using state-action pairs. We therefore compared our method to the recent state-of-the-art learning from observations approach OPOLO (Zhu et al., 2020) 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 which more closely resembles real-world applications in autonomous driving or robotics where it is difficult to define a reward function (Osa et al., 2018).

2 Background

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

[1] An extension to infinite horizon environments is given in section 3

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 $p(s_{t+1}|s_t, a_t)$ and the bounded reward function by $r(s_t, a_t)$. At every time step t the agent interacts with its environment by observing a state s_t and taking an action a_t. This results in a new state s_{t+1} and a reward signal r_{t+1} based on the transition probability and reward function. We will use $\mu^{\pi_\theta}(s_t, a_t)$ to denote the state-action marginals at time step t of the trajectory distribution induced by the policy $\pi_\theta(a_t|s_t)$.

2.1 Maximum Entropy Reinforcement Learning and Soft Actor Critic

The standard objective in reinforcement learning is the expected sum of undiscounted rewards $\sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\mu^{\pi_\theta}}[r(s_t, a_t)]$. The goal of the agent is to learn a policy $\pi_\theta(a_t|s_t)$ which maximises this objective. The maximum entropy objective (Ziebart, 2010) introduces a modified goal for the RL agent, where the agent has to maximise the sum of the reward signal and its output entropy $\mathcal{H}(\pi_\theta(\cdot|s))$. Resulting in the policy objective

$$J(\pi_\theta) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\mu^{\pi_\theta}}[r(s_t, a_t) + \alpha\mathcal{H}(\pi_\theta(\cdot|s))]. \tag{1}$$

The parameter α controls the the stochasticity of the optimal policy by determining the relative importance of the entropy term versus the reward.

Soft Actor-Critic (SAC) proposed by Haarnoja et al. (2018a;b) 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. For a fixed policy π, the soft Q-value can be computed iteratively, starting from any function $Q : S \times A \to \mathbb{R}$ and repeatedly applying a modified Bellman backup operator T^π given by:

$$T^\pi Q(s_t, a_t) = r(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p(s_{t+1}|s_t,a_t)}[V^\pi(s_{t+1})] \tag{2}$$

The parameter γ describes the discount factor. In soft policy iteration the soft state value function is defined by:

$$V^\pi(s_t) := \mathbb{E}_{a_t\sim\pi}[Q(s_t, a_t) - \alpha \log \pi_\theta(a_t|s_t)] \tag{3}$$

The soft Q-function parameters Ψ can be trained to minimize the soft Bellman residual:

$$J_Q = \mathbb{E}_{(s_t,a_t)\sim D_{RB}}[\frac{1}{2}(Q_\Psi(s_t, a_t) - (r(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1}}[V_{\hat{\Psi}}(s_{t+1})]))^2] \tag{4}$$

Where the training data is sampled using a replay buffer D_{RB} containing state-action pairs of previous policy rollouts. To improve training stability it is common to use a target Q-function with parameters $\hat{\Psi}$ that slowly tracks the actual Q-function. Lastly, the policy is optimized by minimizing the following objective:

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

2.2 Imitation Learning

In the imitation learning setup, the agent does not have access to the true environment reward function $r(s_t, a_t)$ 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** (LfD) setup the expert demonstrations $\mathcal{D}_E^{LfD} := \{s_t^k, a_t^k, s_{t+1}^k\}_{k=1}^N$ are given by action-state-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 was done by Ghasemipour et al. (2019). Often the backward KLD is used to measure this discrepancy (Fu et al., 2017; Kostrikov et al., 2020):

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

Learning from observation (LfO) considers a more challenging task where expert actions are not available. Hence, the demonstrations $\mathcal{D}_E^{LfO} := \{s_t^k, s_{t+1}^k\}_{k=1}^N$ 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 imitation learning (Torabi et al., 2018b; Zhu et al., 2020):

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

2.3 Normalizing Flows

Normalizing flows provide a general method to define expressive probability distributions. The change of variable formula is used to describe more complex distributions using simple ones (Rezende & Mohamed, 2015). For a detailed overview, we refer to the work by Papamakarios et al. (2019). A sample from a base distribution $u \sim p(u)$ is transformed to the target distribution x by a mapping $x = M(u; \omega)$. The transformation M with parameter ω has to be *differentiable* and *invertible* and is typically implemented using neural networks. Under these conditions the density of x is well defined and the change of variable formula describes the probability of the resulting density $p_\omega(x)$ by

$$p_\omega(x) = p(u)|\det \text{Jac}_M(u; \omega)|$$

where $\det$ is the determinant. The base distribution $p(u)$ in our work is a standard Gaussian distribution. $\text{Jac}_M(u; \omega)$ is the Jacobian of the transformation M. Several of such transformations M can be composed to construct a complex transformation from multiple simple transformations. A flow is defined by several changes of the densities by transformations M_k. The final sample of a flow is: $x_K = M_K \circ M_{K-1} \cdots \circ M_1(u)$ with its corresponding log probability:

$$\log p_\omega(x_K) = \log p(u) - \sum_{k=1}^{K} \log |\det \text{Jac}_{M_k}(u_k; \omega)|.$$

In this work, we used the setup described by Dinh et al. (2017) called RealNVP and expanded it to the conditional case as done by Ardizzone et al. (2019b). We can then write the conditional density using normalizing flows as follows:

$$p_\omega(s_{t+1}|s_t) = p_u(u)|\det \text{Jac}_M(u; s_t, \omega)|$$

Note that $M(u; s_t, \omega)$ remains invertible in u for fixed s_t while its transformation parameters are now conditioned on the state s_t.

