A Pragmatic Look at Deep Imitation Learning

Kai Arulkumaran KAI_ARULKUMARAN@ARAYA.ORG
Dan Ogawa Lillrank DAN_OGAWA@ARAYA.ORG
Araya Inc., Tokyo, Japan

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

The introduction of the generative adversarial imitation learning (GAIL) algorithm has spurred the development of scalable imitation learning approaches using deep neural networks. Many of the algorithms that followed used a similar procedure, combining on-policy actor-critic algorithms with inverse reinforcement learning. More recently there have been an even larger breadth of approaches, most of which use off-policy algorithms. However, with the breadth of algorithms, everything from datasets to base reinforcement learning algorithms to evaluation settings can vary, making it difficult to fairly compare them. In this work we re-implement 6 different IL algorithms, updating 3 of them to be off-policy, base them on a common off-policy algorithm (SAC), and evaluate them on a widely-used expert trajectory dataset (D4RL) for the most common benchmark (MuJoCo). After giving all algorithms the same hyperparameter optimisation budget, we compare their results for a range of expert trajectories. In summary, GAIL, with all of its improvements, consistently performs well across a range of sample sizes, AdRIL is a simple contender that performs well with one important hyperparameter to tune, and behavioural cloning remains a strong baseline when data is more plentiful.

Keywords: imitation learning; inverse reinforcement learning; benchmarking

1. Introduction

Several years ago, Henderson et al. (2018) formally brought attention to the reproducibility crisis in deep reinforcement learning (DRL). Some solutions have been to settle on evaluation protocols for common benchmarks (Machado et al., 2018), improving the statistical tools with which we evaluate results (Agarwal et al., 2021), or simply just providing reliable algorithm implementations (Raffin et al., 2021). Still, sometimes it is necessary to take a step back and evaluate the (claimed) progress within different areas of machine learning (ML; Oliver et al., 2018; Musgrave et al., 2020). In on-policy RL it has already been observed that implementation details can matter more than the algorithmic contributions of novel algorithms (Engstrom et al., 2020; Andrychowicz et al., 2021), and so in turn we have opted to take a *pragmatic* look at (deep) imitation learning (IL), creating a single, open source codebase to fairly compare algorithms.[1]

IL is the branch of ML that is concerned with learning from "demonstration" data (Hussein et al., 2017). In other words, there exists agents, acting in environments, from which we collect data, in order to train our own agent. IL is intimately linked with RL—learning

1. https://github.com/Kaixhin/imitation-learning

to act optimally in a given environment—and hence is often analysed with respect to concepts that are foundational to RL, such as states, actions, policies, reward functions, value functions, etc. (Sutton and Barto, 2018). Indeed, one of the most prominent approaches to IL is a technique known as inverse reinforcement learning (IRL; Arora and Doshi, 2021).

In the same way that deep learning (DL) has enabled RL to scale to high-dimensional state and action spaces (Arulkumaran et al., 2017), DL has also enabled IL to be applied to more complex domains. The most famous of these algorithms is generative adversarial IL (GAIL; Ho and Ermon, 2016), which uses the generative adversarial framework (Goodfellow et al., 2014) to learn a reward function which can then be used with IRL. In their work, they show strong results against a range of baselines, including the simplest IL method, behavioural cloning (BC; Pomerleau, 1988). Although BC underperformed GAIL, they noted that it "was able to reach satisfactory performance with enough data"—a common observation. However, to "generate the datasets, ... [they] subsampled the expert trajectories". While this reduces the number of samples available, it is not the same as providing fewer trajectories without subsampling, as the former tends to result in better coverage of the state space. *Pragmatically*, collecting trajectory data is a bottleneck for IL, so subsampling should be avoided—and in this case BC's performance improves. IL results where trajectory data has been subsampled is not directly comparable to results where no subsampling has occurred, and unsurprisingly practitioners may miss critical details revealed in an appendix.[2]

Another major issue in the reproducibility of IL algorithms is encountered in the GAIL paper—they generated their own expert trajectories using an RL agent. Out of the works that we review herein, nearly all generate their own expert data. And naturally, as time passes, the RL algorithms used within IL algorithms get replaced with more performant versions. One of the most significant of these changes is the move from on-policy RL algorithms, as used within the original work on GAIL, to more data-efficient off-policy algorithms (Kostrikov et al., 2019; Blondé and Kalousis, 2019). Do some of the on-policy IL algorithms that were competitive with GAIL (Kim and Park, 2018; Wang et al., 2019; Brantley et al., 2020) still perform competitively when converted to be off-policy?

In this work, we review 6 different deep IL algorithms: GAIL (Ho and Ermon, 2016), generative moment matching imitation learning (GMMIL; Kim and Park, 2018), random expert distillation (RED; Wang et al., 2019), disagreement-regularised imitation learning (DRIL; Brantley et al., 2020), adversarial reward-moment imitation learning (AdRIL; Swamy et al., 2021), and primal Wasserstein imitation learning (PWIL; Dadashi et al., 2021). To minimise differences between them, we update GMMIL, RED, and DRIL to be off-policy, give them access to absorbing state indicators (Kostrikov et al., 2019), and use soft-actor critic (SAC; Haarnoja et al., 2018a) as the base RL algorithm for all methods. We then evaluate them on the standard MuJoCo continuous control benchmark environments (Todorov et al., 2012; Brockman et al., 2016), using the D4RL expert trajectory datasets (Fu et al., 2020) for reproducibility. For a fairer comparison, we give each algorithm the same hyperparameter optimisation budget, and then run them with the best settings for several seeds, and report results using current best practices (Agarwal et al., 2021). Given the many improvements to GAIL, as well as prior effort performing extensive hyperparameter tuning on adversarial

2. Never mind implementation details *not* revealed within a paper.

IL methods (Orsini et al., 2021), unsurprisingly it remains one of the best IL algorithms to use. With more trajectories, AdRIL performs similarly to GAIL, whilst remaining simple to implement and tune (unlike GAIL). And, as observed many times before, BC becomes a competitive baseline with enough data.

