Robust learning from observation with model misspecification

Robust Learning from Observation with Model Misspecification

Luca Viano
LIONS, EPFL
Lausanne, Switzerland
luca.viano@epfl.ch

Yu-Ting Huang
EPFL
Lausanne, Switzerland
y.t.huang.tp@gmail.com

Parameswaran Kamalaruban
The Alan Turing Institute
London, United Kingdom
kparameswaran@turing.ac.uk

Craig Innes
The University of Edinburgh
Edinburgh, United Kingdom
craig.innes@ed.ac.uk

Subramanian Ramamoorthy
The University of Edinburgh
Edinburgh, United Kingdom
s.ramamoorthy@ed.ac.uk

Adrian Weller
University of Cambridge,
The Alan Turing Institute
United Kingdom
aw665@cam.ac.uk

ABSTRACT

Imitation learning (IL) is a popular paradigm for training policies in robotic systems when specifying the reward function is difficult. However, despite the success of IL algorithms, they impose the somewhat unrealistic requirement that the expert demonstrations must come from the same domain in which a new imitator policy is to be learned. We consider a practical setting, where (i) state-only expert demonstrations from the real (deployment) environment are given to the learner, (ii) the imitation learner policy is trained in a simulation (training) environment whose transition dynamics is slightly different from the real environment, and (iii) the learner does not have any access to the real environment during the training phase beyond the batch of demonstrations given. Most of the current IL methods, such as generative adversarial imitation learning and its state-only variants, fail to imitate the optimal expert behavior under the above setting. By leveraging insights from the Robust reinforcement learning (RL) literature and building on recent adversarial imitation approaches, we propose a robust IL algorithm to learn policies that can effectively transfer to the real environment without fine-tuning. Furthermore, we empirically demonstrate on continuous-control benchmarks that our method outperforms the state-of-the-art state-only IL method in terms of the zero-shot transfer performance in the real environment and robust performance under different testing conditions.

KEYWORDS

Sim-to-real transfer; Imitation Learning; Learning from Observation; Robust Reinforcement Learning

ACM Reference Format:
Luca Viano, Yu-Ting Huang, Parameswaran Kamalaruban, Craig Innes, Subramanian Ramamoorthy, and Adrian Weller. 2022. Robust Learning from Observation with Model Misspecification. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 18 pages.

1 INTRODUCTION

Deep Reinforcement Learning (RL) [30, 32, 33] methods have demonstrated impressive performance in continuous control [18], and robotics [17]. However, a broader application of these methods in real-world domains is impeded by the challenges in designing a proper reward function [1, 7, 29]. Imitation Learning (IL) algorithms [12, 23, 41] address this issue by replacing reward functions with expert demonstrations, which are easier to collect in most scenarios. However, despite the success of IL algorithms, they typically impose the somewhat unrealistic requirement that the state-action demonstrations must be collected from the same environment as the one in which the imitator is trained. In this work, we focus on a more realistic setting for imitation learning, where:

(1) the expert demonstrations collected from the real (deployment) environment by executing an expert policy only contain states,
(2) the learner is trained in a simulation (training) environment, and does not have access to the real environment during the training phase beyond the batch of demonstrations given, and
(3) the simulation environment does not model the real environment exactly, i.e., there exists a transition dynamics mismatch between these environments.

The learned policy under the above setting is transferred to the real environment on which its final performance is evaluated. Existing IL methods either do not apply under the above setting or result in poor transfer performance.

A large body of work in IL, such as Generative Adversarial Imitation Learning (GAIL [12]) and its variants, has focused on the setting with demonstrations that contain both states and actions, which are difficult to obtain for real-world settings such as learning from videos [11]. Further, closely following the state-action demonstrations limits the the ability to generalize across environments [27]. Training agents in simulation environments not only provides data at low-cost, but also alleviates safety concerns related to the trial-and-error process with real robots. However, building a high-fidelity simulator that perfectly models the real environment would require a large computational budget. Low-fidelity simulations are feasible, due to their speed, but the gap between the simulated and real environments degrades the performance of the policies when transferred to real robots [39]. To this end, we consider the following research question: *how to train an imitator policy in an offline manner with state-only expert demonstrations and a misspecified simulator such that the policy performs well in the real environment?*

The Adversarial Inverse Reinforcement Learning (AIRL) method from [8] recovers reward functions that can be used to transfer behaviors across changes in dynamics. However, one needs to retrain a policy in the deployment environment with the recovered reward

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

Table 1: Comparison of our method with the existing imitation learning methods that also consider dynamics mismatch. However, the existing methods do not fit under the specific setting that we study. The expert, training, and deployment are denoted by M^{exp}, M^{tr}, and M^{dep} respectively. The corresponding transition dynamics are denoted by T^{exp}, T^{tr}, and T^{dep} respectively. Note that the expert demonstrations are collected from M^{exp}, the imitation learning agent is trained on M^{tr}, and the trained policy is finally evaluated on M^{dep}. Our Robust-GAILfO method has minimal access to M^{dep} to select an appropriate α. Note that our robust GAILfO method is applicable in both: (i) $T^{\text{dep}} = T^{\text{exp}} \neq T^{\text{tr}}$ setting, and (ii) $T^{\text{dep}} \neq T^{\text{exp}} \neq T^{\text{tr}}$ setting. In setting (i), our primary motivation is that accessing the deployment environment is costly, e.g., interacting with a remote deployment environment is costly due to communication constraints. In setting (ii), after the deployment, the agent has to be robust against potential environmental changes during the test time.

IL Methods	*Type of Demonstrations*	*Access to M^{dep} during training*	*Dynamics mismatch*
GAIL [12]	state-action	yes	$T^{\text{exp}} = T^{\text{tr}} = T^{\text{dep}}$
GAILfO [36]	state-only	yes	$T^{\text{exp}} = T^{\text{tr}} = T^{\text{dep}}$
AIRL [8]	state-action	yes	$T^{\text{exp}} = T^{\text{tr}} \neq T^{\text{dep}}$
I2L [9]	state-only	yes	$T^{\text{exp}} \neq T^{\text{tr}} = T^{\text{dep}}$
SAIL [20]	state-only	yes	$T^{\text{exp}} \neq T^{\text{tr}} = T^{\text{dep}}$
GARAT [3]	state-only	yes	$T^{\text{dep}} = T^{\text{exp}} \neq T^{\text{tr}}$
HIDIL [14]	state-action	no	$T^{\text{dep}} = T^{\text{exp}} \neq T^{\text{tr}}$
IDDM [38]	state-only	yes	$T^{\text{exp}} = T^{\text{tr}} = T^{\text{dep}}$
ILPO [5]	state-only	yes	$T^{\text{exp}} = T^{\text{tr}} = T^{\text{dep}}$
Robust-GAILfO (ours)	state-only	no	$T^{\text{dep}} = T^{\text{exp}} \neq T^{\text{tr}}$ and $T^{\text{dep}} \neq T^{\text{exp}} \neq T^{\text{tr}}$

function, whereas we consider a zero-shot transfer setting. In addition, AIRL depends on state-action demonstrations. Recently, [9, 20] have studied the imitation learning problem under the transition dynamics mismatch between the expert and the learner environments. However, they do not aim to learn policies that are transferable to the expert (real) environment; instead, they optimize the performance in the learner (simulation) environment. In [3], the authors attempt to match the simulation environment closer to the real environment by interacting with the real environment during the training phase. A setting very close to ours is considered in [14]; their method involves learning an inverse dynamics model of the real environment based on the state-action expert demonstrations. None of these methods are directly applicable under our setting (see Table 1).