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:

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

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:

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

The conditional expert state transition distribution $\mu^E(s_{i+1}|s_i)$ can be learned offline from the demonstrations for example by training a conditional normalizing flow on the given state/next-state pairs. The policy induced conditional state transition distribution can be rewritten with the Bayes theorem using the environment model $p(s_{i+1}|a_i, s_i)$ and the inverse action distribution density $\pi'_\theta(a_i|s_{i+1}, s_i)$:

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

which 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 can be rewritten as

$$\begin{aligned}\min \mathbb{D}_{KL}(\mu^{\pi_\theta}||\mu^E) = \min \sum_{i=0..T-1} \mathbb{E}_{(s_i,a_i,s_{i+1})\sim\pi_\theta}[\log p(s_{i+1}|a_i, s_i) + \log \pi_\theta(a_i|s_i) \\ - \log \pi'_\theta(a_i|s_{i+1}, s_i) - \log \mu^E(s_{i+1}|s_i)]\end{aligned} \tag{10}$$

Now, by defining a reward function (also see A.2)

$$r(a_i, s_i) := \mathbb{E}_{s_{i+1}\sim p(s_{i+1}|a_i,s_i)}[-\log p(s_{i+1}|a_i, s_i) + \log \pi'_\theta(a_i|s_{i+1}, s_i) + \log \mu^E(s_{i+1}|s_i)] \tag{11}$$

that 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 $\pi'_\theta(a_i|s_{i+1}, s_i)$ the state-only trajectory distribution matching problem can be transformed to a max-entropy reinforcement learning task:

$$\begin{aligned}\min \mathbb{D}_{KL}(\mu^{\pi_\theta}||\mu^E) &= \max \sum_{i=0..T-1} \mathbb{E}_{(a_i,s_i)\sim\pi_\theta}[-\log \pi_\theta(a_i|s_i) + r(a_i, s_i)] \\ &= \max \sum_{i=0..T-1} \mathbb{E}_{(a_i,s_i)\sim\pi_\theta}[r(a_i, s_i) + \mathcal{H}(\pi_\theta(\cdot|s)]\end{aligned} \tag{12}$$

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 reinforcement learning algorithms. In this work, we applied the Soft Actor Critic algorithm (Haarnoja et al., 2018a) 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. (2018b) with our reward definition and one obtains the following infinite horizon maximum entropy objective:

$$J_{ME-iH} = \sum_{i=0..\inf} \mathbb{E}_{(a_i,s_i)\sim\pi_\theta}[\sum_{j=i..\inf} \gamma^{j-i}\mathbb{E}_{(a_j,s_j)\sim\pi_\theta}[r(a_j, s_j) + \mathcal{H}(\pi_\theta(\cdot|s_j)|s_i, a_i]] \tag{13}$$

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 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 and 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 $\mu^E(s_{t+1}|s_t)$ is trained offline using the expert dataset D_E which contains K expert state trajectories. After this initial step the following process is repeated until convergence in each episode. The policy interacts with the environment for T steps to collect state-action-next-state information which 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, which we found works well in practice. Afterwards we use the learned models together with the samples from the replay buffer to compute a one-sample monte carlo approximation of the reward to train the Q-function. The policy $\pi_\theta(a_t|s_t)$ is updated using SAC.

[2]Code will be available at https://github.com/*

Algorithm 1 State-Only Imitation Learning by Trajectory Distribution Matching

procedure SOIL-TDM(D_E)
  Train $\mu^E(s_{t+1}|s_t)$ `with` $D_E : \{s_0, s_1, ...s_T\}_{k=0}^K$
  for `episodes` **do**
    for `range(T)` **do** ▷ generate data
      $\hat{a}_t \leftarrow$ sample($\pi_\theta(\hat{a}_t|s_t)$)
      $s_{t+1} \leftarrow p_{sim}(s_{t+1}|s_t, \hat{a}_t)$ ▷ apply action
      `Store` $(s_t, \hat{a}_t, s_{t+1})$ in D_{RB}
    end for
    for `range(N)` **do** ▷ update dynamics models
      $\{(s_t, \hat{a}_t, s_{t+1})\}_{i=1}^B \sim D_{RB}$ ▷ sample batch
      `train` $\mu_\eta(\hat{a}_t|s_{t+1}, s_t)$ and $\mu_\phi(s_{t+1}|\hat{a}_t, s_t)$
    end for
    for `range(N)` **do** ▷ SAC Optimization
      $\{(s_t, \hat{a}_t, s_{t+1})\}_{i=1}^B \sim D_{RB}$ ▷ sample batch
      $a_t \leftarrow$ sample($\pi_\theta(a_t|s_t)$)
      `optimize` $\pi_\theta(a_t|s_t)$ with J_π from eq. 5 ▷ update policy
      ▷ estimate reward
      $r(s_t, \hat{a}_t) \leftarrow -\log \mu_\phi(s_{t+1}|\hat{a}_t, s_t) + \log \mu_\eta(\hat{a}_t|s_{t+1}, s_t) + \log \mu^E(s_{t+1}|s_t)$
      `optimize` $Q_\psi(\hat{a}_t, s_t)$ with J_Q from eq. 4 ▷ update Q-function
    end for
  end for
end procedure