2. Background

2.1. Imitation Learning

The goal of IL is to train a policy[3], $\hat{\pi}(a|s;\theta)$, mapping states s to a distribution over actions a, to mimic an expert policy $\pi^*(a|s)$, given either the expert policy itself, or more commonly, a fixed dataset $\xi^* = \{\tau_1, \ldots, \tau_N\}$, of trajectories $\tau = \{s_0, a_0, s_1, a_1, \ldots s_T, a_T\}$ generated by the expert, where N denotes the number of expert trajectories provided.

A common assumption within IL is that both the expert and our agent inhabit a Markov decision process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, p_0, \gamma)$: $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition dynamics, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $p_0(s)$ is the initial state distribution, and $\gamma \in [0, 1]$ is the discount factor (used to weight immediate vs. future rewards). The expert policy is optimal in the sense that $\pi^* = \text{argmax}_{\pi \in \Pi} \mathbb{E}_{\tau \sim \pi}[R_0]$, where the return at timestep t, R_t, is the discounted sum of rewards following a policy from state s_t until the end of the episode at timestep T: $R_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k+1}$. While in RL the goal is to interact with the environment in order to find π^* (Sutton and Barto, 2018), in IL we do not have access to $\mathcal{R}$, and must instead find π^* assuming that we have access to optimal trajectories.[4] All following methods, unless specified otherwise, can be implemented using neural networks, providing flexible function approximation that can scale to large state and/or action spaces.

2.2. Reduction to Supervised Learning

The simplest method, BC (Pomerleau, 1988), reduces IL to a supervised learning problem. Using a^* to denote the expert's actions, BC can be formulated as minimising the 1-step deviation from the expert trajectories:

$$\text{argmin}_{\theta} \mathbb{E}_{s,a^* \sim \xi^*}[\mathcal{L}(a^*, \hat{\pi}(a|s;\theta))], \tag{1}$$

where $\mathcal{L}$ can be, as in maximum likelihood estimation, the negative log likelihood.

BC is very simple, and benefits from a fixed objective over a stationary data distribution. However, as $\hat{\pi}$ is only trained on $s \sim \xi^*$, it can fail catastrophically when it diverges from the states covered by π^*. In order to mitigate this, $\hat{\pi}$ must be evaluated on the environment in order to correct for discrepancies between $s, a \sim \hat{\pi}$ and $s, a \sim \pi^*$.

Interactive IL methods solve this issue of compounding errors (Ross et al., 2011) by iterating over running $\hat{\pi}$ in the environment, calculating $\pi^*(a|s)$ on $\hat{\pi}$'s state distribution, and using supervised learning on the new data (Daumé et al., 2009; Ross and Bagnell, 2010; Ross et al., 2011). While these approaches solve the data distribution issue, they

3. In our case, a neural network with parameters θ.
4. In this work we do not consider the more complex settings that include suboptimal demonstrations and/or noisy observations.

require access to an interactive expert during training, which may not be available in many scenarios.

2.3. Inverse Reinforcement Learning

IRL instead overcomes this distribution shift by using RL to train $\hat{\pi}$ to mimic π^* in the environment. The procedure consists of iterating between the following two steps:

1. Construct a reward function[5] $\hat{\mathcal{R}}(s, a; \phi)$ using ξ^*, and optionally $\tau \sim \hat{\pi}$
2. Train $\hat{\pi}$ using RL

RL is more complicated than the typical supervised learning setting. In particular, as the policy evolves, the data distribution changes. In the case of IRL, $\hat{\mathcal{R}}$ changing over time can introduce further non-stationarity.

The basic objective of IRL can be stated as:

$$\operatorname*{argmax}_{\theta} \mathbb{E}_{\tau \sim \hat{\pi}(s,a;\theta)}[\hat{R}_0] \quad \text{such that} \quad \pi^* = \operatorname*{argmax}_{\pi \in \Pi} \mathbb{E}_{\tau \sim \pi}[\hat{R}_0], \tag{2}$$

where $\hat{R}$ is the return with respect to the learned reward function $\hat{\mathcal{R}}$. However, this is underspecified (Ng et al., 2000); e.g., any policy is trivially optimal for $\hat{\mathcal{R}} = 0$. IRL algorithms therefore incorporate one or several of the following three properties.

Firstly, one can match the state-action distribution under π^*, known as the expert's occupancy measure $\rho_{\pi^*} = \mathbb{E}_{\tau \sim \pi^*}\left[\sum_{t=0}^{T} \gamma^t \mathbb{1}_{s,a}\right]$ (Syed et al., 2008; Ho and Ermon, 2016), or, alternatively, feature expectations (Ng et al., 2000). This is achieved using the learned reward function, and is hence dependent on the expressivity of $\hat{\mathcal{R}}$. In particular, the constant function is underspecified and allows an infinite set of solutions. Secondly, one can "penalise" following trajectories taken by (previous iterations of) $\hat{\pi}$ (Ng et al., 2000). This allows $\hat{\mathcal{R}}$ to focus on relevant parts of the state-action space, but implicitly assumes that current/past versions of $\hat{\pi}$ are suboptimal. Thirdly, one can use the maximum entropy principle (Jaynes, 1957) to find a unique best solution out of the set of solutions that match the expert's occupancy measure/feature expectations (Ziebart et al., 2008). Using the Lagrangian multiplier λ, and denoting H as the entropy, this results in the following modified RL objective: $\operatorname{argmax}_{\theta} \mathbb{E}_{\tau \sim \pi(s,a;\theta)}[R_0] + \lambda H[\pi(s, a; \theta)]$. Entropy regularisation is a classic technique in RL (Williams and Peng, 1991).

