PRIMAL WASSERSTEIN IMITATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Imitation Learning (IL) methods seek to match the behavior of an agent with that of an expert. In the present work, we propose a new IL method based on a conceptually simple algorithm: Primal Wasserstein Imitation Learning (PWIL), which ties to the primal form of the Wasserstein distance between the expert and the agent state-action distributions. We present a reward function which is derived offline, as opposed to recent adversarial IL algorithms that learn a reward function through interactions with the environment, and which requires little fine-tuning. We show that we can recover expert behavior on a variety of continuous control tasks (beyond locomotion) of the MuJoCo domain in a sample efficient manner in terms of agent interactions and of expert interactions with the environment. Finally, we show that the behavior of the agent we train matches the behavior of the expert with the Wasserstein distance, rather than the commonly used proxy of performance.

1 INTRODUCTION

Reinforcement Learning (RL) has solved a number of difficult tasks whether in games (Tesauro, 1995; Mnih et al., 2015; Silver et al., 2016) or robotics (Abbeel & Ng, 2004; Andrychowicz et al., 2020). However, RL relies on the existence of a reward function, that can be either hard to specify or too sparse to be used in practice. Imitation Learning (IL) is a paradigm that applies to these environments with *hard to specify rewards*: we seek to solve a task by learning a policy from a fixed number of demonstrations generated by an expert. IL methods can typically be folded into two paradigms: Behavioral Cloning, or BC (Pomerleau, 1991; Bagnell et al., 2007; Ross & Bagnell, 2010) and Inverse Reinforcement Learning, or IRL (Russell, 1998; Ng et al., 2000). In BC, we seek to recover the expert's behavior by directly learning a policy that *matches* the expert behavior in some sense. In IRL, we assume that the demonstrations come from an agent that acts optimally with respect to an unknown reward function that we seek to recover, to subsequently train an agent on it. Although IRL methods introduce an intermediary problem to solve (i.e. recovering the environment's reward) they offer the benefit of being less sensitive to distributional shift (Pomerleau, 1991), generalizing to environments with possibly changing dynamics (Piot et al., 2013), or recovering a near-optimal agent from suboptimal demonstrations (Brown et al., 2019; Jacq et al., 2019).

However, IRL methods are usually based on an iterative process alternating between reward estimation and RL, which might result in poor sample-efficiency. Earlier IRL methods (Ng et al., 2000; Abbeel & Ng, 2004; Ziebart et al., 2008) require multiple calls to a Markov decision process solver (Puterman, 2014), whereas recent adversarial IL approaches (Finn et al., 2016; Ho & Ermon, 2016; Fu et al., 2018) interleave the learning of the reward function with the learning process of the agent. Adversarial IL methods are based on an adversarial training paradigm similar to Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), where the learned reward function can be thought of as the confusion of a discriminator that learns to differentiate expert transitions from non expert ones. These methods are well suited to the IL problem since they implicitly minimize an f-divergence between the state-action distribution of an expert and the state-action distribution of the learning agent (Ghasemipour et al., 2019; Ke et al., 2019). However the interaction between a generator (the policy) and the discriminator (the reward function) makes it a minmax optimization problem, and therefore comes with practical challenges that might include training instability, sensitivity to hyperparameters and poor sample efficiency.

In this work, we use the Wasserstein distance as a measure between the state-action distributions of the expert and of the agent. Contrary to f-divergences, the Wasserstein distance is a true distance, it is smooth and it is based on the geometry of the metric space it operates on. The Wasserstein distance

has gained popularity in GAN approaches (Arjovsky et al., 2017) through its dual formulation which comes with challenges (see Section 5). Our approach is novel in the fact that we consider the problem of minimizing the Wasserstein distance through its primal formulation. Crucially, the primal formulation prevents the minmax optimization problem, and requires little fine tuning.

We introduce a reward function computed offline based on an upper bound of the primal form of the Wasserstein distance. The inferred reward function is non-stationary, like adversarial IL methods, but it is not re-evaluated as the agent interacts with the environment, therefore the reward function we define is computed *offline*. We present a true distance to compare the behavior of the expert and the behavior of the agent, rather than using the common proxy of performance with respect to the true return of the task we consider (as it is unknown in general). Our method recovers expert behaviour better than existing state-of-the-art methods while being based on significantly fewer hyperparameters; it operates even in the extreme low data regime of demonstrations, and is the first method that makes *Humanoid* run with a single (subsampled) demonstration.

2 Background and Notations

Markov decision processes We describe environments as episodic Markov Decision Processes (MDP) with finite time horizon (Sutton & Barto, 2018) $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \delta, \rho_0, T)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition kernel, r is the reward function, δ is the discount factor, ρ_0 is the initial state distribution and T is the time horizon. We will denote the dimensionality of $\mathcal{S}$ and $\mathcal{A}$ as $|\mathcal{S}|$ and $|\mathcal{A}|$ respectively. A policy π is a mapping from states to distributions over actions; we denote the space of all policies by Π. In RL, the goal is to learn a policy π^* that maximizes the expected sum of discounted rewards it encounters, that is, the expected return. Depending on the context, we might use the concept of a cost c rather than a reward r (Puterman, 2014), which essentially moves the goal of the policy from maximizing to minimizing its return.

State action distributions Suppose a policy π visits the successive states and actions $s_1, a_1, \ldots, s_T, a_T$ during an episode, we define the empirical state-action distribution $\hat{\rho}_\pi$ as: $\hat{\rho}_\pi = \frac{1}{T}\sum_{t=1}^{T} \delta_{s_t,a_t}$, where δ_{s_t,a_t} is a Dirac distribution centered on (s_t, a_t). Similarly, suppose we have a set of expert demonstrations $\mathcal{D} = \{s^e, a^e\}$ of size D, then the associated empirical expert state-action distribution $\hat{\rho}_e$ is defined as: $\hat{\rho}_e = \frac{1}{D}\sum_{(s,a)\in\mathcal{D}} \delta_{s,a}$.

Wasserstein distance Suppose we have the metric space (M, d) where M is a set and d is a metric on M. Suppose we have μ and ν two distributions on M with finite moments, the p-th order Wasserstein distance (Villani, 2008) is defined as $\mathcal{W}_p^p(\mu,\nu) = \inf_{\gamma\in\Gamma(\mu,\nu)} \int_{M\times M} d(x,y)^p d\gamma(x,y)$, where $\Gamma(\mu,\nu)$ is the set of all couplings between μ and ν. In the remainder, we only consider distributions with finite support. A coupling between two distributions of support cardinal T and D is a doubly stochastic matrix of size $T \times D$. We note Γ the set of all doubly stochastic matrices of size $T \times D$:

$$\Gamma = \Big\{\gamma \in \mathbb{R}_+^{T\times D} \mid \forall j \in [1:D], \sum_{i'=1}^{T} \gamma[i',j] = \frac{1}{D}, \forall i \in [1:T], \sum_{j'=1}^{D} \gamma[i,j'] = \frac{1}{T}\Big\}.$$

The Wasserstein distance between distributions of state-action pairs requires the definition of a metric in the space $(\mathcal{S}, \mathcal{A})$. Defining a metric in an MDP is non trivial (Givan et al., 2003; Ferns et al., 2004; Mahadevan & Maggioni, 2007); we show an example where the metric is learned from demonstrations in Section 4.4. For now, we assume the existence of a metric $d : (\mathcal{S}, \mathcal{A}) \times (\mathcal{S}, \mathcal{A}) \mapsto \mathbb{R}^+$.

3 Method

We present the theoretical motivation of our approach: the minimization of the Wasserstein distance between the state-action distributions of the agent and the expert. We introduce a reward based on an upper-bound of the Wasserstein distance inferred from a relaxation of the optimal coupling condition, and present the resulting algorithm: Primal Wasserstein Imitation Learning (PWIL).