3.2 Relation to learning from observations

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

$$J_{LfO} = \mathbb{D}_{KL}(\mu^{\pi_\theta}(s', s)||\mu^E(s', s)) \tag{14}$$

which can be rewritten as (see A.1)

$$J_{LfO} = \mathbb{D}_{KL}(\mu^{\pi_\theta}(s_T, ..., s_0)||\mu^E(s_T, ..., s_0)) + \sum_{i=1..T-1} \mathbb{D}_{KL}(\mu^{\pi_\theta}(s_i)||\mu^E(s_i)) \tag{15}$$

Thus, this LfO objective minimizes the sum of the KLD between the joint distributions and the KLDs of the marginal distributions. The SOIL-TDM objective in comparison minimizes purely the KLD of the joint distributions. In case of a perfect distribution matching - a zero KLD between the joint distributions - the KLDs of the marginals also vanish so both objectives have the same optimum.

4 Related Work

A fast and straightforward method to train an agent on expert demonstrations is behavioral cloning (BC) (Pomerleau, 1991). For a given set of expert states, the error on predicted actions is minimized. The performance relies heavily on the number of expert demonstrations. In BC, the actor is only trained on states visited by the expert and therefore does not learn how to combat deviations from small errors or handle states not available in the expert dataset. The work by Ross et al. (2011) improves the limitations of BC using a no-regret imitation learning approach called DAgger.

Many recent approaches are based on inverse reinforcement learning (IRL) (Ng & Russell, 2000). In IRL, the goal is to learn a reward signal for which the expert policy is optimal. AIL algorithms are popular methods to perform imitation learning in a reinforcement learning setup (Ho & Ermon, 2016; Fu et al., 2017; Kostrikov et al., 2019; 2020). 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.

A key problem with AIL for LfD and LfO is optimization instability (Miyato et al., 2018). Wang et al. (2019) avoid the instabilities resulting from adversarial optimization by estimating the support of the expert policy to compute a fixed reward function. Similarly, Brantley et al. (2020) use a fixed reward function by estimating the variance of an ensemble of policies. Both methods relied on additional behavioral cloning steps to reach expert-level performance.

LfO can be divided into model-free and model-based approaches. Torabi et al. (2018b) proposed the model-free approach GAILfO which uses the GAIL principle with the discriminator input being state-only. Yang et al. (2019) 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 IDDM is based on an upper bound of this gap in a model-free way. OPOLO proposed by Zhu et al. (2020) 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 (Sun et al., 2019; Edwards et al., 2018) or inverse action models (Torabi et al., 2018a; Liu et al., 2020; Jiang et al., 2020). Sun et al. (2019) proposed a solution based on forward dynamics models to learn time dependant policies. While being provably efficient, it is not suited for infinite horizon tasks. Alternatively, behavior cloning from observations (BCO) (Torabi et al., 2018a) learns an inverse action model based on simulator interactions to infer actions based on the expert state demonstrations. Jiang et al. (2020) investigated imitation learning using few expert demonstrations and a simulator with misspecified dynamics. Similar to these model-based approaches, GPRIL by Schroecker et al. (2019) uses normalizing flows as generative models to learn backward dynamics models to estimate predecessor transitions and augmenting the expert data set with further trajectories, which lead to expert states. A detailed overview of LfO was done by Torabi et al. (2019).

Our proposed method is most similar to the approach by Kim et al. (2021) called Neural Density Imitation (NDI), where density models are used to perform distribution matching. However, in contrast to NDI which is an LfD approach, our state-only approach does not require expert actions. To remove the requirement for expert actions we use three conditional normalizing flows to estimate the expert state conditional likelihood, the forward dynamics, and the backward dynamics. These are combined into an estimated reward which is used to train the max-entropy RL SAC algorithm. We show that the overall algorithm minimizes the KLD between the imitation policy and expert policy joint state distributions.

5 EXPERIMENTS

We evaluate our proposed method described in Section 3 in a variety of different imitation learning tasks and compare it against the baseline method OPOLO. The performance compared to the recent state-only imitation learning approach OPOLO is of particular interest since both methods rely only on state observation data to imitate an expert. For simple reinforcement learning tasks, most current imitation learning methods are able to reach expert level performance using a single trajectory of expert demonstrations. We therefore evaluate and compare all methods in more complex and higher dimensional continuous control environments using the Pybullet[3] physics simulation (Coumans & Bai, 2016–2019). To evaluate the performance of all methods, the episode reward of the trained policies are compared to reward from the expert policy.

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 OPOLO and SOIL-TDM - which employ a conditional Gaussian policy - more challenging and more similar to real-world IL settings. Afterward, the expert trajectories are generated using the trained policy and saved as done by Ho & Ermon (2016)[4]. For the OPOLO baseline, the original implementation with the official default parameters for each environment are used[5].