A simple algorithm that uses these properties is soft Q imitation learning (SQIL; Reddy et al., 2020). SQIL uses the constant reward function:

$$\hat{\mathcal{R}} = \begin{cases} 1 & \text{if } (s, a) \in \xi^* \\ 0 & \text{if } (s, a) \sim \hat{\pi}, \end{cases} \tag{3}$$

which encourages not just imitating the expert's actions, but also visiting the same states. Building upon the maximum entropy model of expert behaviour (Ziebart et al., 2008), Reddy et al. (2020) show that their algorithm can be interpreted as regularised BC with a sparsity penalty on the reward function (Piot et al., 2014).[6] The full SQIL algorithm (for

5. In the parametric case, parameterised by ϕ, but potentially nonparametric.
6. Due to $\hat{\mathcal{R}}$ being $+1$ at expert state-action pairs, and 0 elsewhere.

Table 1: Adversarial imitation reward functions (Ghasemipour et al., 2020).

	$\hat{\mathcal{R}}$	Positive (bounded)	Negative (bounded)
GAIL	$\log D(s,a)$	✗(-)	✓(✗)
AIRL	$h(s,a) = \log(D(s,a)) - \log(1 - D(s,a))$	✓(✗)	✓(✗)
FAIRL	$-h(s,a) \cdot e^{h(s,a)}$	✓(✓)	✓(✗)

continuous action spaces) trains a SAC agent on half-half mixed batches of expert and agent data with its constant reward function. While simple, the downside of the constant reward function is that as the agent improves, its transitions still get labelled with zero rewards, potentially leading to a collapse in performance with over-training.

2.4. Adversarial Imitation Learning

Adversarial IL methods instead learn a reward function online using adversarial training, motivated by maximum entropy occupancy measure matching (Ho and Ermon, 2016). In generative adversarial network training (Goodfellow et al., 2014), the "generator" is trained to output samples that fool the "discriminator" $D : \mathcal{S} \times \mathcal{A} \to (0, 1)$, whilst the discriminator is trained to discriminate between samples from the generator and the data distribution. This is a minimax game, in which the equilibrium solution corresponds to minimising the Jensen-Shannon divergence between the generated and real distributions. In GAIL, $\hat{\pi}$ plays the role of the generator, and the discriminator is trained on state-action pairs from $\hat{\pi}$ and π^*: $\min_G \max_D \mathbb{E}_{s,a\sim\pi^*}[\log(D(s,a))] + \mathbb{E}_{s,a\sim\hat{\pi}}[\log(1 - D(s,a))]$. Under this formulation, higher values indicate how "expert" D believes its input to be.

There are several options for constructing $\hat{\mathcal{R}}$ from D. Prominent examples include those introduced in GAIL, adversarial inverse reinforcement learning (AIRL; Fu et al., 2018) (corresponding to the reverse Kullback-Leibler (KL) divergence $D_{\mathrm{KL}}(\rho_{\hat{\pi}} \| \rho_{\pi^*})$), and forward KL AIRL (FAIRL; Ghasemipour et al., 2020) (Table 1). As discussed by Kostrikov et al. (2019) and empirically investigated by Jena et al. (2020), there is a potential reward bias in these functions. They note that positive $\hat{\mathcal{R}}$, i.e., $-\log(1 - D(s,a))$, biases agents towards survival, whereas negative $\hat{\mathcal{R}}$, i.e., $\log(D(s,a))$ biases agents towards early termination. This bias means that even constant reward functions can outperform either of these depending on the type of the environment. We recommend the original works for discussions on the properties of various reward functions (Kostrikov et al., 2019; Jena et al., 2020; Ghasemipour et al., 2020). Kostrikov et al. (2019) also make the observation that many IL algorithms do not correctly handle terminal states, and propose appending an absorbing state indicator to states, which allows IRL algorithms to properly estimate values for terminal states. This requires processing complete trajectories from both the expert and the agent, and allowing both the RL agent and the discriminator to learn from the indicator feature.

While GAIL implicitly returns a reward function, if trained to optimality then D will return 0.5 for state-action pairs from both $\hat{\pi}$ and π^*. Finn et al. (2016) propose changing the form of the D to $\frac{\exp(f(\tau))}{\exp(f(\tau))+\hat{\pi}(\tau)}$, allowing the optimal reward function to be recovered as $f(+\text{const})$. AIRL makes a practical algorithm from this by changing D to operate over state-action pairs, as in GAIL, and also further disentangling the recovered reward function

f as the sum of a reward approximator $g(s, a)$ and a reward shaping term (Ng et al., 1999) $h(s)$: $f(s, a, s') = g(s, a) + \gamma h(s') - h(s)$, where s' is the successor state.

One of the most significant improvements to adversarial IL methods came from moving to more sample-efficient off-policy RL algorithms (Kostrikov et al., 2019; Blondé and Kalousis, 2019), which perform updates on batches of data stored in an experience replay memory (Lin, 1992). The discriminator can similarly be more efficiently trained on replay data, and although this should include an importance weighting term to account for the change in data distribution, in practice this is not needed (Kostrikov et al., 2019).

There are countless more advances within adversarial IL, making it difficult to know which techniques increase performance robustly. Orsini et al. (2021) performed a large-scale hyperparameter search over many of these methods. The key takeaways were that off-policy RL algorithms help improve sample efficiency, discriminator regularisation is key, and that hyperparameter choices which are optimal for AI-generated trajectories are not always the same for human-generated trajectories—a valuable distinction that lies out of the scope of this work. Their work also shows the importance of large-scale empirical evaluation, as their results overturned theoretical claims about the importance of discriminator regularisation (Blondé et al., 2022).