We propose a robust IL method for learning robust policies under the above-discussed setting that can be effectively transferred to the real environment without further fine-tuning during deployment. Our method is built upon the robust RL literature [13, 24, 26, 34] and the IL literature inspired by GAN-based adversarial learning [12, 36]. In particular, our algorithm is a robust variant of the Generative Adversarial Imitation Learning from Observation (GAILfO [36]) algorithm, a state-only IL method based on GAIL. We discuss how our method addresses the dynamics mismatch issue by exploiting the equivalence between the robust MDP formulation and the two-player Markov game [26, 34]. In the finite MDP setting, [37] have proposed a robust inverse reinforcement learning method to address the transition dynamics mismatch between the expert and the learner. Our Markov game formulation in Section 4.1 closely follows that of [37], and in Section 4.2, we scale it high-dimensional continuous control setting using the techniques from GAIL literature. On the empirical side, we are interested in the sim-to-real transfer performance, whereas [37] have considered the performance in the learner environment itself.

We evaluate the efficacy of our method on the continuous control MuJoCo environments. In our experiments, we consider different sources of dynamics mismatch such as joint-friction, and agent-mass. An expert policy is trained under the default dynamics (acting as the real environment). The imitator policy is learned under a modified dynamics (acting as the simulation environment), where one of the mass and friction configurations is changed. The experimental results show that, with appropriate choice of the level of adversarial perturbation, the robustly trained IL policies in the simulator transfer successfully to the real environment compared to the standard GAILfO. We also empirically show that the policies learned by our method are robust to environmental shift during testing.

2 RELATED WORK

Imitation Learning. Ho and Ermon [12] propose a framework, called Generative Adversarial Imitation Learning (GAIL), for directly extracting a policy from trajectories without recovering a reward function as an intermediate step. GAIL utilizes a discriminator to distinguish between the state-action pairs induced by the expert and the learner policy. GAIL was further extended by Fu et al. [8] to produce a scalable inverse reinforcement learning algorithm based on adversarial reward learning. This approach gives a policy as well as a reward function. Our work is closely related to the state-only IL methods that do not require actions in the expert demonstrations [36, 38]. Inspired by GAIL, [36] have proposed the Generative Adversarial Imitation Learning from Observation (GAILfO) algorithm for state-only IL. GAILfO tries to minimize the divergence between the state transition occupancy measures of the learner and the expert.

Robust Reinforcement Learning. In the robust MDP formulation [13, 24], the policy is evaluated by the worst-case performance in a class of MDPs centered around a reference environment. In the context

of forward RL, some works build on the robust MDP framework, such as [21, 25, 28]. However, our work is closer to the line of work that leverages the equivalence between action-robust and robust MDPs. In [22], the authors have introduced the notion of worst-case disturbance in the H_∞-control literature to the reinforcement learning paradigm. They consider an adversarial game where an adversary tries to make the worst possible disturbance while an agent tries to make the best control input. Recent literature in RL has proposed a range of robust algorithms based on this game-theoretic perspective [4, 15, 26, 34].

3 PROBLEM SETUP AND BACKGROUND

This section formalizes the learning from observation (LfO) problem with model misspecification.

Environment and Policy. The environment is formally represented by a Markov decision process (MDP) $M_c := (\mathcal{S}, \mathcal{A}, T, \gamma, P_0, c)$. The state and action spaces are denoted by $\mathcal{S}$ and $\mathcal{A}$, respectively. $T : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ captures the state transition dynamics, i.e., $T(s' \mid s, a)$ denotes the probability of landing in state s' by taking action a from state s. Here, $c : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ is the cost function, $\gamma \in (0, 1)$ is the discounting factor, and $P_0 : \mathcal{S} \to [0, 1]$ is an initial distribution over the state space $\mathcal{S}$. We denote an MDP without a cost function by $M = M_c \backslash c = \{\mathcal{S}, \mathcal{A}, T, \gamma, P_0\}$. We denote a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ as a mapping from a state to a probability distribution over the action space. The set of all stationary stochastic policies is denoted by Π. For any policy π in the MDP M, we define the state transition occupancy measure as follows: $\rho_M^\pi(s, s') := \sum_a T(s' \mid s, a) \cdot \pi(a \mid s) \cdot \sum_{t=0}^\infty \gamma^t \mathbb{P}[S_t = s \mid \pi, M]$. Here, $\mathbb{P}[S_t = s \mid \pi, M]$ denotes the probability of visiting the state s after t steps by following the policy π in M. The total expected cost of any policy π in the MDP M_c is defined as follows: $\mathbb{E}_{\rho_M^\pi}[c(s, s')] := \mathbb{E}\left[\sum_{t=0}^\infty \gamma^t c(s_t, s_{t+1})\right]$, where $s_0 \sim P_0, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim T(\cdot|s_t, a_t)$. A policy π is *optimal* for the MDP M_c if $\pi \in \arg\min_{\pi'} \mathbb{E}_{\rho_M^{\pi'}}[c(s, s')]$, and we denote an optimal policy by $\pi^*_{M_c}$.

Learner and Expert. We have two entities: an imitation learner, and an expert. We consider two MDPs, $M^{\text{sim}} = \left\{\mathcal{S}, \mathcal{A}, T^{\text{sim}}, \gamma, P_0\right\}$ and $M^{\text{real}} = \left\{\mathcal{S}, \mathcal{A}, T^{\text{real}}, \gamma, P_0\right\}$, that differ only in the transition dynamics. The true cost function $c^* : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ is known only to the expert. The learner is trained in the MDP M^{sim} and is not aware of the true cost function, i.e., it only has access to $M^{\text{sim}}_{c^*} \backslash c^*$. The expert provides demonstrations to the learner by following the optimal policy $\pi^*_{M^{\text{real}}_{c^*}}$ in the expert MDP M^{real}. Typically, in the imitation learning literature, it is assumed that $T^{\text{sim}} = T^{\text{real}}$. In this work, we consider the setting where there is a transition dynamics mismatch between the learner and the expert, i.e., $T^{\text{sim}} \neq T^{\text{real}}$. The learner tries to recover a policy that closely matches the intention of the expert, based on the occupancy measure $\rho_E(s, s') := \rho^{\pi^*_{M^{\text{real}}_{c^*}}}_{M^{\text{real}}}(s, s')$ (or the demonstrations drawn according to it) received from the expert. The learned policy is evaluated in the expert environment w.r.t. the true cost function, i.e., $M^{\text{real}}_{c^*}$.