[3]https://github.com/bulletphysics/bullet3/tree/master/examples/pybullet/gym/pybullet_envs/examples (Open Source MIT License)

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

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

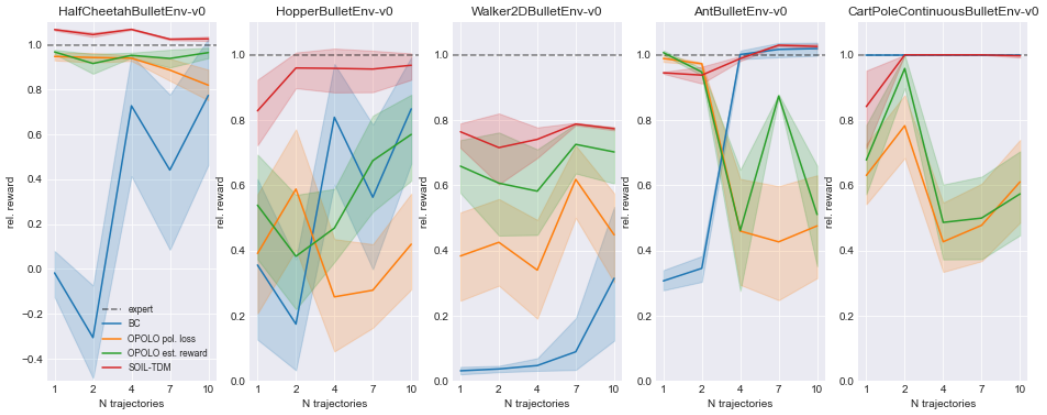

Figure 2: Unkown true environment reward selection criteria: Performance comparison on continuous control environments using the Pybullet physics simulation. Relative reward for different amount of expert trajectories using OPOLO, BC and SOIL-TDM as imitation learning methods. The best policies based on estimated convergence values were selected. The value 1 corresponds to expert policy performance.

We want to answer the question which method is best suited to train a policy if no environment reward for early stopping and only a limited amount of expert demonstrations are available. The experiments in OPOLO (Zhu et al., 2020) used the environment reward to select the best performing policy after training was done. This was a fair comparison to find the method resulting in the overall best performance, since the authors used the same approach for all methods they compared. We argue, however, that such a policy selection criterion might not be available in many real-world use cases.

We use convergence estimates available during training which do not rely on the environment reward to select the best policy for each method. In adversarial training the duality gap (Grnarova et al., 2019; Sidheekh et al., 2021) 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 reward over ten consecutive epochs. To estimate the convergence of SOIL-TDM the policy loss based on the KLD from equation 10 is used. It can be estimated using the same models used for training the policy.

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. We plot the rewards normalized so that 1 corresponds to expert performance. Implementation details of our method are described in the Appendix A.3.

The evaluation results of the discussed methods on a suite of continuous control tasks with unkown true environment reward as a selection criterion are shown in Figure 2. The achieved rewards are plotted with respect to the number of expert trajectories provided for training the agent. The confidence intervals are plotted using lighter colors.

If the true environment reward is unkown the results show that SOIL-TDM achieves or surpasses the performance of OPOLO for both OPOLO policy selection methods over all numbers of expert demonstrations on all tested environments (with the exception of one and two expert trajectories in the Ant environment). In general the adversarial method OPOLO exhibits a high variance of the achieved rewards. The stability of the SOIL-TDM training method without an adversarial loss is evident from the small confidence band of the results which gets smaller for more expert demonstrations.

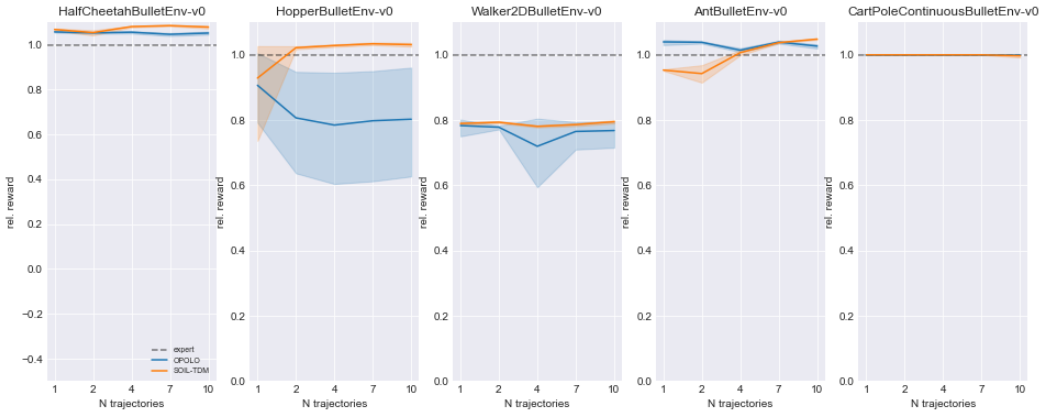

Figure 3: Best true environment reward selection criterion: Performance comparison on continuous control environments using the Pybullet physics simulation. Relative reward for different amount of expert trajectories using OPOLO and SOIL-TDM as imitation learning methods. The value 1 corresponds to expert policy performance.

Figure 3 shows the benchmark results of OPOLO and SOIL-TDM if the true environment reward is used as an early stopping criterion. This is the method applied by the OPOLO authors. In this setup, our method still achieves competitive performance or surpasses OPOLO with the exception of one and two expert trajectories for the Ant environment.

6 Conclusion

In this work we propose a non-adversarial state-only imitation learning approach based on the minimization of the Kulback-Leibler divergence between the policy and the expert 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 modeled using conditional normalizing flows. The proposed approach is compared to the state-of-the-art learning from observations method OPOLO in a scenario with unkown environment rewards and achieves superior performance. The non-adversarial training objective leads to stable and reproducible results compared to the adversarial method across all tasks.