2.5. Distribution Matching Imitation Learning

One disadvantage of adversarial training is the requirement for the discriminator, which is also undergoing training as part of the minimax game, to provide a useful training signal to the generator. There are several other IRL algorithms that also attempt to match the expert and agent's state-action distributions, but use non-adversarial methods.

One solution is to replace the discriminator with a nonparametric model (Li et al., 2015; Dziugaite et al., 2015). Specifically, distribution matching can be achieved by minimising the maximum mean discrepancy (MMD; Gretton et al., 2012) defined over a reproducing kernel Hilbert space (RKHS). Given distributions, P and Q, and a mapping $\psi : \mathcal{X} \to \mathcal{H}$ from features $X \in \mathcal{X}$ to an RKHS $\mathcal{H}$, the MMD is the distance between the mean embeddings of the features: $\text{MMD}(P, Q) = \|\mathbb{E}_{x\sim P}[\psi(x)] - \mathbb{E}_{y\sim Q}[\psi(y)]\|_{\mathcal{H}}$. Using a kernel function k, one can calculate $\text{MMD}^2(P, Q) = \mathbb{E}_{x,x'\sim P}k(x, x') + \mathbb{E}_{y,y'\sim Q}k(y, y') - 2\mathbb{E}_{x\sim P,y\sim Q}k(x, y)$.

GMMIL (Kim and Park, 2018) extends this principle to the IL setting. Dropping terms that are constant with respect to $\hat{\pi}$, GMMIL has the reward function:

$$\hat{\mathcal{R}} = \frac{1}{M}\sum_{i=1}^{M} k((s, a), (s_i^*, a_i^*)) - \frac{1}{N}\sum_{j=1}^{N} k((s, a), (s_j, a_j)), \tag{4}$$

where M and N are the number of state-action pairs from π^* and $\hat{\pi}$, respectively.

Two disadvantages of GMMIL are that 1) the "discriminator" cannot learn relevant features, and 2) it has $O(MN)$ complexity. RED (Wang et al., 2019) solves these issues by building upon random network distillation (RND; Burda et al., 2018). In RND, a predictor network $f_\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^K$ is trained to minimise the mean squared error (MSE) against a fixed, randomly initialised network $f_{\bar{\phi}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^K$. Empirically, the MSE indicates how out-of-distribution new data is. RED uses a Gaussian function over the MSE, resulting in

$$\hat{\mathcal{R}} = \exp(-\sigma\|f_\phi(s, a) - f_{\bar{\phi}}(s, a)\|_2^2), \tag{5}$$

where σ is a bandwidth hyperparameter. Wang et al. (2019) interpret RND as an approximate support estimation method, and hence the RED reward function encourages the agent to have a support over its state-action distribution that matches the expert's.

Similarly to RED, DRIL (Brantley et al., 2020) constructs a reward function based on the disagreement between models trained on the expert data, and can also be interpreted as a support estimation method. However, unlike the other methods which operate over the joint distribution of state-action pairs, DRIL builds simply upon BC, operating over $p(a|s)$. DRIL first trains an ensemble of E different policies using the BC objective (Equation 1) on the expert data, and then uses a function of the (negative of the) variance between the policies to estimate a reward for the agent:

$$\hat{\mathcal{R}} = -C_{\mathrm{U}}^{\mathrm{clip}}(s,a) = \begin{cases} 1 & \text{if } \mathrm{Var}_{\pi \in \Pi_E}[\pi(a|s)] \leq q \\ -1 & \text{otherwise,} \end{cases} \tag{6}$$

where the q is a top quantile of the uncertainty cost computed over the expert dataset. While deep ensembles are known to produce reasonable uncertainty estimates (i.e., variance in outputs) on out-of-distribution data (Lakshminarayanan et al., 2017), it is also possible to approximate them using sampling with dropout (Srivastava et al., 2014). Brantley et al. (2020) showed empirically that this performed comparatively to using independent models.

The Wasserstein distance is another way of defining a distance between two probability distributions on a given metric space $\mathcal{M}$, and minimising it can be interpreted as finding the optimal coupling, γ, for transporting probability mass from one distribution to other, whilst minimising the transport cost given by a metric d on $\mathcal{M}$ (Villani, 2009).[7] PWIL (Dadashi et al., 2021) aims to minimise the Wasserstein-2 distance between the agent and expert's state-action distributions:

$$\inf_{\pi \in \Pi} \mathcal{W}_2^2(\rho_{\hat{\pi}}, \rho_{\pi^*}) = \inf_{\pi \in \Pi} \inf_{\gamma \in \Gamma} \sum_{t=1}^{T} \underbrace{\sum_{m=1}^{M} d((s_t, a_t), (s_t^*, a_t^*))^2 \gamma[t,m]}_{c_{t,\pi}}, \tag{7}$$

where $c_{t,\pi}$ is the (time-dependent) optimal transport cost.

The optimal coupling for policy π, γ_π^*, requires the full trajectory generated by π, so Dadashi et al. (2021) define a greedy coupling γ_π^g that transports probability mass at each timestep t, allowing the cost to be calculated online as the agent interacts with the environment. The cost with the greedy coupling, $c_{t,\pi}^g$, is an upper bound to the Wasserstein distance, and hence optimising it still minimises the distance between the agent and expert's state-action distribution. The reward can be defined by applying a monotonously decreasing function to the cost:

$$\hat{\mathcal{R}}_t = \alpha \exp(-\frac{\beta T}{\sqrt{|\mathcal{S}| + |\mathcal{A}|}} c_{t,\pi}^g), \tag{8}$$

where α and β are reward scale hyperparameters, and d is set to the Euclidean distance between Z-score normalised state-action pairs.