3.1 Wasserstein distance minimization

Central to our approach is the minimization of the Wasserstein distance between the state-action distribution of the policy we seek to train $\hat{\rho}_\pi$ and the state-action distribution of the expert $\hat{\rho}_e$. In

other words, we aim at optimizing the following problem:

$$\inf_{\pi \in \Pi} \mathcal{W}_2^2(\hat{\rho}_\pi, \hat{\rho}_e) = \inf_{\pi \in \Pi} \inf_{\gamma \in \Gamma} \sum_{i=1}^{T} \sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^2 \gamma[i,j]. \tag{1}$$

We can interpret the Wasserstein distance using the earth's movers analogy (Villani, 2008). Consider that the state-action pairs of the expert are D *holes* of mass D^{-1} and that the state-action pairs of the policy are *piles of dirt* of mass T^{-1}. A coupling γ is a transport strategy between the piles of dirt and the holes, where $\gamma[i,j]$ stands for how much of the pile of dirt i should be moved towards the hole j. The optimal coupling is the one that minimizes the distance that the earth mover travels to put all piles of dirt to holes. Note that to compute the optimal coupling, we need knowledge of the locations of all piles of dirt. In the context of RL, this means having access to the full trajectory generated by π. From now on, we write γ_π^* as the optimal coupling for the policy π, that we inject in Eq. (1):

$$\gamma_\pi^* = \arg\min_{\gamma \in \Gamma} \sum_{i=1}^{T} \sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^2 \gamma[i,j]$$

$$\inf_{\pi \in \Pi} \mathcal{W}_2^2(\hat{\rho}_\pi, \hat{\rho}_e) = \inf_{\pi \in \Pi} \sum_{i=1}^{T} c_{i,\pi}^* \text{ with } c_{i,\pi}^* = \sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^2 \gamma_\pi^*[i,j]. \tag{2}$$

In Eq. (2), we have introduced $c_{i,\pi}^*$, which we interpret as a cost to minimize using RL. As $c_{i,\pi}^*$ depends on the optimal coupling γ_π^*, we can only define $c_{i,\pi}^*$ at the very end of an episode. This can be problematic if an agent learns in an online manner or in large time-horizon tasks. Thus, we introduce an upper bound to the Wasserstein distance that yields a cost we can compute online, based on a suboptimal coupling strategy.

3.2 Greedy coupling

In this section we introduce the greedy coupling $\gamma_\pi^g \in \Gamma$, defined recursively for $1 \leq i \leq T$ as:

$$\gamma_\pi^g[i,:] = \arg\min_{\gamma[i,:] \in \Gamma_i} \sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^2 \gamma[i,j] \tag{3}$$

$$\text{with } \Gamma_i = \Big\{ \gamma[i,:] \in \mathbb{R}_+^D \;\Big|\; \underbrace{\sum_{j'=1}^{D} \gamma[i,j'] = \frac{1}{T}}_{\text{constraint}(a)}, \underbrace{\forall k \in [1:D], \sum_{i'=1}^{i-1} \gamma_g[i',k] + \gamma[i,k] \leq \frac{1}{D}}_{\text{constraint}(b)} \Big\}.$$

Similarly to Equation (1), Equation (3) can be interpreted using the earth mover's analogy. Contrary to Equation (1) where we assume knowledge of the positions of the T piles of dirts, we now consider that they appear sequentially, and that the earth's mover needs to transport the new pile of dirt to holes *right when it appears*. To do so, we derive the distances to all holes and move the new pile of dirt to the closest remaining available holes, hence the *greedy* nature of it. In Equation (3), the constraint (a) means that all the dirt that appears at the i-th timestep needs to be moved, and the constraint (b) means that we cannot fill the holes more than their capacity $\frac{1}{D}$. We show the difference between the greedy coupling and the optimal coupling in Figure 1, and provide the pseudo-code to derive it in Algorithm 1.

We can now define an upper bound of the Wasserstein distance using the greedy coupling (since by definition it is suboptimal):

$$\inf_{\pi \in \Pi} \mathcal{W}_2^2(\hat{\rho}_\pi, \hat{\rho}_e) = \inf_{\pi \in \Pi} \sum_{i=1}^{T} \sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^2 \gamma_\pi^*[i,j] \leq \inf_{\pi \in \Pi} \sum_{i=1}^{T} \sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^2 \gamma_\pi^g[i,j] \tag{4}$$

In Section 4, we empirically validate this bound. We infer a cost from Equation (1) at each timestep i:

$$c_{i,\pi}^g = \sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^2 \gamma_\pi^g[i,j]. \tag{5}$$

Note that the greedy coupling $\gamma_\pi^g[t,.]$ defined in Equation (3) depends on all the previous state-actions visited by the policy π, which makes the cost $c_{i,\pi}^g$ non-stationary, similarly to adversarial IL methods.

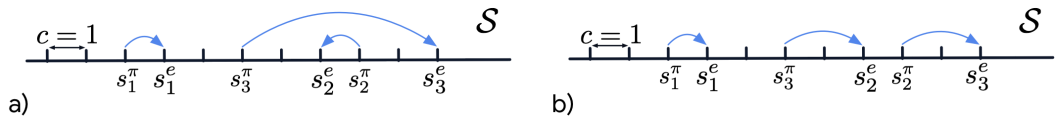

Figure 1: Illustration of the difference between the a) greedy coupling and the b) optimal coupling. We present an MDP where we drop the dependency on the action. The state space is $\mathbb{R}$ and the metric associated is the Euclidean distance. We note the states visited by the policy: $s_1^\pi, s_2^\pi, s_3^\pi$ and the states visited by the expert: s_1^e, s_2^e, s_3^e. When the policy encounters the state s_2^π, and because we do not know s_3^π yet, the greedy coupling strategy consists in coupling it with s_2^e although the optimal coupling strategy would be to couple it with s_3^e. Note that the total cost with the greedy coupling is 7 whereas the total cost with the optimal coupling is 5. This highlights that the optimal coupling needs knowledge about the whole policy's trajectory to be derived.

Although this cost function is non-stationary, it does not have to be re-estimated as the agent interacts with the environment, and hence is said to be derived *offline*. We can infer a reward from the cost, $r_{i,\pi} = f(c^g_{i,\pi})$ where f is a monotonously decreasing function. Crucially, we have defined a reward function that we seek to maximize, without introducing an inner minimization problem (which is the case with adversarial IL approaches).

We can now derive an algorithm building on this reward function (Algorithm 1). The algorithm presented is written using pseudo-code for a generic agent A which implements a policy π_A, it observes tuples (s, a, r, s') and possibly updates its components. The algorithm computes the greedy coupling with a complexity $\mathcal{O}((|\mathcal{S}| + |\mathcal{A}|)D)$. Note that in the case where $T = D$, computing the greedy coupling is simply a lookup of the expert state-action pair that minimizes the distance with the agent state-action pair, followed by a pop-out of this minimizing expert state-action pair from the set of expert demonstrations. We give an example rundown of the algorithm in Section D.