Reproducibility and Ethics Statement

To ensure reproducibility the used hyperparmeters and neural network topologies of our training setup are described in detail in Appendix A.3. Additionally, the code will be made publicly available with a camera ready submission.

Societal impacts of improved imitation learning algorithms can be increased automation with adverse effects on the job market for human labor in areas such as transportation or industrial assembly tasks. Another problem might be the potential misuse of such algorithms for non-civil purposes. Real-world application of imitation learning algorithms might also pose a safety risk and should therefore be regulated, especially in domains where human life might be harmed such as traffic.

References

Lynton Ardizzone, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. In *International Conference on Learning Representations, ICLR*, 2019a. URL `https://github.com/VLL-HD/FrEIA`.

Lynton Ardizzone, Carsten Lüth, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Guided image generation with conditional invertible neural networks. *ArXiv*, 2019b. URL `http://arxiv.org/abs/1907.02392`.

Kiante Brantley, Wen Sun, and Mikael Henaff. Disagreement-regularized imitation learning. In *International Conference on Learning Representations, ICLR*, 2020.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp, 2017. URL `https://arxiv.org/abs/1605.08803`.

Ashley D. Edwards, Himanshu Sahni, Yannick Schroecker, and Charles L. Isbell. Imitating latent policies from observation. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *International Conference on Learning Representations, ICLR*, 2017.

Seyed Kamyar Seyed Ghasemipour, Shane Gu, and Richard S. Zemel. Understanding the relation between maximum-entropy inverse reinforcement learning and behaviour cloning. In *DGS@International Conference on Learning Representations*, 2019.

Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. In *arxiv:cs.LG*, 2017.

Paulina Grnarova, Kfir Y. Levy, Aurelien Lucchi, Nathanael Perraudin, Ian Goodfellow, Thomas Hofmann, and Andreas Krause. A domain agnostic measure for monitoring and evaluating gans. In *Advances in Neural Information Processing Systems*, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. Technical report, 2018b.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2016.

Shengyi Jiang, Jingcheng Pang, and Yang Yu. Offline imitation learning with a misspecified simulator. In *Advances in Neural Information Processing Systems*, 2020.

Chi Jin, Praneeth Netrapalli, and Michael Jordan. What is local optimality in nonconvex-nonconcave minimax optimization? In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Kuno Kim, Akshat Jindal, Yang Song, Jiaming Song, Yanan Sui, and Stefano Ermon. Imitation with neural density models. In *arxiv:cs.LG*, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations, ICLR*, 2015.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, 2018.

Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International Conference on Learning Representations, ICLR*, 2019.

Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. In *International Conference on Learning Representations, ICLR*, 2020.

Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 204–211, 2017.

Fangchen Liu, Zhan Ling, Tongzhou Mu, and Hao Su. State alignment-based imitation learning. In *International Conference on Learning Representations, ICLR*, 2020.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations, ICLR*, 2018.

A. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *"International Conference on Machine Learning"*, 2000.

Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. In *Foundations and Trends in Robotics*, 2018.

George Papamakarios, Eric T. Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2019.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Dean A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. In *Neural Computation*, 1991.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.

Yannick Schroecker, Mel Vecerík, and Jonathan Scholz. Generative predecessor models for sample-efficient imitation learning. In *International Conference on Learning Representations, ICLR*, 2019.

Sahil Sidheekh, Aroof Aimen, Vineet Madan, and N. C. Krishnan. On duality gap as a measure for monitoring gan training. *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021.

Wen Sun, Anirudh Vemula, Byron Boots, and Drew Bagnell. Provably efficient imitation learning from observation alone. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 4950–4957, 2018a.

Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. In *International Conference on Machine Learning Workshop on Imitation, Intent, and Interaction (I3)*, 2018b.

Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.

Ruohan Wang, Carlo Ciliberto, Pierluigi Vito Amadori, and Yiannis Demiris. Random expert distillation: Imitation learning via expert policy support estimation. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Chao Yang, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Huaping Liu, Junzhou Huang, and Chuang Gan. Imitation learning from observations by minimizing inverse dynamics disagreement. In *Advances in Neural Information Processing Systems*, 2019.

Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. Off-policy imitation learning from observations. In *Advances in Neural Information Processing Systems*, 2020.

Brian D. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, 2010.

A Appendix

A.1 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:

$$\min J_{LfO} = \min \mathbb{D}_{KL}(\mu^{\pi_\theta}(s', s)||\mu^E(s', s)) \tag{16}$$