Imitation Learning. We consider the imitation learner model that matches the expert's state transition occupancy measure ρ_E [12, 36, 41]. In particular, the learner policy is obtained via solving the following primal problem:

$$\min_{\pi \in \Pi} \quad -H_{\rho^\pi_{M^{\text{sim}}}}(\pi) \tag{1}$$

$$\text{subject to} \quad \rho^\pi_{M^{\text{sim}}}(s, s') = \rho_E(s, s'), \quad \forall s, s' \in \mathcal{S}, \tag{2}$$

where $H_{\rho^\pi_{M^{\text{sim}}}}(\pi) := \mathbb{E}\left[\sum_{t=0}^\infty -\gamma^t \log \pi(a_t|s_t)\right]$ is the γ-discounted causal entropy of π. The corresponding dual problem is given by:

$$\max_{c \in \mathbb{R}^{\mathcal{S}\times\mathcal{S}}} \left(\min_{\pi \in \Pi} -H_{\rho^\pi_{M^{\text{sim}}}}(\pi) + \mathbb{E}_{\rho^\pi_{M^{\text{sim}}}}\left[c(s, s')\right]\right) - \mathbb{E}_{\rho_E}\left[c(s, s')\right],$$

where the costs $c(s, s')$ serve as dual variables for the equality constraints.

Maximum Causal Entropy (MCE) Inverse Reinforcement Learning (IRL). MCE-IRL algorithm [40, 41] involves a two-step procedure. First, it looks for a cost function $c \in C$ that assigns low cost to the expert policy and high cost to other policies. Then, it learns a policy by solving a certain reinforcement learning problem with the found cost function. Formally, given a convex cost function regularizer[1] $\psi : \mathbb{R}^{\mathcal{S}\times\mathcal{S}} \to \overline{\mathbb{R}}$, first, we recover a cost function $\tilde{c}$ by solving the following ψ-regularized problem:

$$\text{IRL}_\psi(\rho_E) = \arg\max_{c \in \mathbb{R}^{\mathcal{S}\times\mathcal{S}}} -\psi(c) - \mathbb{E}_{\rho_E}\left[c(s, s')\right] + \left(\min_{\pi \in \Pi} -\lambda H_{\rho^\pi_{M^{\text{sim}}}}(\pi) + \mathbb{E}_{\rho^\pi_{M^{\text{sim}}}}\left[c(s, s')\right]\right)$$

Then, we input the learned cost function $\tilde{c} \in \text{IRL}_\psi(\rho_E)$ into an entropy-regularized reinforcement learning problem:

$$\text{RL}(c) = \arg\min_{\pi \in \Pi} -\lambda H_{\rho^\pi_{M^{\text{sim}}}}(\pi) + \mathbb{E}_{\rho^\pi_{M^{\text{sim}}}}\left[c(s, s')\right],$$

which aims to find a policy that minimizes the cost function and maximizes the entropy.

Generative Adversarial Imitation Learning from Observation (GAILfO). Recently, [12, 36] have shown that, for a specific choice of the regularizer ψ, the two-step procedure $\text{RL} \circ \text{IRL}_\psi(\rho_E)$ of the MCE-IRL algorithm can be reduced to the following optimization problem using GAN discriminator:

$$\min_{\pi \in \Pi} \max_{D \in (0,1)^{\mathcal{S}\times\mathcal{S}}} -\lambda H_{\rho^\pi_{M^{\text{sim}}}}(\pi) + \mathbb{E}_{\rho^\pi_{M^{\text{sim}}}}\left[\log D(s, s')\right] + \mathbb{E}_{\rho_E}\left[\log\left(1 - D(s, s')\right)\right],$$

where $D : \mathcal{S} \times \mathcal{S} \to (0, 1)$ is a classifier trained to discriminate between the state-next state pairs that arise from the expert and the imitator. Excluding the entropy term, the above loss function is similar to the loss of generative adversarial networks [10]. Even though the occupancy measure matching methods were shown to scale well to high-dimensional problems, they are not robust against dynamics mismatch [9].

4 ROBUST LEARNING FROM OBSERVATION VIA MARKOV GAME

4.1 Markov Game

In this section, we focus on recovering a learner policy via imitation learning framework in a robust manner, under the setting described

[1] $\overline{\mathbb{R}}$ denotes the extended real numbers $\mathbb{R} \cup \{+\infty\}$

in Section 1. To this end, we consider a class of transition matrices such that it contains both T^{sim} and T^{real}. In particular, for a given $\alpha > 0$, we define the class $\mathcal{T}^{\alpha}$ as follows:

$$\mathcal{T}^{\alpha} := \left\{ \alpha T^{\text{sim}}(s'|s,a) + \bar{\alpha} \sum_{b} \pi(b|s) \cdot T^{\text{sim}}(s'|s,b), \forall \pi \in \Pi \right\}, \quad (3)$$

where $\bar{\alpha} = (1-\alpha)$. We define the corresponding class of MDPs as follows: $\mathcal{M}^{\alpha} := \{\{\mathcal{S}, \mathcal{A}, T^{\alpha}, \gamma, P_0\}, \forall T^{\alpha} \in \mathcal{T}^{\alpha}\}$. We need to choose α such that $M^{\text{real}} \in \mathcal{M}^{\alpha}$.