Algorithm 1 PWIL: Primal Wasserstein Imitation Learning

Input: Expert demonstrations $\mathcal{D} = \{s^e_j, a^e_j\}_{j\in[1:D]}$, Agent A, number of episodes N
for $k = 1$ **to** N **do**
  Copy expert demonstrations with weight: $\mathcal{D}' := \{s^e_j, a^e_j, w^e_j\}_{j\in[1:D]}$, with $w^e_j = \frac{1}{D}$
  Reset environment, initial state s
  for $i = 1$ **to** T **do**
    Take action $a := \pi_A(s)$, observe next state s'
    Initialize weight $w^\pi := \frac{1}{T}$, cost $c := 0$
    while $w^\pi > 0$ **do**
      Compute $s^e, a^e, w^e := \arg\min_{(s^e,a^e,w^e)\in\mathcal{D}'} d((s,a),(s^e,a^e))$
      if $w^\pi \geq w^e$ **then**
        $c := c + w^e d((s,a),(s^e,a^e))^2$
        $w^\pi := w^\pi - w^e$
        $\mathcal{D}'$.pop(s^e, a^e, w^e)
      else
        $c := c + w^\pi d((s,a),(s^e,a^e))^2$
        $w^e := w^e - w^\pi$
        $w^\pi := 0$
    $r := f(c)$
    A observes (s, a, r, s'), updates itself

4 Experiments

In this section, we present the implementation of PWIL and perform an empirical evaluation, based on the ACME framework (Hoffman et al., 2020). We test PWIL on the standard imitation learning MuJoCo locomotion tasks and compare it to the state of the art GAIL-like algorithm DAC (Kostrikov et al., 2019) and against the common BC baseline. As DAC is based on TD3 which is a variant of DDPG, we use a DDPG-based agent for fair comparison. We also provide a proof-of-concept experiment on a door opening task where the demonstrations do not include actions and are pixel-

based and therefore the metric of the MDP has to be learned. In this setup, we use SAC as the direct RL method. We answer the following questions: 1) Does PWIL recover expert behavior? 2) How sample efficient is PWIL? 3) Does PWIL actually minimize the Wasserstein distance between the distributions of the agent and the expert? 4) Does PWIL extend to visual based observations where the definition of an MDP metric is not straightforward? We also show the dependence of PWIL on multiple of its components through an ablation study. The complete description of the experiments is in the appendix. We attached the code and expert demonstrations for locomotion experiments, as well as videos of the agents in the supplementary material.

4.1 IMPLEMENTATION

We test PWIL following the same evaluation protocol as GAIL (Ho & Ermon, 2016) and DAC (Kostrikov et al., 2019), on six environments from the OpenAI Gym MuJoCo suite (Todorov et al., 2012; Brockman et al., 2016). For each environment, multiple demonstrations are gathered using a DDPG agent with a distributional critic (Lillicrap et al., 2016; Barth-Maron et al., 2018) trained on the actual reward of these environments (which was chosen as it is a base agent in the ACME framework (Hoffman et al., 2020)). We subsample demonstrations by a factor of 20: we select one out of every 20 transitions with a random offset at the beginning of the episode. The point of subsampling is to simulate a low data regime, so that a pure behavioral cloning approach suffers the behavioral drift and hence does not perform to the level of the expert. We run experiments with multiple number of demonstrations: $\{1, 4, 11\}$ (consistently with the evaluation protocol of DAC).

Central to our method is a metric between state-action pairs. We use the *standardized Euclidean distance* which is the L2 distance, weighted along each dimension by the inverse standard deviation of the expert demonstrations. From the cost c_i defined in Equation (5), we define the following reward:

$$r_i = \alpha \exp(-\frac{\beta T}{\sqrt{|\mathcal{S}| + |\mathcal{A}|}} c_i). \tag{6}$$

For all environments, we use $\alpha = 5$ and $\beta = 5$. The scaling factor $\frac{T}{\sqrt{|\mathcal{S}|+|\mathcal{A}|}}$ acts as a normalizer on the dimensionality of the state and action spaces and on the time horizon of the task. We pick $f : x \mapsto \exp(-x)$ as the monotonously decreasing function to get a reward from a cost because it is smooth and has a bounded range $[0, 1]$.

We test our method against a state-of-the-art imitation learning algorithm: Discriminator Actor Critic (DAC) (Kostrikov et al., 2019) and BC (Pomerleau, 1991). DAC is based on GAIL: it discriminates between expert and non-expert early termination states and it uses an off-policy algorithm TD3 (Fujimoto et al., 2018) which is a variant of DDPG instead of the on-policy algorithm TRPO (Schulman et al., 2015). We used the open-source code provided by the authors. The Humanoid environment is not studied in the original DAC paper. The hyperparameters the authors provided having poor performance, we led an exhaustive search detailed in Section A.2, and present results for the best hyperparameters found.

For our method, we use a DDPG agent (Lillicrap et al., 2016) with a distributional critic (Bellemare et al., 2017; Barth-Maron et al., 2018) (see details in the appendix). We initialize the replay buffer by filling it with expert state-action pairs with maximum reward (i.e. α). We train PWIL and DAC in the limit of 1M environment interactions (2.5M for Humanoid) on 10 seeds. Every 10k environment steps, we perform 10-episode rollouts per seed of the policy without exploration noise and report performance with respect to the environment's original reward in Figure 2.

4.2 RESULTS

In Figure 2, PWIL shows improvement on final performance for Hopper, Ant, HalfCheetah and Humanoid over DAC, similar performance for Walker2d, and is worse on Reacher. On Humanoid, although we ran an exhaustive hyperparameter search (see Section A.2) DAC has poor performance, thus exemplifying the brittleness of GAIL-like approaches. On the contrary, PWIL has near-optimal performance, even with a single demonstration.

In terms of sample efficiency, DAC outperforms PWIL on HalfCheetah, Ant and Reacher and has similar performance for Hopper and Walker2d. Note that ablated versions of PWIL are more sample-efficient than DAC on HalfCheetah and Ant (Section C). This seems rather contradictory with the

claim that GAIL-like methods have poor sample efficiency. However the reward function defined in DAC requires careful tuning (e.g. architecture, optimizer, regularizers, learning rate schedule) which demands experimental cycles and therefore comes with a large number of environment interactions.

Remarkably, PWIL is the first method able to consistently learn policies on Humanoid with an average score over 7000 even with a single demonstration, which means that the agent can actually *run*, (other works consider Humanoid is "solved" for a score of 5000 wich corresponds to *standing*).

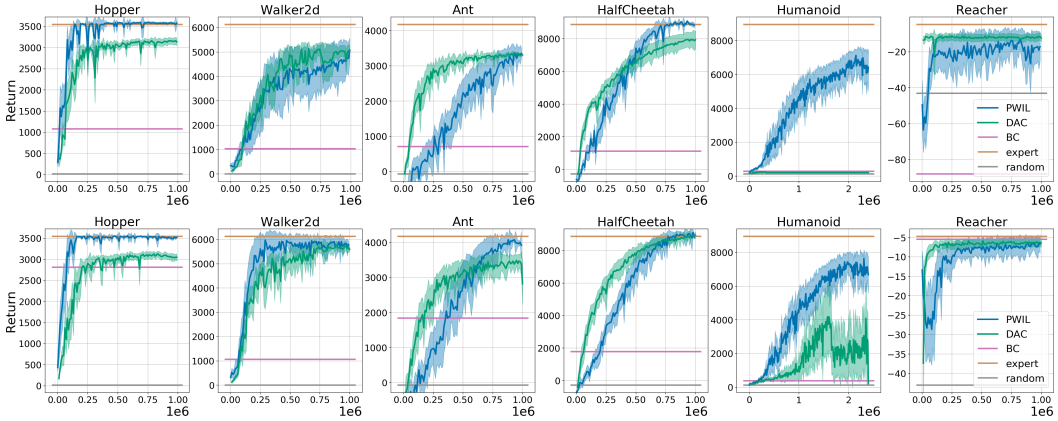

Figure 2: Mean and standard deviation return of the evaluation policy over 10 rollouts and 10 seeds, reported every 10k environment steps. The return is in term of the environment's original reward. Top row: 1 demonstration, bottom row: 11 demonstrations.