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

$$\begin{aligned}
J_{LfO} &= \sum_{i=0..T-1} \mathbb{D}_{KL}(\mu^{\pi_\theta}(s_{i+1}, s_i)||\mu^E(s_{i+1}, s_i)) \\
&= \sum_{i=0..T-1} \int \mu^{\pi_\theta}(s_{i+1}, s_i)(\log \mu^{\pi_\theta}(s_{i+1}, s_i) - \log \mu^E(s_{i+1}, s_i)) \\
&= \sum_{i=0..T-1} \int \mu^{\pi_\theta}(s_T, .., s_0)(\log \mu^{\pi_\theta}(s_{i+1}, s_i) - \log \mu^E(s_{i+1}, s_i)) \\
&= \int \mu^{\pi_\theta}(s_T, .., s_0) \sum_{i=0..T-1} (\log \mu^{\pi_\theta}(s_{i+1}, s_i) - \log \mu^E(s_{i+1}, s_i)) \\
&= \int \mu^{\pi_\theta}(s_T, .., s_0) \sum_{i=0..T-1} (\log \mu^{\pi_\theta}(s_{i+1}|s_i) + \log \mu^{\pi_\theta}(s_i) - \log \mu^E(s_{i+1}|s_i)) - \log \mu^E(s_i) \\
&= \mathbb{E}_{(s_T,...,s_0)\sim\mu^{\pi_\theta}}[\log \frac{\mu^{\pi_\theta}(s_T, ..., s_0)}{\mu^E(s_T, ..., s_0)} + \log \prod_{i=1..T-1} \frac{\mu^{\pi_\theta}(s_i)}{\mu^E(s_i)}] \\
&= \mathbb{D}_{KL}(\mu^{\pi_\theta}(s_T, ..., s_0)||\mu^E(s_T, ..., s_0)) + \sum_{i=1..T-1} \mathbb{E}_{(s_T,...,s_0)\sim\mu^{\pi_\theta}}[\log \frac{\mu^{\pi_\theta}(s_i)}{\mu^E(s_i)}] \\
&= \mathbb{D}_{KL}(\mu^{\pi_\theta}(s_T, ..., s_0)||\mu^E(s_T, ..., s_0)) + \sum_{i=1..T-1} \mathbb{D}_{KL}(\mu^{\pi_\theta}(s_i)||\mu^E(s_i))
\end{aligned} \tag{17}$$

A.2 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

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

are bounded which holds if

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

for some ϵ and H which is a rather strong assumption which requires compact action and state spaces and a non-zero probability to reach 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 the clipping the logarithms to a maximum negative value still provides a reward signal which guides the imitation learning to policies which achieve higher rewards.

A.3 IMPLEMENTATION DETAILS

We use the same policy implementation for all our SOIL-TDM experiments. The stochastic policies $\pi_\theta(a_t|s_t)$ 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. (2018b) and the used hyperpameter 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.

Table 1: Training Hyperparameter

SAC Parameter	Value
Optimizer	Adam (Kingma & Ba, 2015)
learning rate policy	$1 \cdot 10^{-4}$
learning rate Q-function	$3 \cdot 10^{-4}$
discount γ	0.7 (Ant 0.9)
mini batch size	2048
replay buffer size	$2 \cdot 10^6$
target update interval	1
number of environments	16
max number of environment steps	$1.6 \cdot 10^6$ (Ant $8 \cdot 10^6$)
SOIL-TDM Parameter	**Value**
expert transition model training steps	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

We implement all our models for SOIL-TDM using the PyTorch framework version 1.9.0 (Paszke et al., 2017). To estimate the imitation reward in SOIL-TDM a model for the expert transitions $\mu_E(s'|s)$ as well as a forward $\mu(s'|s,a)$ and backward dynamics model $\mu(a|s',s)$ has to be learned. All three density models are based on RealNVPs (Dinh et al., 2017) 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 Kingma & Dhariwal (2018). To implement these models, we use the publicly available VLL-FrEIA[6] framework version 0.2 (Ardizzone et al., 2019a) 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 low amount 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 compute time for the SOIL-TDM method depends on the time to convergence and is from 4h to 16h.

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

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
Conditional hidden neurons	64	48	192
Conditional hidden layer	2	2	2
Conditional feature size	32	32	8
Flow block hidden neurons	64	48	192
Flow block hidden layer	2	2	2
Exponent clamping	8	1	1

A.4 Additional Results

The following figures (Figure 4 - 7) show the policy loss and the estimated reward together with the environment reward during the training on different pybullet environments for OPOLO 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 with the exception of the walker environment. It is possible that the policy loss of SOIL-TDM is lower than 0 since its based not on the true distributions. Instead its 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 (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.