7. Note that in this subsection alone we use γ for couplings, in line with optimal transport literature.

Another view on distribution matching, known as moment matching[8], can be achieved through optimising integral probability metrics (IPM; Müller, 1997). IPMs provide a distance function between two distributions, $\sup_{f \in \mathcal{F}} \mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{y \sim Q}[f(y)]$, for a function class $\mathcal{F}$ containing functions $f : \mathcal{X} \to \mathbb{R}$. Different function classes $\mathcal{F}$ recover different IPMs; for instance, $\mathcal{F} = \{f : \|f\|_{\mathcal{H}} \leq 1\}$ with RKHS $\mathcal{H}$ gives the MMD and $\mathcal{F} = \{f : \|f\|_L \leq K\}$ with bounded Lipschitz constant K gives the Wasserstein distance.

Swamy et al. (2021) use this to provide a more general view on IL, arguing that training an agent to match the moments of the expert's reward or action-value distributions will achieve the same performance. When the agent is able to interact with the environment, Swamy et al. (2021) show that it is possible to do reward moment-matching, with a form that is similar to GMMIL, penalising the difference in moments between the agent and expert's state-action pairs. However, in their view on IL they also focus on the moment matching happening within a minimax game between the agent and the reward function, which motivates the need to update the reward function. Solving for a closed-form reward function in an RKHS with the indicator kernel function, the AdRIL reward function is:

$$\hat{\mathcal{R}} = \begin{cases} \frac{1}{|\xi^*|} & \text{if } (s,a) \in \xi^* \\ 0 & \text{if } (s,a) \sim \hat{\pi} \\ \frac{-1}{\text{round} \cdot |\xi|} & \text{if } (s,a) \sim \hat{\pi}_{\text{old}}, \end{cases} \tag{9}$$

where the final term assigns a negative reward to state-action pairs from old trajectories, inversely proportional to the number of rounds of updates (a hyperparameter that corresponds to a fixed number of updates), and the current number of agent trajectories, $|\xi|$. AdRIL can therefore be considered an improvement upon SQIL's constant reward function, obviating the need for early stopping (Reddy et al., 2020).

3. Experiments

3.1. Environments + Data

We evaluate all algorithms on the popular MuJoCo simulated robotics benchmarks: Ant, HalfCheetah, Hopper, and Walker2D (Todorov et al., 2012; Brockman et al., 2016). To improve reproducibility and enable fairer comparison against other reported results, we use the D4RL "expert-v2" trajectory datasets (Fu et al., 2020).[9] When loading the expert data we process each episode to distinguish between "true" and time-dependent terminations (Pardo et al., 2018), and provide absorbing state indicators (Kostrikov et al., 2019); these are also tracked for agent episodes. By default, we maximise available data by not subsampling expert transitions. We choose 3 different trajectory "budgets" for the IL algorithms to learn from: 5, 10 and 25 expert trajectories.

3.2. Algorithms + Hyperparameter Search + Evaluation

All IL algorithms use SAC (Haarnoja et al., 2018a), with automatic entropy tuning (Haarnoja et al., 2018b), as the base RL agent, as theoretically required by SQIL and AdRIL, and

8. Matching the moments of the model distribution to the empirical target distribution.
9. Although this benchmark was developed for offline RL, we use it for IL by ignoring the saved rewards.

as was empirically shown to be performant for adversarial IL algorithms (Orsini et al., 2021). The actor applies a tanh transformation to scale actions $\in (-1, 1)$, and dual critics are trained to reduce value overestimation (Fujimoto et al., 2018). We also include BC as a baseline, with the same actor architecture. All algorithms are optimised with AdamW (Loshchilov and Hutter, 2019). We use PyTorch (Paszke et al., 2019) for all of our code.

We group the IL algorithms and their variants into 6 key methods: AdRIL, DRIL, GAIL, GMMIL, PWIL, and RED. Our hyperparameter search spaces were determined based on hyperparameter ranges within the original works, a large subset of options tried by Orsini et al. (2021), and general hyperparameters such as learning rate and batch size, resulting in 7-18 hyperparameters to tune per algorithm. For each trajectory budget, we give each algorithm 30 hyperparameter evaluations using Bayesian optimisation (Balandat et al., 2020), with the minimum of the cumulative reward for each of the 4 environments used as the optimisation objective, which can be seen as minimising regret over a set of environments. We then evaluate agents over 10 seeds with the best hyperparameters found, and report performance according to best practices (Agarwal et al., 2021). We therefore train 2880 agents for the final results, not including the extensive training and testing of agents performed while replicating and augmenting IL algorithms.

DRIL, GAIL and RED include several options for their trained discriminators, including network hidden size, depth, activation function, dropout, and weight decay. The GAIL discriminator has additional options, detailed below.

AdRIL options include balanced sampling (alternating sampling expert and agent data batches vs. mixed batches, Swamy et al., 2021), and the discriminator update frequency ≥ 1, which determines the number of "rounds". We also include "0" in the search space, which if chosen reverts to using the SQIL reward function.

DRIL options include the quantile cutoff $\in [0, 1]$. As per the original work (Brantley et al., 2020), we include an auxiliary BC loss during training. For simplicity we use the dropout ensemble. As using absorbing state indicators requires importance sampling, we adapt the BC loss used within DRIL to account for importance weights.

GAIL options include reward shaping (Fu et al., 2018), subtracting $\log \hat{\pi}(a|s)$ from the predicted reward (Fu et al., 2018), the GAIL, AIRL and FAIRL reward functions (Ho and Ermon, 2016; Fu et al., 2018; Ghasemipour et al., 2020), discriminator gradient penalty (Kostrikov et al., 2019; Blondé and Kalousis, 2019), discriminator spectral normalisation (Blondé et al., 2022), discriminator entropy bonus ≥ 0 (Orsini et al., 2021), binary cross-entropy, Mixup, and nn-PUGAIL discriminator loss functions (Ho and Ermon, 2016; Chen et al., 2021; Xu and Denil, 2021), and 3 additional hyperparameters for these loss functions (Mixup alpha ≥ 0, positive class prior $\geq 0, \leq 1$, and non-negative margin ≥ 0).