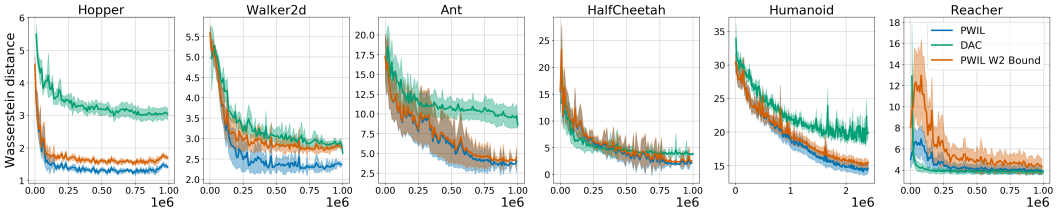

Figure 3: Mean of the Wasserstein distance between the state-action distribution of the evaluation policy and the state-action distribution of the expert over 10 rollouts and 10 seeds. Agents were trained with 11 demonstrations. For PWIL, we include the upper bound on the Wasserstein distance based on the greedy coupling defined in Equation (4).

Wasserstein distance. In Figure 3, we show the Wasserstein distance to expert demonstrations throughout the learning procedure for PWIL and DAC. For both methods, we notice that the distance decreases while learning. PWIL leads to a smaller Wasserstein distance than DAC on all environments but HalfCheetah where it is similar and Reacher where it is larger. This should not come as a surprise since PWIL's objective is to minimize the Wasserstein distance. PWIL defines a reward from an upper bound to the Wasserstein distance between the state-action distributions of the expert and the agent, that we include as well in Figure 3. Notice that our upper bound is "empirically tight", which validates the choice of the greedy coupling as an approximation to the optimal coupling. We emphasize that this distance is useful in real settings of IL, regardless of the method used, where we cannot have access to the actual reward of the task.

4.3 Ablation Study

We run an ablation study on the components of PWIL to highlight the influence of its components. We provide comprehensive results in the appendix and summarize findings here. Pre-filling the replay buffer with expert demonstrations is necessary on Reacher and Walker to get near-expert performance, and otherwise slightly helps. PWIL is sensitive to the quality of the metric as we have that the Euclidean distance (instead of the standardized one) leads to poor performance. PWIL is somewhat close to expert support estimation methods (Brantley et al., 2020; Wang et al., 2019), with

the difference being that in the case of PWIL the expert support "shrinks" as the demonstrations used to compute rewards throughout the episode are removed; we therefore implemented a version of PWIL where the demonstrations are not removed, and found a large drop in performance. Finally, if the distance in the MDP is based on the state rather than the state and action, which is the Learning from Observation (LfO) setting (Torabi et al. (2018); Edwards et al. (2019)), PWIL is still able to recover near-expert performance on all tasks but Walker2d.

4.4 Door Opening Task: Reward From Pixel-Based Observations

In this section, we show that PWIL extends to visual-based setups, where the MDP metric has to be learned. We use the door opening task from Rajeswaran et al. (2018), with *human* demonstrations generated with Haptix (Kumar & Todorov, 2015). The goal of the task is for the controlled hand to open the door (Figure 4) whose location changes at each episode. We add an early termination constraint when the door is opened (hence suppressing the incentive for survival). We assume that we can only access the high-dimensional visual rendering of the demonstrations rather than the internal state (and hence have no access to actions, that is LfO setting).

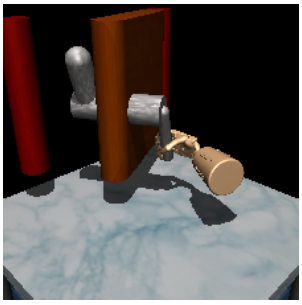

Figure 4: Visual rendering of the door opening task.

The PWIL reward is therefore defined using a distance between the visual rendering of the current state and the visual rendering of the demonstrations. This distance is learned offline through self-supervised learning: we construct an embedding of the frames using Temporal Cycle-Consistency Learning (TCC) (Dwibedi et al., 2019); TCC builds an embedding by aligning frames of videos of a task coming from multiple examples (in this case the demonstrations). The distance follows as the L2 distance between embeddings. The distance is learned on the 25 human demonstrations provided by Rajeswaran et al. (2018) which are then re-used to define the PWIL reward. From the cost c_i defined in Equation (5), we define the following reward: $r_i = \alpha \exp(-\beta T c_i)$, with $\alpha = 5, \beta = 1$. We used SAC (Haarnoja et al., 2018a) as the forward RL algorithm, which was more robust than DDPG on the task. We train the agent on 1.5M environment steps and show that we indeed recover expert behaviour and consistently solve the task in Figure 5.

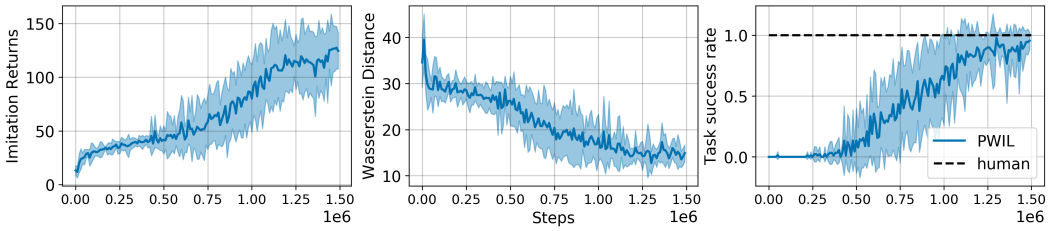

Figure 5: We present the average and standard deviation of evaluation performance of PWIL on the door environment. Metrics are computed every 10k environment steps and aggregated over 10 seeds and 10 rollouts. Left: the sum of imitation reward defined by PWIL. Center: Wasserstein distance in the embedding space to expert demonstrations. Right: Success rate of the agent, *i.e.* ratio of episodes where the agent manages to open the door.

5 Related Work

Adversarial IL Similarly to our method, adversarial IL methods aim at matching the state-action distribution of the agent with the distribution of the expert, using different measures of similarity. For instance, GAIL (Ho & Ermon, 2016) considers the Shannon-Jensen divergence while AIRL (Fu et al., 2018) considers the Kullback-Leibler divergence. In the context of GANs, Arjovsky et al. (2017) show that replacing the f-divergence by the Wasserstein distance through the Kantorovich-Rubinstein duality (Villani, 2008) leads to better training stability, which a number of methods in IL have leveraged (Li et al., 2017; Lacotte et al., 2019; Kostrikov et al., 2019; Xiao et al., 2019; Liu et al., 2020). However, the Kantorovich-Rubinstein duality only holds for the $\mathcal{W}_1$ distance (Peyré et al.,

2019), its implementation comes with a number of practical approximations (*e.g.*, weight clipping to ensure Lipchitz continuity). Although these methods are based on the Wasserstein distance through its dual formulation, they drastically differ from ours as they rely on a minmax optimization problem. DAC (Kostrikov et al., 2019) demonstrates that adversarial IL approaches, while based on GAIL, can be sample efficient. However, they rely on a carefully tuned discriminator (e.g. network architecture, regularization strategy, scheduled learning rates) which requires to interact with the environment. In contrast, the reward function we present relies on only two hyperparameters.