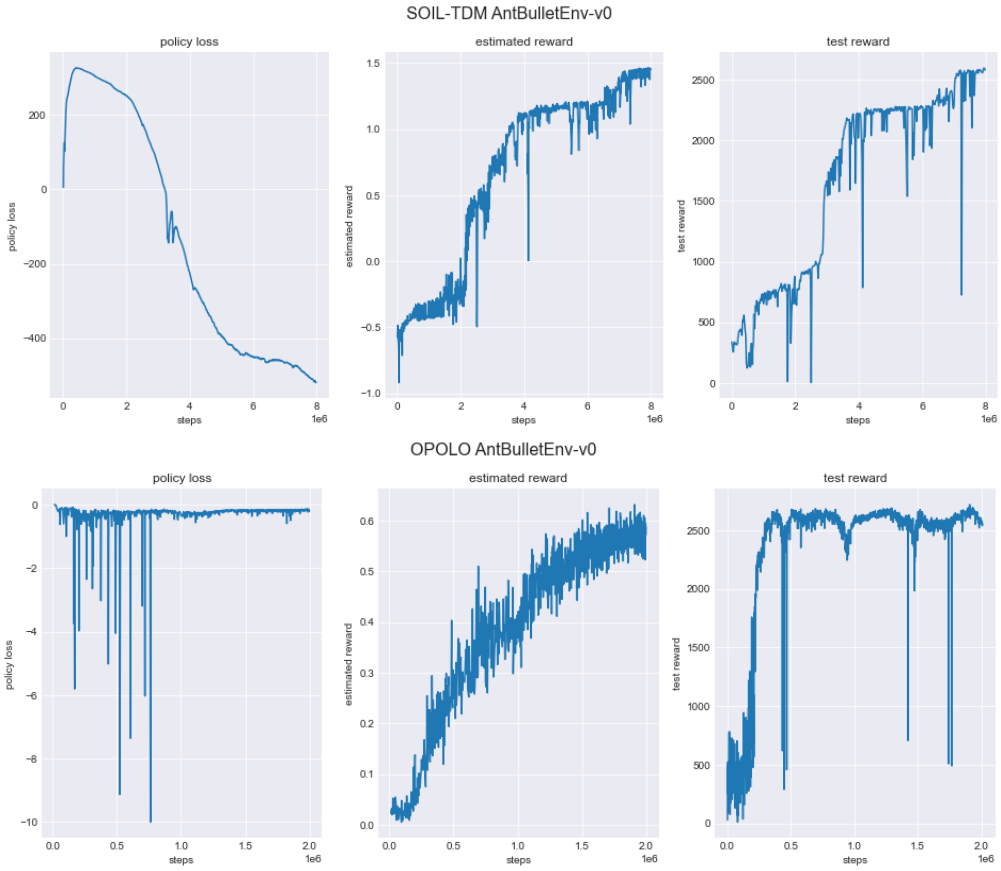

Figure 4: The policy loss, estimated reward (based on discriminator output for OPOLO) and the environment test loss during training in the pybullet Ant environment using our proposed SOIL-TDM and the original OPOLO implementation with 4 expert trajectories.

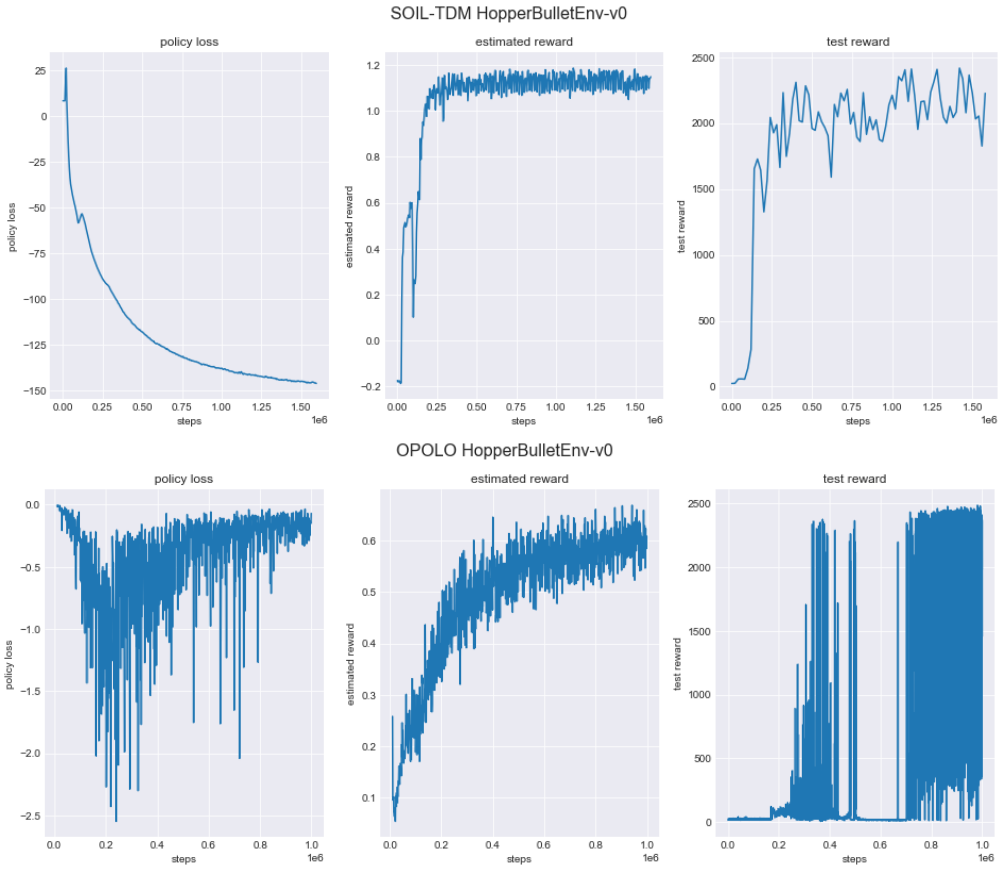

Figure 5: The policy loss, estimated reward (based on discriminator output for OPOLO) and the environment test reward during training in the pybullet Hopper environment using our proposed SOIL-TDM and the original OPOLO implementation with 4 expert trajectories.