Our aim is to find a learner policy that performs well in the MDP $M^{\text{real}}_{c^*}$ by using the state-only demonstrations from ρ_E, without knowing or interacting with M^{real} during training. Thus, we try to learn a robust policy over the class $\mathcal{M}^{\alpha}$, while aligning with the expert's state transition occupancy measure ρ_E, and acting only in M^{sim}. By doing this, we ensure that the resulting policy performs reasonably well on any MDP $M \in \mathcal{M}^{\alpha}$ including M^{real} w.r.t. the true cost function c^*. Based on this idea, we propose the following robust learning from observation (LfO) problem:

$$\min_{\pi^{\text{pl}} \in \Pi} \max_{M \in \mathcal{M}^{\alpha}} \quad -H_{\rho_M^{\pi^{\text{pl}}}}\left(\pi^{\text{pl}}\right) \quad (4)$$

$$\text{subject to} \quad \rho_M^{\pi^{\text{pl}}}(s,s') = \rho_E(s,s'), \forall s,s' \in \mathcal{S}, \quad (5)$$

where our learner policy matches the expert's state transition occupancy measure ρ_E under the most adversarial MDP belonging to the set $\mathcal{M}^{\alpha}$. The corresponding dual problem is given by:

$$\max_{c \in \mathbb{R}^{S \times S}} \left(\min_{\pi^{\text{pl}} \in \Pi} \max_{M \in \mathcal{M}^{\alpha}} -H_{\rho_M^{\pi^{\text{pl}}}}\left(\pi^{\text{pl}}\right) + \mathbb{E}_{\rho_M^{\pi^{\text{pl}}}}\left[c(s,s')\right] \right) - \mathbb{E}_{\rho_E}\left[c(s,s')\right]. \quad (6)$$

In the dual problem, for any c, we attempt to learn a robust policy over the class $\mathcal{M}^{\alpha}$ with respect to the entropy regularized reward function. The parameter c plays the role of aligning the learner's policy with the expert's occupancy measure via constraint satisfaction.

For any given c, we need to solve the inner min-max problem of (6). However, during training, we only have access to the MDP M^{sim}. To this end, we utilize the equivalence between the robust MDP [13, 24] formulation and the action-robust MDP [26, 34] formulation shown in [34]. We can interpret the minimization over the environment class as the minimization over a set of opponent policies that with probability $1-\alpha$ take control of the agent and perform the worst possible move from the current agent state. We can write:

$$\begin{aligned} &\min_{\pi^{\text{pl}} \in \Pi} \max_{M \in \mathcal{M}^{\alpha}} -H_{\rho_M^{\pi^{\text{pl}}}}\left(\pi^{\text{pl}}\right) + \mathbb{E}_{\rho_M^{\pi^{\text{pl}}}}\left[c(s,s')\right] \\ &= \min_{\pi^{\text{pl}} \in \Pi} \max_{T^{\alpha} \in \mathcal{T}^{\alpha}} \mathbb{E}\left[G_c \mid \pi^{\text{pl}}, P_0, T^{\alpha}\right] \\ &= \min_{\pi^{\text{pl}} \in \Pi} \max_{\pi^{\text{op}} \in \Pi} \mathbb{E}\left[G_c \mid \alpha\pi^{\text{pl}} + (1-\alpha)\pi^{\text{op}}, M^{\text{sim}}\right], \end{aligned} \quad (7)$$

where $G_c := \sum_{t=0}^{\infty} \gamma^t \left\{ c(s_t, s_{t+1}) - H^{\pi^{pl}}(A|S = s_t) \right\}$. The above equality holds due to the derivation in section 3.1 of [34]. We can formulate the problem (7) as a two-player zero-sum Markov game [19] with transition dynamics given by

$$T^{\text{two},\alpha}(s'|s, a^{\text{pl}}, a^{\text{op}}) = \alpha T^{\text{sim}}(s'|s, a^{\text{pl}}) + (1-\alpha) T^{\text{sim}}(s'|s, a^{\text{op}}),$$

where a^{pl} is an action chosen according to the player policy and a^{op} according to the opponent policy. As a result, we reach a two-player Markov game with a regularization term for the player as follows:

$$\arg\min_{\pi^{\text{pl}} \in \Pi} \max_{\pi^{\text{op}} \in \Pi} \mathbb{E}\left[G_c \mid \pi^{\text{pl}}, \pi^{\text{op}}, M^{\text{two},\alpha}\right], \quad (8)$$

where $M^{\text{two},\alpha} = \left\{\mathcal{S}, \mathcal{A}, \mathcal{A}, T^{\text{two},\alpha}, \gamma, P_0\right\}$ is the two-player MDP associated with the above game.

Algorithm 1 Robust GAILfO

Input: state-only expert demonstrations $\mathcal{D}^E$, opponent strength parameter α.
Initialize: discriminator D_w, actor policy π_θ, and adversary policy π_ϕ.
for $n \in \{1, 2, \ldots, N\}$ **do**
  collect trajectories τ_i by executing the policies π_θ^{pl} and π_ϕ^{op} (see Algorithm 2), and store them in the demonstrations buffer $\mathcal{D}$.
  update the discriminator D_w to classify the expert demonstrations $\tau_E \in \mathcal{D}^E$ from the samples $\tau_i \in \mathcal{D}$, i.e., update w via gradient ascent with the following gradient:

$$\widehat{\mathbb{E}}_{\tau_i \in \mathcal{D}}[\nabla_w \log D_w(s,s')] + \widehat{\mathbb{E}}_{\tau_E \in \mathcal{D}^E}[\nabla_w \log(1 - D_w(s,s'))].$$

  update the reward function $R_w(s,s') \leftarrow -\log D_w(s,s')$.
  compute the following gradient estimates:

$$\widehat{\nabla}_\theta J(\theta,\phi) = \frac{1}{|\mathcal{D}|} \sum_{\tau_i \in \mathcal{D}} \sum_t \gamma^t \nabla_\theta \log \pi^{\text{mix}}_{\theta,\phi}(a_t^i|s_t^i) \left[G_t^i + \lambda G_t^{\log,i} \right]$$

$$\widehat{\nabla}_\phi J(\theta,\phi) = \frac{1}{|\mathcal{D}|} \sum_{\tau_i \in \mathcal{D}} \sum_t \gamma^t \nabla_\phi \log \pi^{\text{mix}}_{\theta,\phi}(a_t^i|s_t^i) \left[G_t^i + \lambda G_t^{\log,i} \right],$$

  where $G_t^i = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_w(s_k^i, s_{k+1}^i)$ and $G_t^{\log,i} = \sum_{k=t+1}^{T} -\gamma^{k-t-1} H^{\pi_\theta^{\text{pl}}}(A|S = s_k^i)$
  update the policies π_θ^{pl} and π_ϕ^{op} using PPO with the gradient estimates above.
end for

Algorithm 2 Collecting Trajectories

Input: total number of trajectories N_{traj}, reward function R_w.
for $n \in \{1, 2, \ldots, N_{\text{traj}}\}$ **do**
  $t \leftarrow 0$
  initialize an empty trajectory τ.
  while not done **do**
    observe state s_t.
    sample actions $a_t^{\text{pl}} \sim \pi_\theta^{\text{pl}}(\cdot|s_t)$, $a_t^{\text{op}} = \pi_\phi^{\text{op}}(\cdot|s_t)$.
    execute a_t^{op} with probability $\bar{\alpha}$, or a_t^{pl} with probability α.
    observe $r_{t+1} = R_w(s_t, s_{t+1})$, next state s_{t+1}, and done.
    store the tuple $(s_t, a_t^{\text{pl}}, a_t^{\text{op}}, s_{t+1}, r_{t+1})$ in the trajectory τ.
  end while
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau\}$.
end for
Output: $\mathcal{D}$