Expert support estimation Another line of research in IL consists in estimating the state-action support of the expert and define a reward that encourages the agent to stay on the support of the expert (Piot et al., 2014; Schroecker & Isbell, 2017; Wang et al., 2019; Brantley et al., 2020; Reddy et al., 2020). Note that the agent might stay on the support of the expert without recovering its state-action distribution. Soft Q Imitation Learning (Reddy et al., 2020) assigns a reward of 1 to all expert transitions and 0 to all transitions the expert encounters; the method learns to recover expert behavior by balancing out the expert transitions with the agent transitions in the replay buffer of a value-based agent. Random Expert Distillation (Wang et al., 2019) estimates the support by using a neural network trained on expert transitions whose target is a fixed random neural network. Disagreement Regularized Imitation Learning (Brantley et al., 2020) estimates the expert support through the variance of an ensemble of BC agents, and use a reward based on the distance to the expert support and the KL divergence with the BC policies. PWIL differs from these methods since it is based on the distance to the support of the expert, with a support that *shrinks* through "pop-outs" to enforce the agent to visit the whole support. We showed in the ablation study that "pop-outs" are *sine qua non* for recovering expert behaviour (see *PWIL-support*).

Trajectory-Based IL Similarly to the door opening experiment, Aytar et al. (2018) learns an embedding offline and reinforces a reward based on the distance in the embedding space to a single demonstration. However, it is well suited for deterministic environments since they use a single trajectory that the agent aims at reproducing while the door experiment shows that PWIL is able to generalize as the door and handle locations are picked at random. Peng et al. (2018) reinforces a temporal reward function (the reward at time t is based on the distance of the state of the agent to the state of the expert at time t). Contrary to PWIL, the imitation reward function is added to a task-dependent reward function; the distance to expert demonstrations is defined along handpicked dimensions of the observation space; the temporal constraint makes it hard to generalize to environment with stochastic initial states.

MMD-IL Generative Moment Matching Imitation Learning (Kim & Park, 2018) assumes a metric on the MDP and aims at minimizing the Maximum Mean Discrepancy (MMD) between the agent and the expert. However, the MMD (like the Wasserstein distance) requires complete rollouts of the agent, and does not present a natural relaxation like the greedy coupling. MMD is based on an on-policy agent (well adapted to rollouts) which makes it significantly less sample-efficient than PWIL. We were not able to implement a version of MMD based on an off-policy agent.

6 Conclusion

In this work, we present Imitation Learning as a distribution matching problem and introduce a reward function which is based on an upper bound of the Wasserstein distance between the state-action distributions of the agent and the expert. A number of IL methods are developed on synthetic tasks, where the evaluation of the IL method can be done with the actual return of the task. We emphasize that in our work, we present a direct measure of similarity between the expert and the agent, that we can thus use in real IL settings, that is in settings where the reward of the task cannot be specified.

The reward function we introduce is learned offline, in the sense that it does not need to be updated with interactions with the environment. It requires little tuning (2 hyperparameters) and it can recover near expert performance with as little as 1 demonstration on all considered environments, even the challenging Humanoid. Finally our method extends to the harder visual-based setting, based on a metric learned offline from expert demonstrations using self-supervised learning.

REFERENCES

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, 2004.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 2020.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, 2017.

Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. In *Advances in Neural Information Processing Systems*, 2018.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

JA Bagnell, Joel Chestnutt, David M Bradley, and Nathan D Ratliff. Boosting structured prediction for imitation learning. In *Advances in Neural Information Processing Systems*, 2007.

Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *International Conference on Learning Representations*, 2018.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, 2017.

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

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning*, 2019.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *International Conference on Learning Representations*, 2016.

Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Ashley Edwards, Himanshu Sahni, Yannick Schroecker, and Charles Isbell. Imitating latent policies from observation. In *International Conference on Machine Learning*, 2019.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *Conference on Uncertainty in Artificial Intelligence*, 2004.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, 2016.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *International Conference on Learning Representations*, 2018.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018.

Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. *Conference on Robot Learning*, 2019.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 2003.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *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, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *AAAI Conference on Artificial Intelligence*, 2018.

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

Matt Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Feryal Behbahani, Tamara Norman, Abbas Abdolmaleki, Albin Cassirer, Fan Yang, Kate Baumli, et al. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020.

Alexis Jacq, Matthieu Geist, Ana Paiva, and Olivier Pietquin. Learning from a learner. In *International Conference on Machine Learning*, 2019.

Liyiming Ke, Matt Barnes, Wen Sun, Gilwoo Lee, Sanjiban Choudhury, and Siddhartha Srinivasa. Imitation learning as f-divergence minimization. *arXiv preprint arXiv:1905.12888*, 2019.

Kee-Eung Kim and Hyun Soo Park. Imitation learning via kernel mean embedding. In *AAAI Conference on Artificial Intelligence*, 2018.

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

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. *International Conference on Learning Representations*, 2019.

V. Kumar and E. Todorov. Mujoco haptix: A virtual reality system for hand manipulation. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015.

Jonathan Lacotte, Mohammad Ghavamzadeh, Yinlam Chow, and Marco Pavone. Risk-sensitive generative adversarial imitation learning. In *International Conference on Artificial Intelligence and Statistics*, 2019.

Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In *Advances in Neural Information Processing Systems*, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *International Conference on Learning Representations*, 2016.

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

Sridhar Mahadevan and Mauro Maggioni. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 2007.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, 2000.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 2018.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. *Foundations and Trends in Machine Learning*, 2019.

Bilal Piot, Matthieu Geist, and Olivier Pietquin. Learning from demonstrations: Is it worth estimating a reward function? In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2013.

Bilal Piot, Matthieu Geist, and Olivier Pietquin. Boosted and reward-regularized classification for apprenticeship learning. In *International conference on autonomous agents and multi-agent systems*, 2014.

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

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

Siddharth Reddy, Anca D Dragan, and Sergey Levine. Sqil: imitation learning via regularized behavioral cloning. *International Conference on Learning Representations*, 2020.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics*, 2010.

Stuart Russell. Learning agents for uncertain environments. In *Conference on Computational learning theory*, 1998.

Yannick Schroecker and Charles L Isbell. State aware imitation learning. In *Advances in Neural Information Processing Systems*, 2017.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, 2015.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2018.

Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 1995.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, 2012.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *International Joint Conference on Artificial Intelligence*, 2018.

Cédric Villani. *Optimal transport: old and new*. 2008.

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

Huang Xiao, Michael Herman, Joerg Wagner, Sebastian Ziesche, Jalal Etesami, and Thai Hong Linh. Wasserstein adversarial imitation learning. *arXiv preprint arXiv:1906.08113*, 2019.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2008.

A LOCOMOTION EXPERIMENTS

A.1 PWIL IMPLEMENTATION

The agent we use is DDPG (Lillicrap et al., 2016) with a distributional critic (Barth-Maron et al., 2018). We use the default architecture from the ACME framework (Hoffman et al., 2020).

The actor architecture is a 4-layer neural network: the first layer has size 256 with tanh activation and layer normalization (Ba et al., 2016), the second layer and third layer have size 256 with elu activation (Clevert et al., 2016), the last layer is of size the dimension of the action space, with a tanh activation scaled to the action range of the environment. To enable sufficient exploration with use a Gaussian noise layer on top of the last layer with standard deviation $\sigma = 0.2$, that we clip to the action range of the environment. We evaluate the agent without exploration noise.

For the critic network we use a 4-layer neural network: the first layer has size 512 with tanh activation and layer normalization, the second layer is of size 512 with elu activation, the third layer is of size 256 with elu activation, the last layer is of dimension 201 with a softmax activation. The 201 neurons stand for the weights of the distribution supported within equal distance in the range $[-150, 150]$, as a categorical distribution (Bellemare et al., 2017).

We use the Adam optimizer (Kingma & Ba, 2015) with $\lambda_a = 5\times10^{-5}$ for the actor and $\lambda_r = 7\times10^{-5}$ for the critic. We use a batch size of 256. We clip both gradients from the critic and the actor to limit their L2 norm to 40.