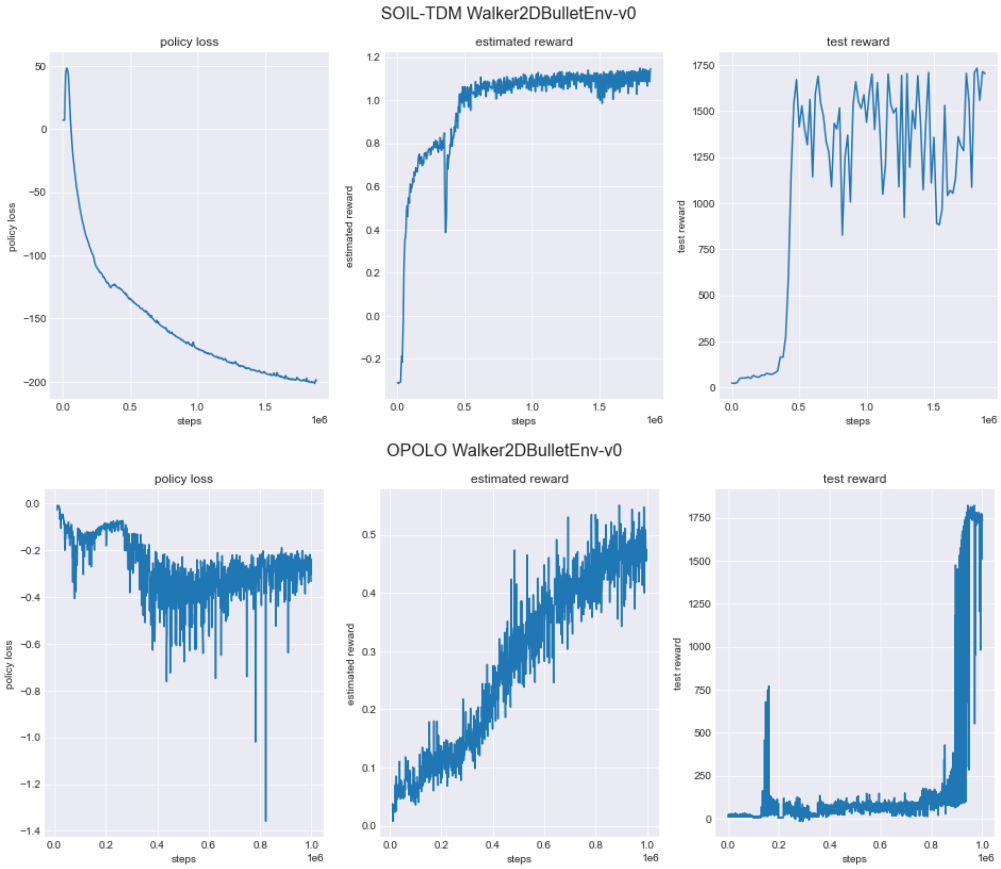

Figure 6: The policy loss, estimated reward (based on discriminator output for OPOLO) and the environment test reward during training in the pybullet Walker2D environment using our proposed SOIL-TDM and the original OPOLO implementation with 4 expert trajectories.

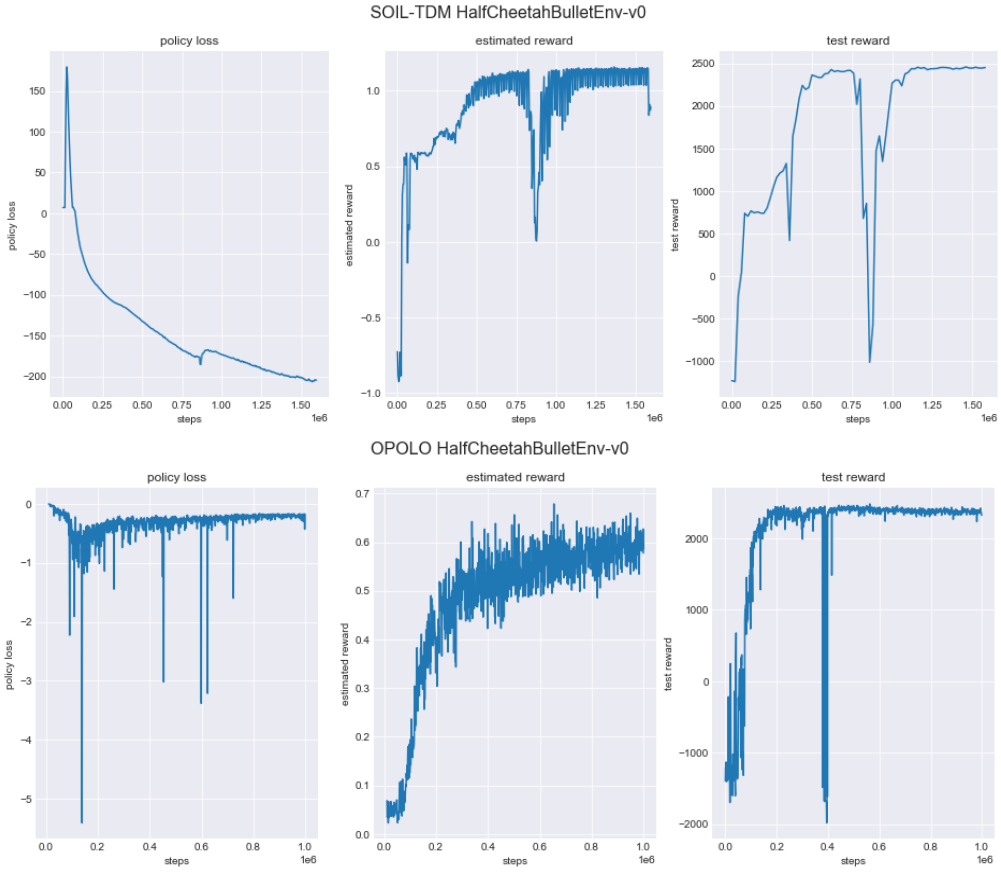

Figure 7: The policy loss, estimated reward (based on discriminator output for OPOLO) and the environment test reward during training in the pybullet HalfCheetah environment using using our proposed SOIL-TDM and the original OPOLO implementation with 4 expert trajectories.