4.2 Robust GAILfO

In this section, we present our robust Generative Adversarial Imitation Learning from Observation (robust GAILfO) algorithm based on the discussions in Section 4.1. We begin with the robust variant of the two-step procedure $\mathrm{RL} \circ \mathrm{IRL}_{\psi}\left(\rho_E\right)$ of the MCE-IRL algorithm:

$$\begin{aligned}\mathrm{IRL}_{\psi}\left(\rho_E\right) = & \arg\max_{c \in \mathbb{R}^{\mathcal{S}\times\mathcal{S}}} -\psi\left(c\right) - \mathbb{E}_{\rho_E}\left[c\left(s, s'\right)\right] \\ & + \min_{\pi^{\mathrm{pl}} \in \Pi} \max_{\pi^{\mathrm{op}} \in \Pi} -\lambda H_{\rho^{\pi^{\mathrm{mix}}}_{M^{\mathrm{sim}}}}\left(\pi^{\mathrm{pl}}\right) + \mathbb{E}_{\rho^{\pi^{\mathrm{mix}}}_{M^{\mathrm{sim}}}}\left[c\left(s, s'\right)\right] \\ \mathrm{RL}\left(c\right) = & \arg\min_{\pi^{\mathrm{pl}} \in \Pi} \max_{\pi^{\mathrm{op}} \in \Pi} -\lambda H_{\rho^{\pi^{\mathrm{mix}}}_{M^{\mathrm{sim}}}}\left(\pi^{\mathrm{pl}}\right) + \mathbb{E}_{\rho^{\pi^{\mathrm{mix}}}_{M^{\mathrm{sim}}}}\left[c\left(s, s'\right)\right],\end{aligned}$$

where $\pi^{\mathrm{mix}} = \alpha\pi^{\mathrm{pl}} + (1-\alpha)\pi^{\mathrm{op}}$. Then, similar to [12, 36], the above two step procedure can be reduced to the following optimization problem using the discriminator $D : \mathcal{S} \times \mathcal{S} \to (0, 1)$:

$$\begin{aligned}\min_{\pi^{\mathrm{pl}} \in \Pi} \max_{\pi^{\mathrm{op}} \in \Pi} \max_{D \in (0,1)^{\mathcal{S}\times\mathcal{S}}} & -\lambda H_{\rho^{\pi^{\mathrm{mix}}}_{M^{\mathrm{sim}}}}\left(\pi^{\mathrm{pl}}\right) + \mathbb{E}_{\rho^{\pi^{\mathrm{mix}}}_{M^{\mathrm{sim}}}}\left[\log D\left(s, s'\right)\right] \\ & + \mathbb{E}_{\rho_E}\left[\log\left(1 - D\left(s, s'\right)\right)\right].\end{aligned}$$

We parameterize the policies and the discriminator as $\pi^{\mathrm{pl}}_{\theta}$, π^{op}_{ϕ}, and D_w (with parameters θ, ϕ, and w), and rewrite the above problem as follows:

$$\begin{aligned}\min_{\theta} \max_{\phi} \max_{w} & -\lambda H_{\rho^{\pi^{\mathrm{mix}}_{\theta,\phi}}_{M^{\mathrm{sim}}}}\left(\pi^{\mathrm{pl}}_{\theta}\right) + \mathbb{E}_{\rho^{\pi^{\mathrm{mix}}_{\theta,\phi}}_{M^{\mathrm{sim}}}}\left[\log D_w\left(s, s'\right)\right] \\ & + \mathbb{E}_{\rho_E}\left[\log\left(1 - D_w\left(s, s'\right)\right)\right],\end{aligned}$$

where $\pi^{\mathrm{mix}}_{\theta,\phi} = \alpha\pi^{\mathrm{pl}}_{\theta} + (1-\alpha)\pi^{\mathrm{op}}_{\phi}$. We solve the above problem by taking gradient steps alternatively w.r.t. θ, ϕ, and w. The calculation for the gradient estimates are given in Appendix B. Following [12, 36], we use the proximal policy optimization (PPO) [31]) to update the policies parameters. Our complete algorithm is given in Algorithm 1.

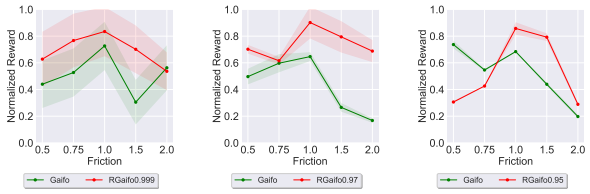
We also note that one could use any robust RL approach (including domain randomization) to solve the inner min-max problem of (6). In our work, we used the action-robustness approach since: (i) in the robust RL literature, the equivalence between the domain randomization approach and the action-robustness approach is already established [34], and (ii) compared to the domain randomization approach, the action-robustness approach only requires access to a single simulation environment and creates a range of environments via action perturbations.

5 EXPERIMENTS

We compare the performance of our robust GAILfO algorithm with different values of $\alpha \in \{1.0, 0.999, 0.99, 0.98, 0.97, 0.96, 0.95, 0.90\}$ against the standard GAILfO algorithm proposed in [36]. To the best of our knowledge, GAILfO is the only large-scale imitation learning method that is applicable under the setting described in Section 1 (see Table 1).

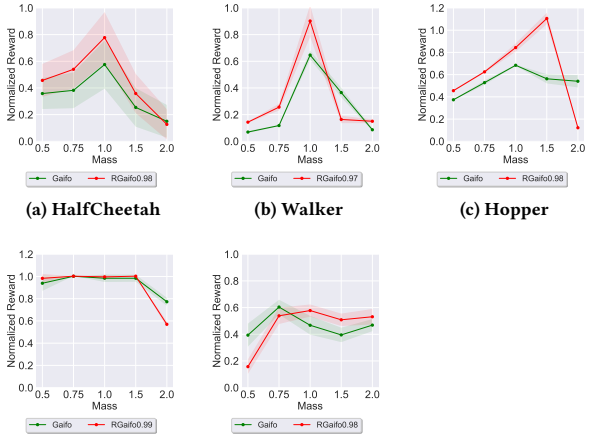
5.1 Continuous Control Tasks on MuJoCo

In this section, we evaluate the performance of our method on standard continuous control benchmarks available on OpenAI Gym [2] utilizing the MuJoCo environment [35]. Specifically, we benchmark on five tasks: Half-Cheetah, Walker, Hopper, Swimmer, and



(a) HalfCheetah (b) Walker (c) Hopper

Figure 1: The average (over 3 seeds) transfer performance of Algorithm 1 with different values of α for each MuJoCo task as reported in the legend of each plot. The x-axis denotes the relative friction of the learner environment M^{sim}. The policies are evaluated in $M^{\mathrm{real}}_{c^*}$ over $1e5$ steps truncating the last episode if it does not terminate.