We use a replay buffer of size 10^6, a discount factor $\gamma = 0.99$ and n step returns with $n = 5$. We prefill the replay buffer with 50000 state-action pairs from the set of demonstrations (which means that we put multiple times the same expert transitions in the buffer). We perform updates on the actor and the critic every $k = 4$ interactions with the environment. The hyperparameters search is detailed in Table 1.

Parameters	Values
λ_a	$10^{-5}, 5\times10^{-5}, 7\times10^{-5}, 10^{-4}$
λ_c	$10^{-5}, 5\times10^{-5}, 7\times10^{-5}, 10^{-4}$
σ	$0.1, 0.2, 0.3$
k	$2, 4, 8, 16$

Table 1: DDPG hyperparameters search.

For the reward function, we did run a hyperparameter search on α and β with the following values for α and β in $\{1, 5, 10\}$.

A.2 DAC IMPLEMENTATION

We used the open-source implementation of DAC provided by Kostrikov et al. (2019). For Humanoid (which was not reported in the paper), we did run a large hyperparameter search described in Table 2. Out of the 729 experiments, 7 led to an average performance above 1000 at the 2.5M training environment steps mark.

Actor learning rate (lr)	Critic lr	Discriminator lr	WGAN regularizer	Exploration noise	Decay lr
0.001, 0.0005, 0.0001	0.001, **0.0005**, 0.0001	**0.001**, 0.0005, 0.0001	5, **10**, 20	**0.1**, 0.2, 0.3	0.3, 0.5, **0.8**

Table 2: Hyperparameter search on Humanoid for 11 demonstrations. (tested/**best**)

A.3 BC IMPLEMENTATION

We used a 3-layer neural network, the first layer has size 128 with relu activation, the second layer has size 64 and the last layer has size of the action space with a tanh activation scaled to the action range of the environment. We found out that normalizing the observations with the average and standard deviation of the expert demonstrations' observations helped performance. We trained the network using the mean squared error as a loss, with an Adam optimizer. We ran an hyperpameter search on the learning rate in $\{10^{-5}, 10^{-4}, 10^{-3}\}$ and on the batch size $\{128, 256\}$.

A.4 PWIL LEARNING CURVES

We present the learning curves of PWIL, in terms of the reward defined in Equation 6 as well as the original reward of the task.

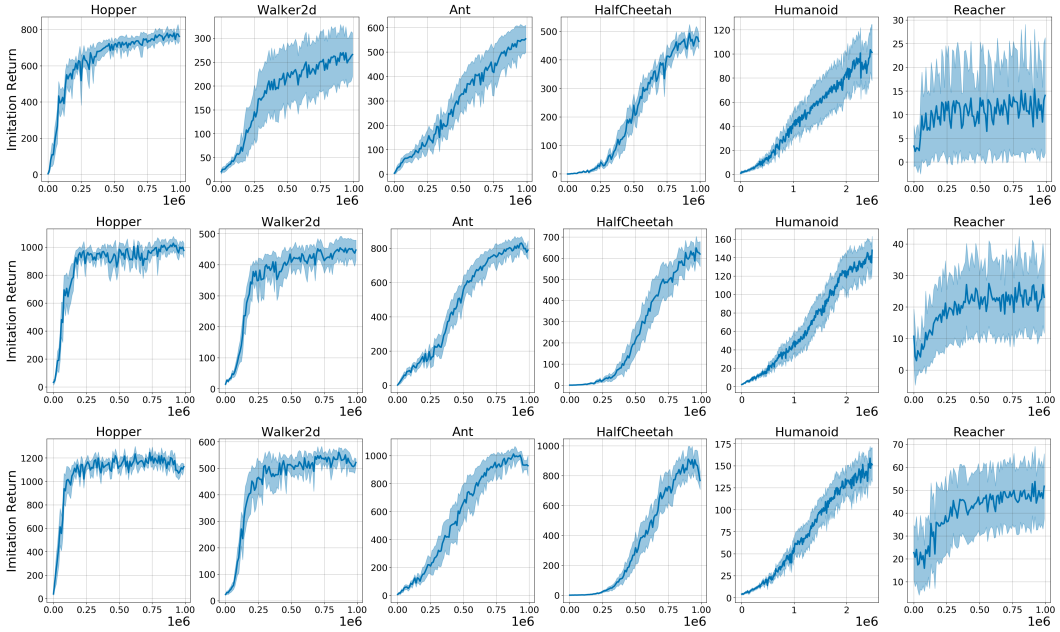

Figure 6: Mean and standard deviation of the imitation return of the agent. We perform 10 rollouts over 10 seeds every 10k environment steps over 1M steps. The return here is in term of the reward defined with Equation (6). Top row: 1 demonstration, middle row: 4 demonstrations, bottom row 11 demonstrations.

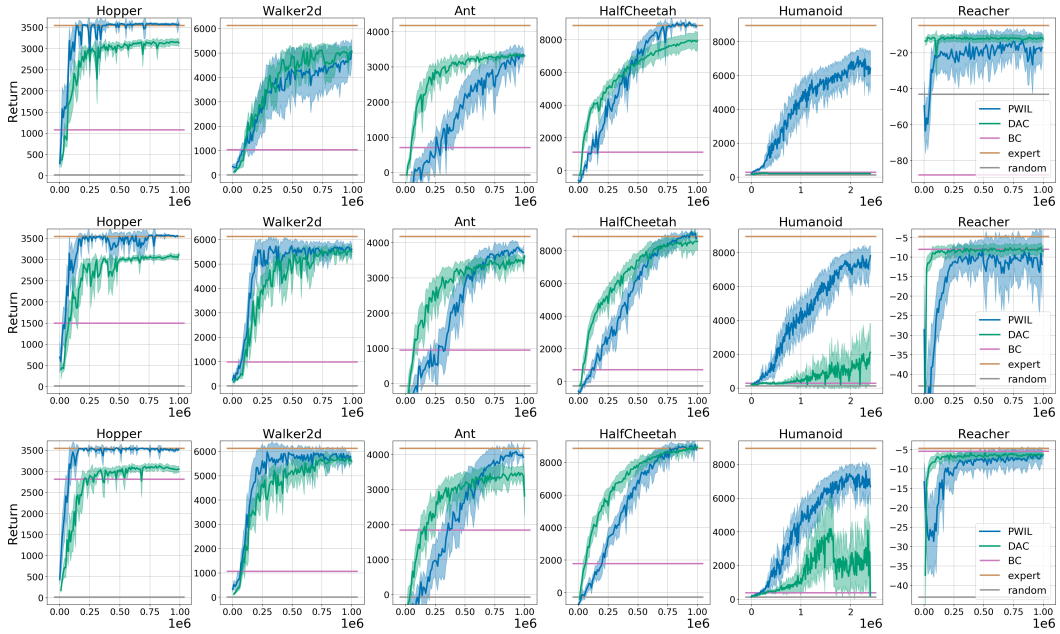

Figure 7: Mean and standard deviation of the original environment returns of the evaluation policy over 10 rollouts and 10 seeds, reported every 10k environment steps over 1M steps. Top row: 1 demonstation, middle row: 4 demonstrations, bottom row: 11 demonstrations.