To adapt GMMIL for absorbing states, we adapt it to use the weighted MMD (Yan et al., 2017). Due to the $O(MN \cdot \dim(X))$ complexity of MMD, it is prohibitive to use the entire expert dataset per update in the off-policy setting, and hence we randomly sample a batch of expert transitions per update. For this reason we also restrict the maximum batch size of GMMIL's hyperparameter search space.

PWIL options include the reward scale $\alpha \geq 0$ and reward bandwidth scale $\beta \geq 0$. To make PWIL more comparable to the other algorithms, we use the "nofill" variant, which does not prefill the replay buffer with expert transitions (Dadashi et al., 2021).

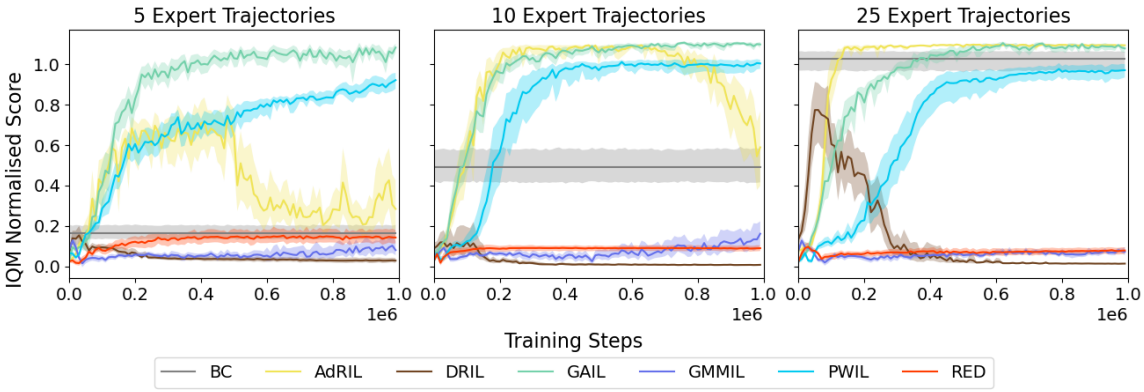

Figure 1: Normalised scores over different trajectory budgets (IQM ± 95% CI). GAIL performs well across all budgets, with AdRIL performing joint-best for high budgets. AdRIL's weakness is its sensitivity to the discriminator update frequency, which can cause policy collapse if not tuned well. PWIL is reasonably strong across budgets. BC is a strong baseline for high budgets.

To adapt RED for absorbing states, we train its discriminator with a weighted MSE loss.

The hyperparameter search spaces and optimised hyperparameters can be found documented in the codebase.

3.3. Results

All IL algorithms (except BC) are trained with 10^6 environment interaction steps, and are evaluated 30 times with a deterministic policy every 10^4 steps. To aggregate test returns, we calculate the interquartile mean (IQM) over the 30 evaluation episodes, and then once again over seeds, reporting the final IQM ± 95% confidence interval (CI) using 1000 stratified bootstrap samples (Agarwal et al., 2021). The final returns over all environments are reported in Table 2.[10] We also show performance over time in Figure 1, where the returns are aggregated over environments, normalised by the D4RL environment min and max reference scores: normalised score $= \frac{\text{score - random score}}{\text{expert score - random score}}$, with the reference scores produced by random and expert agents.

Overall, GAIL performs robustly across all trajectory budgets, presumably due to the large amount of research that has gone into improving the performance and robustness of adversarial IL methods. However, AdRIL, which is practically much simpler (and faster), performs the same (with higher sample efficiency) with higher budgets, as near-expert data from the agent is less likely to overpower the effect of the expert data. This can also be seen in the hyperparameter optimisation, with the discriminator update frequency going from 0 (reverting to the SQIL reward function) to 12500 to 25000 as the trajectory budget increased. PWIL also performed well across budgets. Hyperparameter optimisation set the reward (α) and reward bandwidth (β) scales to 1 for 5 trajectories, but increased the values

10. For reference we also include the results of our SAC implementation, trained for 3×10^6 steps.

Table 2: Returns over different trajectory budgets and environments (IQM ± 95% CI). Optimising for minimum regret across all environments results in hyperparameter choices that are suboptimal for individual environments. Although GAIL performs the best over all trajectory budgets for most environments, AdRIL is always the most effective in the Hopper environment.