(a) HalfCheetah (b) Walker (c) Hopper

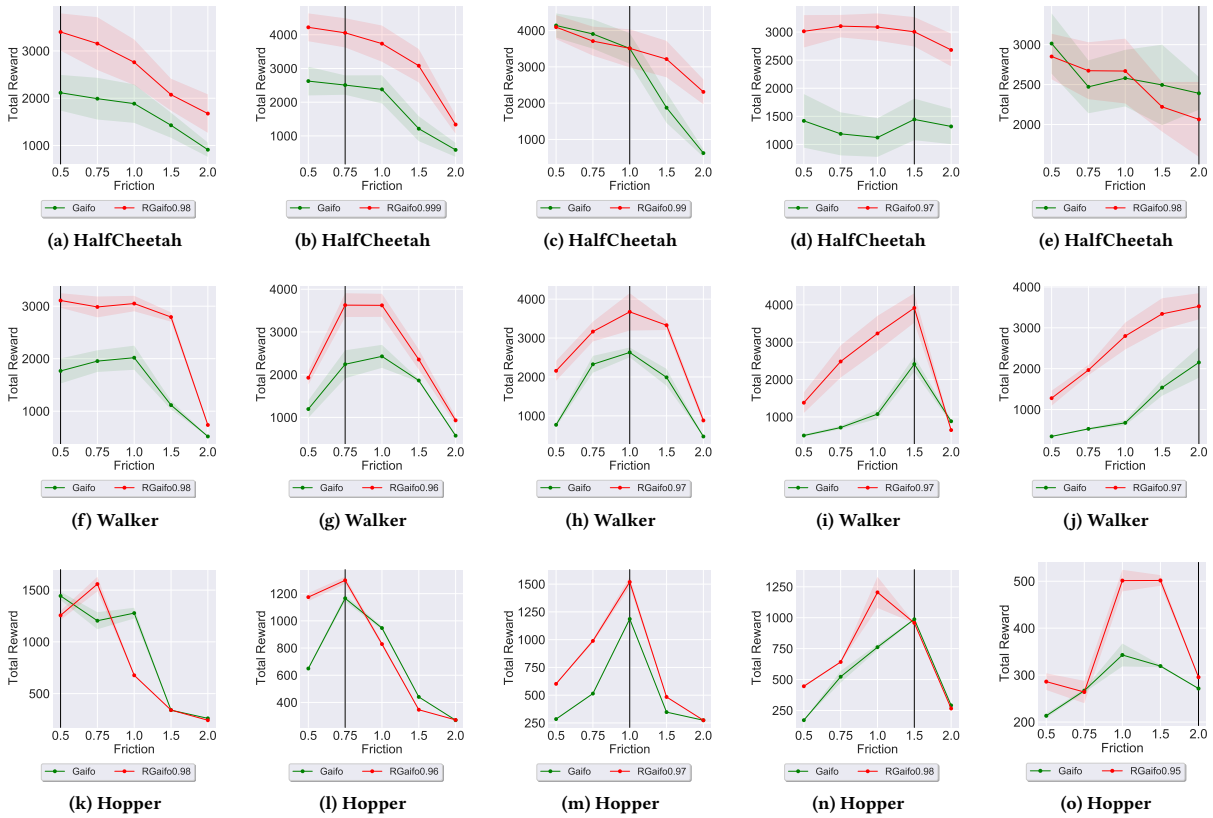
(d) InvDoublePend (e) Swimmer

Figure 2: The average (over 3 seeds) transfer performance of Algorithm 1 with different values of α for each MuJoCo task as reported in the legend of each plot. The x-axis denotes the relative mass of the learner environment M^{sim}. The policies are evaluated in $M^{\mathrm{real}}_{c^*}$ over $1e5$ steps truncating the last episode if it does not terminate.

Inverted-Double-Pendulum. Details of these environments can be found in [2] and on the GitHub website.

The default configurations of the MuJoCo environment (provided in OpenAI Gym) is regarded as the real or deployment environment (M^{real}), and the expert demonstrations are collected there. We do not assume any access to the expert MDP beyond this during the training phase. We construct the simulation or training environments (M^{sim}) for the imitator by modifying some parameters independently: (i) the mass of the bot in M^{sim} is $\{0.5, 0.75, 1.0, 1.5, 2.0\} \times$ the mass in M^{real}, and (ii) the friction coefficient on all the joints of the bot in M^{sim} is $\{0.5, 0.75, 1.0, 1.5, 2.0\} \times$ the coefficient in M^{real}.

We train an agent on each task by proximal policy optimization (PPO) algorithm [31] using the rewards defined in the OpenAI Gym [2]. We use the resulting stochastic policy as the expert policy



(a) HalfCheetah (b) HalfCheetah (c) HalfCheetah (d) HalfCheetah (e) HalfCheetah

(f) Walker (g) Walker (h) Walker (i) Walker (j) Walker

(k) Hopper (l) Hopper (m) Hopper (n) Hopper (o) Hopper

Figure 3: The average (over 3 seeds) robust performance of Algorithm 1 with different values of α for each MuJoCo task as reported in the legend of each plot. The expert environment M^{real}, in which the demonstrations are collected, has relative friction 1.0. In each plot, the black vertical line corresponds to the relative friction of the learner environment M^{sim} where we trained the policy with Algorithm 1. The x-axis denotes the relative friction of the test environment M^{test} in which the policies are evaluated. The policies are evaluated over $1e5$ steps truncating the last episode if it does not terminate.

π^E. In all our experiments, 10 state-only expert demonstrations collected by the expert policy π^E in the real environment M^{real} is given to the learner.