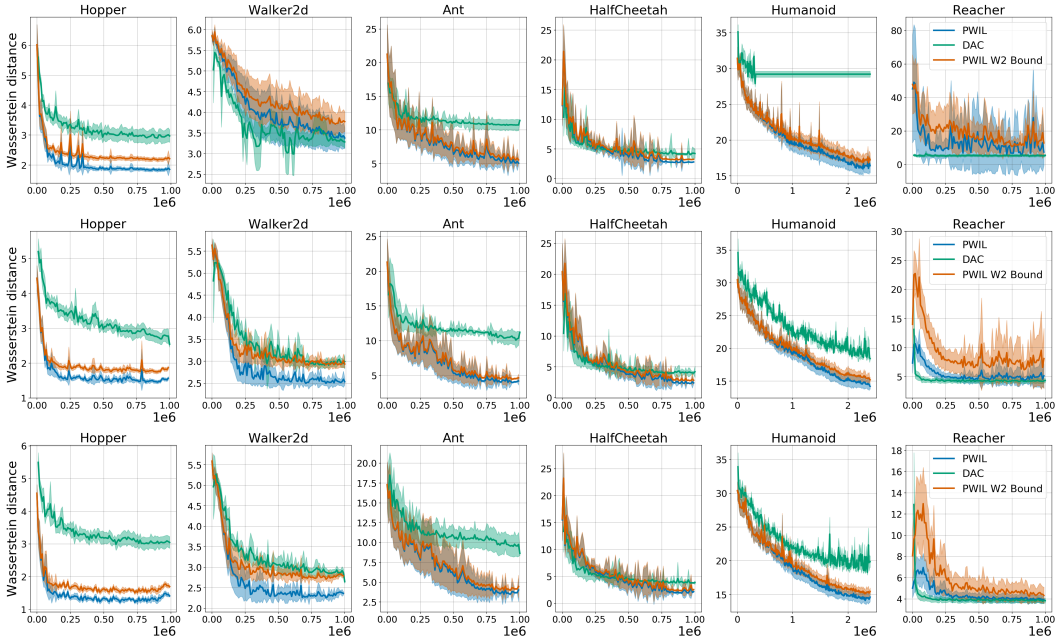

Figure 8: Mean of the Wasserstein distance between the state-action distribution of the evaluation policy and the state-action distribution of the expert over 10 rollouts and 10 seeds, reported every 10k environment steps. For PWIL, we include the upper bound on the Wasserstein distance based on the greedy coupling defined in Equation (4). Top row: 1 demonstration, middle row: 4 demonstrations, bottom row: 11 demonstrations.

B DOOR OPENING EXPERIMENTS

In this section we present the details of the experiment presented in Section 4.4. We are working on open-sourcing the code of these experiments.

B.1 ENVIRONMENT

The default door opening environment has a fixed horizon of 200 timesteps. We add an early termination condition to the environment when the door is opened to demonstrate that PWIL extends to tasks where there is no incentive for survival. Note that similarly to AIRL (Fu et al., 2018), early termination states are considered to be absorbing. Therefore if the expert reaches an early termination state early, its state distribution as a large weight on these states, which leads the PWIL agent to similarly reach the early termination state. 25 demonstrations were collected from humans using a virtual reality system (Kumar & Todorov (2015)).

B.2 EMBEDDING

As we assume that we only have access to the visual rendering of the demonstrations (with resolution 84×84), we build a lower dimensional latent space. We use a self-supervised learning method: TCC, to learn the latent space. We used the classification version of the algorithm detailed in Dwibedi et al. (2019) (Section 3.2).

The encoding network is the following: a convolutional layer with 128 filters of size 8 and stride 4, a relu activation, a convolutional layer with 64 filters of size 4 and stride 2, a relu activation, a convolutional layer with 64 filters of size 3 and stride 1, a relu activation, a linear layer of size 128, a relu activation and an output layer of size 32.

We separated the demonstrations into a train set of 20 demonstrations and a validation set of 5 demonstrations. Similarly to Dwibedi et al. (2019), we select the encoding network by taking the one that maximizes the alignment according to Kendall's tau score on the validation set.

We ran an hyperparameter search over the batch size $\{1, 4, 16, 32\}$, the subsampling ratio $\{1, 4, 8\}$ and the learning rate $\{10^{-2}, 10^{-3}, 10^{-4}\}$ (using the Adam optimizer). The best performing encoder had an average Kendall's tau score of 0.98 on the training set and 0.91 on the validation set.

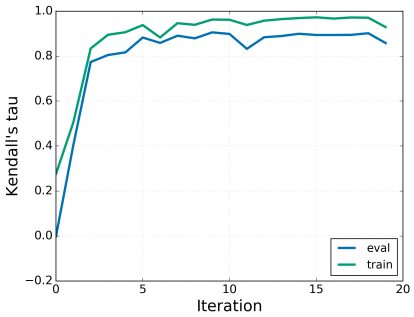

Figure 9: Kendall's tau score of the encoding network on the train and validation sets. An iteration corresponds to 1000 gradient updates.

B.3 AGENT

In this setup, we found that SAC leads to better results than D4PG. We detail the implementation of SAC below.

The actor is a 3-layer network that outputs the parameters of a normal distribution with diagonal covariance matrix. We enforce the terms of the covariance matrix to be greater than 10^{-3}. The first layer has size 256 with relu activation, the second layer has size 256 with relu activation and the last layer has size 48 (the action space has size 24).

The critic is a 3-layer network. We pass the concatenation of the state and the action of the environment as an input. There are two hidden layers of size 256 with relu activation. We do use a twin critic.

We use an adaptive temperature α as described by Haarnoja et al. (2018b).

For the three losses we use the Adam optimizer, and we use Polyak averaging for the critic networks with a rate τ. We ran an hyperparameter search over the learning rates $\lambda_a, \lambda_c, \lambda_\alpha$ in $\{10^{-4}, 3 \times 10^{-4}\}$, over τ $\{0.01, 0.005\}$, over the batch size $\{64, 128, 256\}$ and over the number of interactions with the environment between each actor, critic and temperature update $\{1, 4, 8\}$.

C ABLATION STUDY

In this section, we present the evaluation performance of PWIL in the presence of ablations and report results in Figure 10. We keep the hyperparameters of the original PWIL agent fixed.

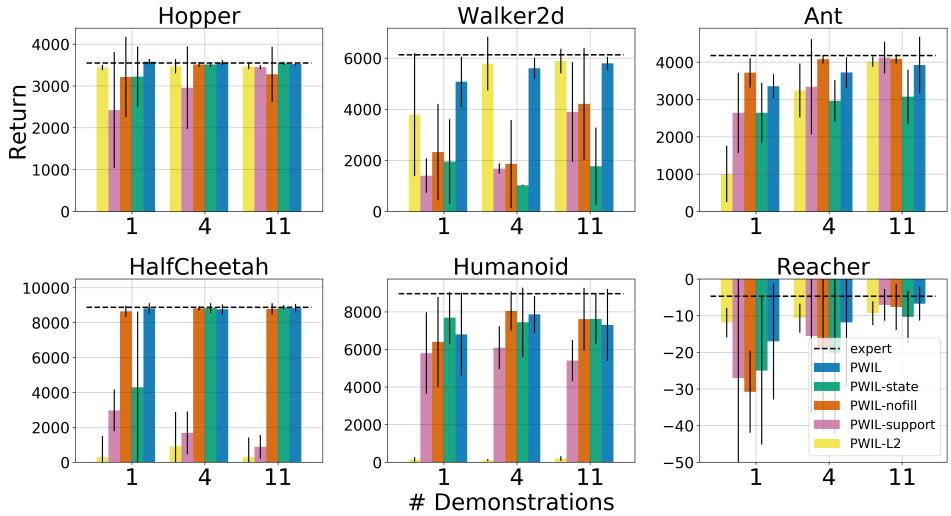

Figure 10: Mean and standard deviation of the evaluation performance of PWIL and ablated variants at the 1M environment interactions mark. The results are computed over 10 seeds and 10 rollouts for each of the seed.