# Trajectories		Ant	HalfCheetah	Hopper	Walker2D
	SAC	5155.97 ± 812.74	13994.64 ± 2304.52	2086.81 ± 608.70	5903.79 ± 340.09
5	BC	587.82 ± 255.80	52.48 ± 153.96	701.03 ± 134.96	909.28 ± 558.42
	AdRIL	1191.88 ± 787.27	36.58 ± 60.07	**3584.20 ± 30.61**	169.10 ± 576.42
	DRIL	−89.15 ± 84.87	−63.48 ± 20.80	367.35 ± 324.27	−20.69 ± 58.24
	GAIL	**5439.19 ± 253.85**	**11238.76 ± 217.55**	3517.76 ± 178.49	**5110.03 ± 66.72**
	GMMIL	575.01 ± 903.94	299.80 ± 272.75	820.35 ± 488.22	60.05 ± 56.30
	PWIL	3168.17 ± 1285.61	10775.81 ± 386.64	3529.61 ± 32.47	3678.89 ± 167.96
	RED	−90.94 ± 252.32	−111.72 ± 466.09	2500.67 ± 1024.09	1052.36 ± 184.32
10	BC	2178.80 ± 311.20	184.06 ± 176.27	1139.22 ± 559.33	4687.82 ± 380.03
	AdRIL	115.84 ± 520.84	5957.19 ± 2747.32	**3594.14 ± 8.50**	2267.06 ± 2121.80
	DRIL	−318.65 ± 104.92	−117.86 ± 32.44	37.51 ± 103.46	−12.32 ± 8.58
	GAIL	**5368.46 ± 54.49**	**11571.12 ± 154.39**	3580.90 ± 85.99	**5041.87 ± 51.92**
	GMMIL	590.47 ± 1022.66	448.40 ± 643.33	1325.33 ± 307.86	235.79 ± 196.27
	PWIL	5133.04 ± 72.93	10591.22 ± 1204.07	3558.52 ± 224.55	4185.11 ± 490.59
	RED	6.15 ± 168.53	978.77 ± 408.87	322.68 ± 62.43	216.50 ± 197.14
25	BC	4698.29 ± 207.67	5550.51 ± 1780.47	3271.88 ± 369.44	4955.85 ± 21.21
	AdRIL	5246.33 ± 70.14	11244.16 ± 129.42	**3579.96 ± 12.15**	4962.86 ± 15.00
	DRIL	1077.66 ± 475.96	−60.11 ± 27.60	9.24 ± 1.16	−29.86 ± 10.68
	GAIL	**5557.74 ± 81.01**	**11392.09 ± 377.15**	3467.73 ± 299.07	**5055.76 ± 64.13**
	GMMIL	−7.81 ± 176.54	620.01 ± 319.07	854.59 ± 244.31	13.29 ± 48.66
	PWIL	5040.18 ± 114.00	10785.65 ± 170.43	3418.18 ± 434.71	3855.43 ± 296.41
	RED	284.14 ± 181.12	−62.76 ± 181.27	323.86 ± 22.49	246.23 ± 165.69

for these for 10 and 25 trajectories. However, we note that since each environment step requires computing the reward between the current state-action pair and a subset of the expert data, starting with the entire dataset at the start of each episode, it is computationally expensive for high budgets. BC scales well as the number of trajectories increase.

Unfortunately, we were unable to successfully optimise off-policy versions of DRIL, GMMIL and RED. In an early version of our codebase we were able to optimise their original, on-policy versions successfully, so with considerable effort put into hyperparameter tuning and trying additional regularisation strategies, we believe that their is some fundamental issue caused from going from training on on-policy to off-policy returns. One would expect that for successful training the inferred rewards for the agent's trajectories should increase over time, but this was observed for runs of these methods as well, and is therefore not predictive of successful imitation. Q-values are a function of the predicted rewards, so did not provide further diagnostic insights. We also created variants of DRIL and RED in which the discriminators were trained online, similarly to GAIL, but were unsuccessful; however, there are many ways to do so, and our attempts do not preclude a successful online discriminator variant from being developed. Finally, we note that DRIL performs better at the start of training, which we can attribute to the BC auxiliary loss; experiments with uncertainty-only DRIL (UO-DRIL; Brantley et al., 2020) did not show this trend, with scores remaining low during the entirety of training.

Some weak trends we noticed from hyperparameter optimisation were that both batch and discriminator sizes increased with the trajectory budget. We hypothesise that the former is due to added stochasticity in optimisation aiding when data is scarce, whilst the latter is due to the need to prevent overfitting in low-data regimes. However, we caution that these trends do not always hold, as, for example, the optimal batch size for GAIL decreased with the trajectory budget.

4. Discussion

In this paper, we took a pragmatic look at deep IL methods, reviewing the relationships between the different approaches, updated older methods to use more data-efficient off-policy RL algorithms, and finally performed a fair comparison between them on a standard benchmark. As BC is simple and does not involve environment interaction, we recommend that it should always be considered as a baseline. AdRIL is an attractive option for deep IL due to its simplicity and strong performance, although it has one critical hyperparameter that needs tuning. And although the myriad of options for GAIL make it more complicated to work with, we have empirical data on what does and doesn't work (Orsini et al., 2021).

Although we were only able to test extensively on standard environments with expert data, we plan to release our framework to enable further, fair experiments on different environments, datasets, algorithms. Valuable open questions in the field of IL remain in the use of proxy reward functions for evaluating IL (Hussenot et al., 2021), and how best to learn from human demonstration data (Orsini et al., 2021).

Acknowledgments

This work was supported by JST, Moonshot R&D Grant Number JPMJMS2012.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In *NeurIPS*, 2021.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What Matters in On-policy Reinforcement Learning? A Large-scale Empirical Study. In *ICLR*, 2021.

Saurabh Arora and Prashant Doshi. A Survey of Inverse Reinforcement Learning: Challenges, Methods and Progress. *Artif. Intell.*, 297:103500, 2021.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep Reinforcement Learning: A Brief Survey. *IEEE SPM*, 34(6):26–38, 2017.

Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *NeurIPS*, 2020.

Lionel Blondé and Alexandros Kalousis. Sample-efficient Imitation Learning via Generative Adversarial Nets. In *AISTATS*, 2019.

Lionel Blondé, Pablo Strasser, and Alexandros Kalousis. Lipschitzness is All You Need to Tame Off-policy Generative Adversarial Imitation Learning. *Mach. Learn.*, 111(4): 1431–1521, 2022.

Kianté Brantley, Wen Sun, and Mikael Henaff. Disagreement-regularized Imitation Learning. In *ICLR*, 2020.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv:1606.01540*, 2016.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by Random Network Distillation. In *ICLR*, 2018.