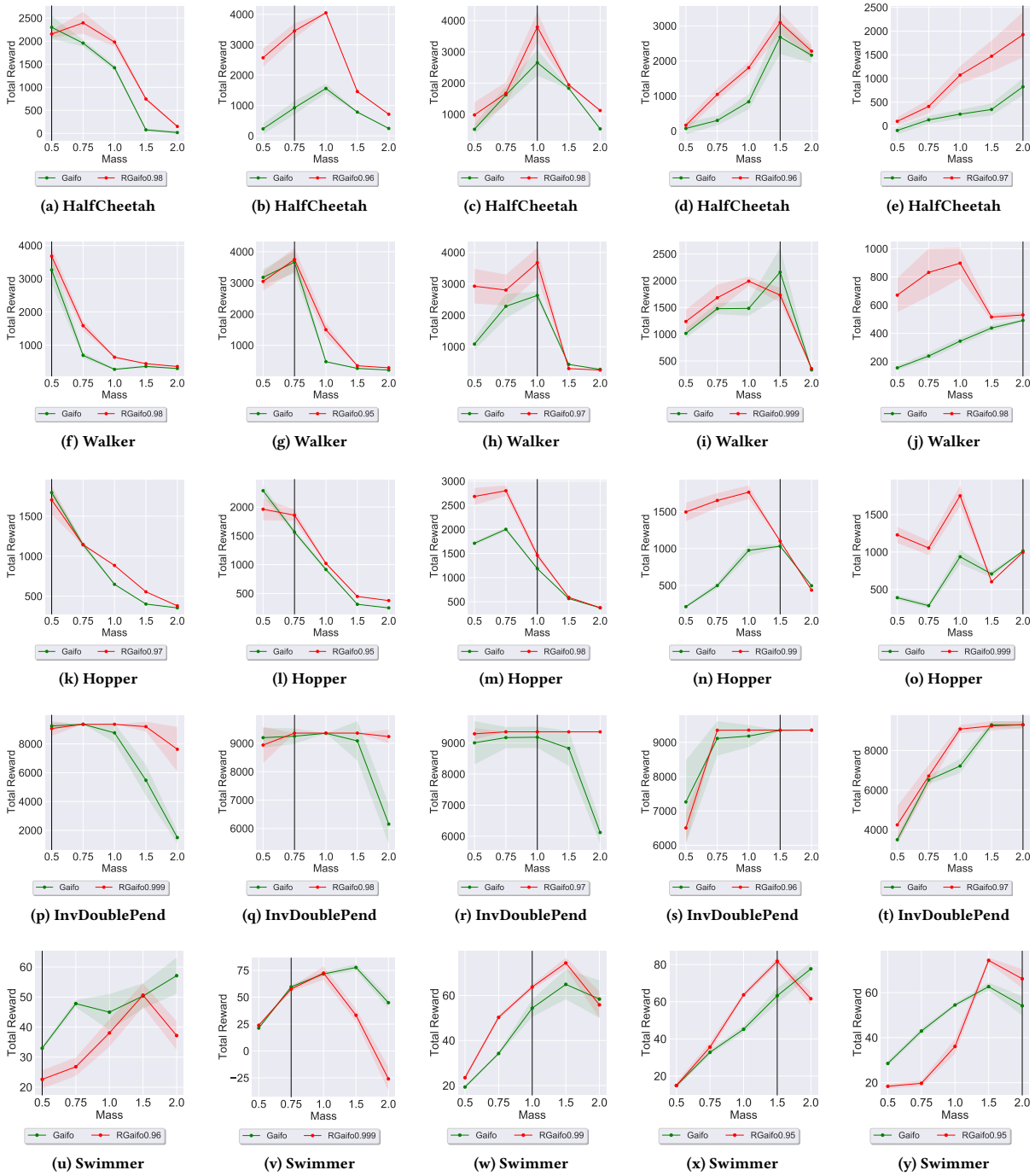
Our Algorithm 1 implementation is based on the codebase from https://github.com/Khrylx/PyTorch-RL. We use a two-layer feed-forward neural network structure of (128, 128, tanh) for both actors (agent and adversary) and discriminator. The actor or policy networks are trained by the proximal policy optimization (PPO) method. For training the discriminator D, we use Adam [16] with a learning rate of $1e-4$. For each environment-mismatch pair, we identified the best performing α parameter based on the ablation study reported in Appendix C. The learner is trained in the simulator M^{sim} for ≈3M time steps. We run our experiments, for each environment, with 3 different seeds. We report the mean and standard error of the performance (cumulative true rewards) over 3 trials. The cumulative reward is normalized with ones earned by π^E and a random policy so that 1.0 and 0.0 indicate the performance of π^E and the random policy, respectively.

Figures 1, and 2 plot the performance of the policy evaluated on the deployment environment (M^{real}). The x-axis corresponds to the simulation environment (M^{sim}) on which the policy is trained on. We observe that our robust GAILfO produces policies that can be successfully transferred to the M^{real} environment from M^{sim} compared to the standard GAILfO.

Finally, we evaluate the robustness of the policies trained by our algorithm (with different dynamics mismatch) under different testing conditions. At test time, we evaluate the learned policies by changing the mass and friction values and estimating the cumulative rewards. As shown in Figures 3 and 4, our Algorithm 1 outperforms the baseline in terms of robustness as well.

5.2 Continuous Gridworld Tasks under Additive Transition Dynamics Mismatch

In this section, we evaluate the effectiveness of our method on a continuous gridworld environment under a transition dynamics mismatch induced by additive noise. Specifically, we consider a 2D environment, where we denote the horizontal coordinate as $x \in [0, 1]$ and vertical one as $y \in [0, 1]$. The agent starts in the upper left corner, i.e., the coordinate $(0, 1)$, and the episode ends when the agent reaches the lower right region defined by the indicator



(a) HalfCheetah (b) HalfCheetah (c) HalfCheetah (d) HalfCheetah (e) HalfCheetah

(f) Walker (g) Walker (h) Walker (i) Walker (j) Walker

(k) Hopper (l) Hopper (m) Hopper (n) Hopper (o) Hopper

(p) InvDoublePend (q) InvDoublePend (r) InvDoublePend (s) InvDoublePend (t) InvDoublePend

(u) Swimmer (v) Swimmer (w) Swimmer (x) Swimmer (y) Swimmer

Figure 4: The average (over 3 seeds) robust performance of Algorithm 1 with different values of α for each MuJoCo task as reported in the legend of each plot. The expert environment M^{real}, in which the demonstrations are collected, has relative mass 1.0. In each plot, the black vertical line corresponds to the relative mass of the learner environment M^{sim} where we trained the policy with Algorithm 1. The x-axis denotes the relative mass of the test environment M^{test} in which the policies are evaluated. The policies are evaluated over $1e5$ steps truncating the last episode if it does not terminate.