PWIL-state In this version of PWIL, we do not assume that we have access to the actions taken by the expert. Therefore, we try to match the state distribution of the agent with the state distribution of the expert (instead of the state-action distribution). The setup is referred to as Learning from Observation (LfO) (Torabi et al., 2018; Edwards et al., 2019). The reward is defined similarly to PWIL, using a state distance rather than a state-action distance. Note that in this version, we cannot pre-fill the replay buffer with expert state-action pairs since we do not have access to actions. Remarkably, PWIL-state recovers non-trivial behaviors on all environments.

PWIL-L2 In this version of PWIL, we use the Euclidean distance between state-action pairs, rather than the standardized Euclidean distance. In other words, we do not weight the state-action distance by the inverse standard deviation of the expert demonstrations along each dimension. There is a significant drop in performance for all environments but Hopper-v2. This shows that the performance of PWIL is sensitive to the quality of the MDP metric.

PWIL-nofill In this version, we do not prefill the replay buffer with expert transitions. This leads to a drop in performance which is significant for Walker and Ant. This should not come as a surprise since a number of IL methods leverage the idea of expert transitions in the replay buffer (Reddy et al., 2020; Hester et al., 2018).

PWIL-support In this version of PWIL, we define the reward as a function of the following cost:

$$\forall i \in [1:T],\ c_{i,\pi} = \inf_{\substack{\gamma_1,\ldots,\gamma_D \in \mathbb{R} \\ \sum_{j=1}^{D} \gamma_j \leq \frac{1}{T}}} \sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^2 \gamma_j.$$

We can interpret this cost in the context of Section 3 as the problem of moving piles of dirt of mass $\frac{1}{T}$ into holes of infinite capacity. It is thus reminiscent of methods that consider IL as a support estimation problem (Wang et al., 2019; Brantley et al., 2020). This leads to a significant drop in performance for Ant, Walker and HalfCheetah.

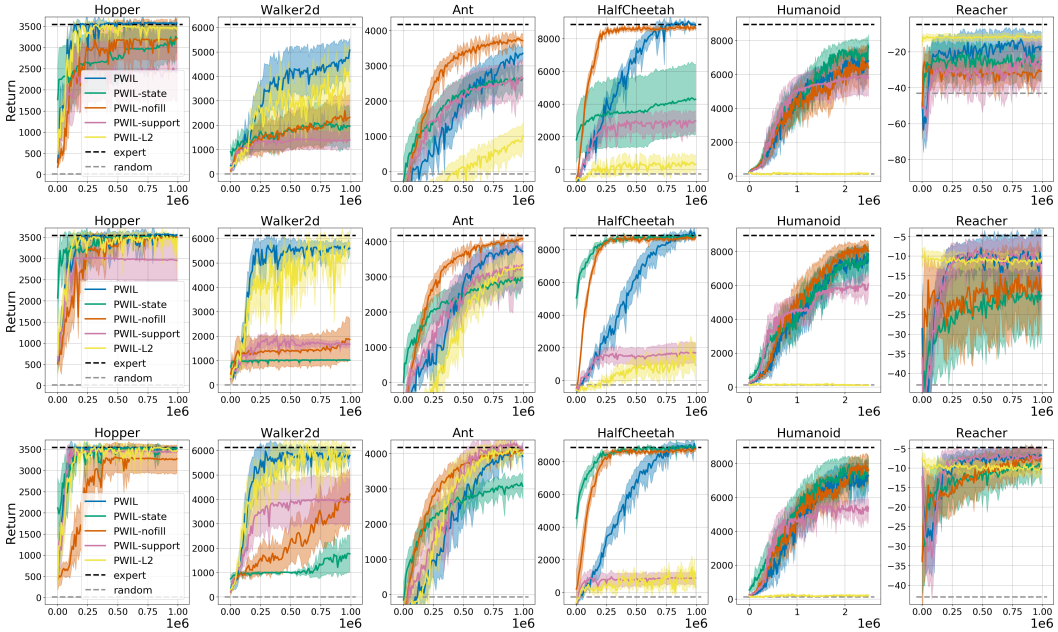

Figure 11: Mean of the evaluation performance of different variants of PWIL over 10 rollouts and 10 seeds, reported every 10k environment steps over 1M steps. The return here is in term of the environment's original reward. Top row: 1 demonstation, middle row: 4 demonstrations, bottom row: 11 demonstrations.

D ALGORITHM RUNDOWN

We present an example rundown of PWIL in Figure 12.

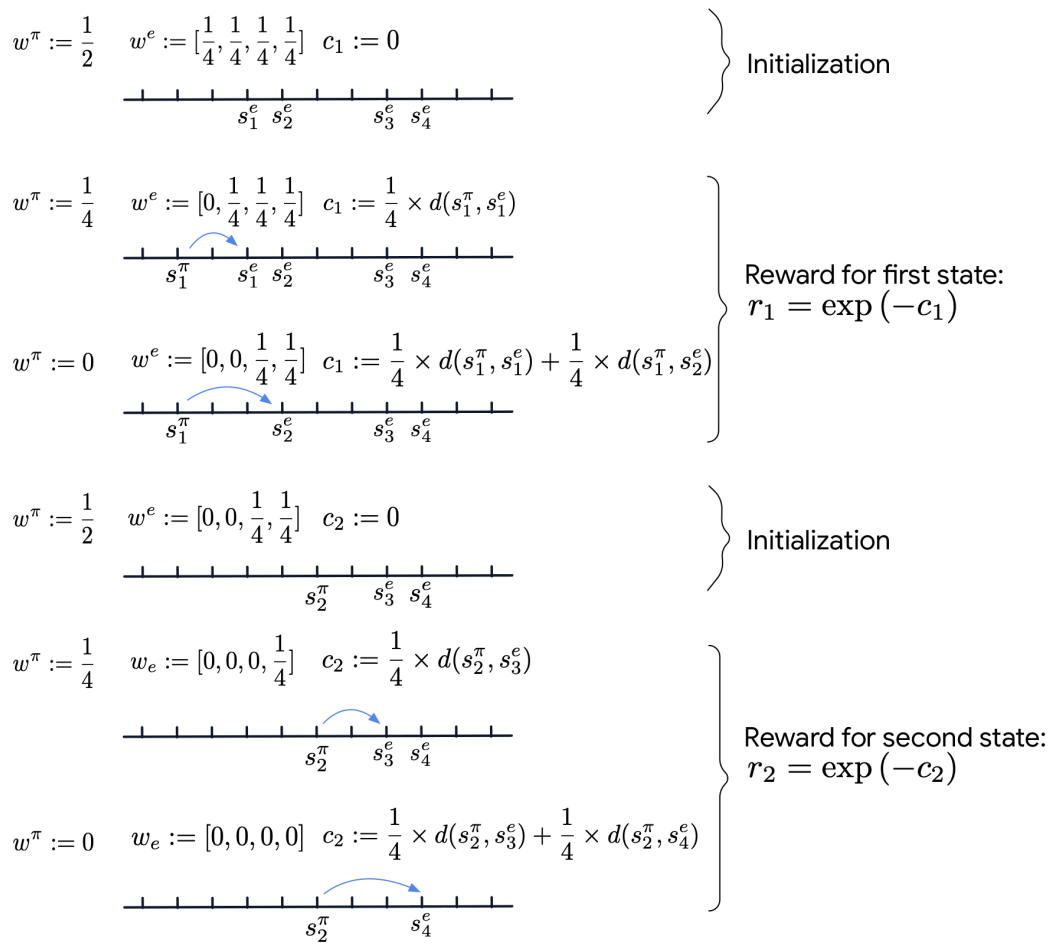

Figure 12: PWIL example rundown. We drop the dependency on the action. The example task has an horizon $T = 2$, and the demonstration set has $D = 4$ elements. We take $f : c \mapsto \exp(-c)$.