Annie S Chen, HyunJi Nam, Suraj Nair, and Chelsea Finn. Batch Exploration with Examples for Scalable Robotic Reinforcement Learning. *IEEE RA-L*, 6(3):4401–4408, 2021.

Robert Dadashi, Leonard Hussenot, Matthieu Geist, and Olivier Pietquin. Primal Wasserstein Imitation Learning. In *ICLR*, 2021.

Hal Daumé, John Langford, and Daniel Marcu. Search-based Structured Prediction. *Mach. Learn.*, 75(3):297–325, 2009.

Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training Generative Neural Networks via Maximum Mean Discrepancy Optimization. In *UAI*, 2015.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO. In *ICLR*, 2020.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A Connection Between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-based Models. *arXiv:1611.03852*, 2016.

Justin Fu, Katie Luo, and Sergey Levine. Learning Robust Rewards with Adversarial Inverse Reinforcement Learning. In *ICLR*, 2018.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for Deep Data-driven Reinforcement Learning. *arXiv:2004.07219*, 2020.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing Function Approximation Error in Actor-critic Methods. In *ICML*, 2018.

Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A Divergence Minimization Perspective on Imitation Learning Methods. In *CoRL*, 2020.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. In *NeurIPS*, 2014.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A Kernel Two-sample Test. *JMLR*, 13(1):723–773, 2012.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-critic: Off-policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *ICML*, 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:1812.05905*, 2018b.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning that Matters. In *AAAI*, 2018.

Jonathan Ho and Stefano Ermon. Generative Adversarial Imitation Learning. In *NeurIPS*, 2016.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation Learning: A Survey of Learning Methods. *ACM CSUR*, 50(2):1–35, 2017.

Leonard Hussenot, Marcin Andrychowicz, Damien Vincent, Robert Dadashi, Anton Raichuk, Lukasz Stafiniak, Sertan Girgin, Raphael Marinier, Nikola Momchev, Sabela Ramos, et al. Hyperparameter Selection for Imitation Learning. In *ICML*, 2021.

Edwin T Jaynes. Information Theory and Statistical Mechanics. *Phys. Rev.*, 106(4):620, 1957.

Rohit Jena, Siddharth Agrawal, and Katia Sycara. Addressing Reward Bias in Adversarial Imitation Learning with Neutral Reward Functions. In *Deep RL Workshop, NeurIPS*, 2020.

Kee-Eung Kim and Hyun Soo Park. Imitation Learning via Kernel Mean Embedding. In *AAAI*, 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 *ICLR*, 2019.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *NeurIPS*, 2017.

Yujia Li, Kevin Swersky, and Rich Zemel. Generative Moment Matching Networks. In *ICML*, 2015.

Long-Ji Lin. Self-improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Mach. Learn.*, 8(3-4):293–321, 1992.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019.

Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *JAIR*, 61:523–562, 2018.

Alfred Müller. Integral Probability Metrics and Their Generating Classes of Functions. *Adv. Appl. Probab.*, 29(2):429–443, 1997.

Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A Metric Learning Reality Check. In *ECCV*, 2020.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *ICML*, 1999.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for Inverse Reinforcement Learning. In *ICML*, 2000.

Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic Evaluation of Deep Semi-supervised Learning Algorithms. In *NeurIPS*, 2018.

Manu Orsini, Anton Raichuk, Léonard Hussenot, Damien Vincent, Robert Dadashi, Sertan Girgin, Matthieu Geist, Olivier Bachem, Olivier Pietquin, and Marcin Andrychowicz. What Matters for Adversarial Imitation Learning? In *NeurIPS*, 2021.

Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time Limits in Reinforcement Learning. In *ICML*, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019.

Bilal Piot, Matthieu Geist, and Olivier Pietquin. Boosted and Reward-regularized Classification for Apprenticeship Learning. In *AAMAS*, 2014.

Dean A Pomerleau. ALVINN: An Autonomous Land Vehicle in a Neural Network. In *NeurIPS*, 1988.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable Reinforcement Learning Implementations. *JMLR*, 22(1):12348–12355, 2021.

Siddharth Reddy, Anca D Dragan, and Sergey Levine. SQIL: Imitation Learning via Reinforcement Learning with Sparse Rewards. In *ICLR*, 2020.

Stéphane Ross and Drew Bagnell. Efficient Reductions for Imitation Learning. In *AISTATS*, 2010.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-regret Online Learning. In *AISTATS*, 2011.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *JMLR*, 15 (1):1929–1958, 2014.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

Gokul Swamy, Sanjiban Choudhury, J Andrew Bagnell, and Steven Wu. Of Moments and Matching: A Game-theoretic Framework for Closing the Imitation Gap. In *ICML*, 2021.

Umar Syed, Michael Bowling, and Robert E Schapire. Apprenticeship Learning using Linear Programming. In *ICML*, 2008.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A Physics Engine for Model-based Control. In *IROS*, 2012.

Cédric Villani. *Optimal Transport: Old and New*. Springer, 2009.

Ruohan Wang, Carlo Ciliberto, Pierluigi Vito Amadori, and Yiannis Demiris. Random Expert Distillation: Imitation Learning via Expert Policy Support Estimation. In *ICML*, 2019.

Ronald J Williams and Jing Peng. Function Optimization using Connectionist Reinforcement Learning Algorithms. *Conn. Sci.*, 3(3):241–268, 1991.

Danfei Xu and Misha Denil. Positive-unlabeled Reward Learning. In *CoRL*, 2021.

Hongliang Yan, Yukang Ding, Peihua Li, Qilong Wang, Yong Xu, and Wangmeng Zuo. Mind the Class Weight Bias: Weighted Maximum Mean Discrepancy for Unsupervised Domain Adaptation. In *CVPR*, 2017.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum Entropy Inverse Reinforcement Learning. In *AAAI*, 2008.