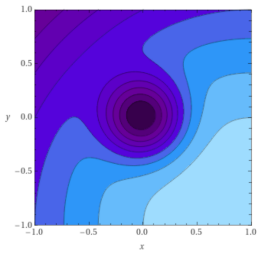
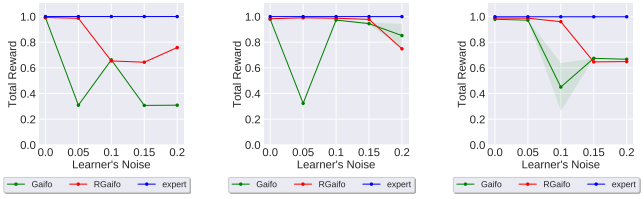


Figure 5: The contour curves for the reward function of the 2D gridworld environment.

function $\mathbf{1}\{x \in [0.95, 1], y \in [-1, -0.95]\}$. The reward function is given by: $R(x, y) = -(x-1)^2 - (y+1)^2 - 80e^{-8(x^2+y^2)} + 10 \cdot \mathbf{1}\{x \in [0.95, 1], y \in [-1, -0.95]\}$. Figure 5 provides a graphical representation of the reward function. Note that the central region of the 2D environment represents a low reward area that should be avoided. The action space for the agent is given by $\mathcal{A} = [-0.5, 0.5]^2$, and the transition dynamics are given by: $s_{t+1} = s_t + \frac{a_t}{10}$ with probability (w.p.) $1 - \epsilon$, and $s_{t+1} = s_t - \frac{s_t}{10\|s_t\|_2}$ w.p. ϵ. Thus, with probability ϵ, the environment does not respond to the action taken by the agent, but it takes a step towards the low reward area centered at the origin, i.e., $-\frac{s_t}{10\|s_t\|_2}$. The agent should therefore pass far enough from the origin. The parameter ϵ can be varied to create a dynamic mismatch, e.g., higher ϵ corresponds to a more difficult environment.

We use three experts trained with $\epsilon = 0.0, \epsilon = 0.05,$ and $\epsilon = 0.1$. The learners act in a different environment with the following values for ϵ: 0.0, 0.05, 0.1, 0.15, 0.2. Figure 6 plots the performance of the trained learner policy evaluated on the expert environment. The x-axis corresponds to the learner environment on which the learner policy is trained. In general, we observe a behavior comparable to the MuJoCo experiments. We can often find an appropriate value for α such that Robust GAILfO learns to imitate under mismatch largely better than standard GAILfO.



(a) Expert $\epsilon = 0.0$ **(b) Expert $\epsilon = 0.05$** **(c) Expert $\epsilon = 0.1$**

Figure 6: Average performance (over 3 seeds) of Algorithm 1 with different values of α for each mismatch (i.e., each point on the x-axis) in the environment shown in Figure 5. The α values are chosen based on the ablation study in Figure 14 (see Appendix E). The x-axis denotes the ϵ value of the learner environment. The policies are evaluated over $1e5$ steps truncating the last episode if it does not terminate. In Appendix F, we verify that our strategy of choosing appropriate α value does not introduce maximization bias.

5.3 Choice of α

We note that one has to carefully choose the value of α to avoid too conservative behavior (see Figure 7 in Appendix C). In principle, given a rough estimate $\widehat{T}^E$ of the expert dynamics T^E, one could choose this value based on Eq. (3). However, the choice of suitable α value is also affected by the other design choices of the algorithm, e.g., how many iterations the player and adversary are updated in the inner loop, and function approximators used.

In order to estimate the accuracy of the simulator, we can execute a safe baseline policy in both the simulator and the real environment, collect trajectories or datasets, and compute an estimate of the transition-dynamics distance between them. We can also utilize the performance difference lemma from [6] to obtain a lower bound on the transition dynamics mismatch based on the value function difference in the two environments.

Apart from the final evaluation, we also minimally access (in our experiments) the deployment environment for choosing the appropriate value for α. Compared to training a policy in the deployment environment from scratch, accessing the deployment environment to choose α is sample-efficient. We only need to evaluate the final policies (trained in the simulation environment) once for each value of α. When we already have a reasonable estimate of α, we can also reduce these evaluations.

6 CONCLUSIONS

In this work, we propose a robust LfO method to solve an offline imitation-learning problem, in which a few state-only expert demonstrations and a simulator with misspecified dynamics are given to the learner. Even though our Algorithm 1 is not essentially different from the standard robust RL methods, the robust optimization problem formulation to derive our algorithm is important and novel in the IL context. Experiment results in continuous control tasks on MuJoCo show that our method clearly outperforms the standard GAILfO in terms of the transfer performance (with model misspecification) in the real environment, as well as the robust performance under varying testing conditions.

Our algorithm falls under the category of zero-shot sim-to-real transfer [39] with expert demonstrations, making our method well suited for robotics applications. In principle, one can easily incorporate the two-player Markov game idea into any imitation learning algorithm and derive its robust version. This work can be considered a direction towards improving the sample efficiency of IL algorithms in terms of the number of environment interactions through robust training on a misspecified simulator.

ACKNOWLEDGMENTS

Luca Viano has received financial support from the Enterprise for Society Center (E4S). Parameswaran Kamalaruban acknowledges support from The Alan Turing Institute. Craig Innes and Subramanian Ramamoorthy are supported by a grant from the UKRI Strategic Priorities Fund to the UKRI Research Node on Trustworthy Autonomous Systems Governance and Regulation (EP/V026607/1, 2020-2024). Adrian Weller acknowledges support from a Turing AI Fellowship under grant EP/V025379/1, EPSRC grant EP/V056522/1, The Alan Turing Institute, and the Leverhulme Trust via CFI.

REFERENCES

[1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[3] Siddharth Desai, Ishan Durugkar, Haresh Karnan, Garrett Warnell, Josiah Hanna, and Peter Stone. 2020. An Imitation from Observation Approach to Transfer Learning with Dynamics Mismatch. In *Advances in Neural Information Processing Systems*.

[4] John C Doyle, Bruce A Francis, and Allen R Tannenbaum. 2013. *Feedback control theory*. Courier Corporation.

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

[6] Eyal Even-Dar and Yishay Mansour. 2003. Approximate equivalence of Markov decision processes. In *Learning Theory and Kernel Machines*. Springer, 581–594.

[7] Tom Everitt and Marcus Hutter. 2016. Avoiding wireheading with value reinforcement learning. In *International Conference on Artificial General Intelligence*.

[8] Justin Fu, Katie Luo, and Sergey Levine. 2018. Learning Robust Rewards with Adversarial Inverse Reinforcement Learning. In *International Conference on Learning Representations*.

[9] Tanmay Gangwani and Jian Peng. 2020. State-only Imitation with Transition Dynamics Mismatch. In *International Conference on Learning Representations*.

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

[11] Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Yu-Wei Chao, Qian Wan, Stan Birchfield, Nathan Ratliff, and Dieter Fox. 2020. DexPilot: Vision-Based Teleoperation of Dexterous Robotic Hand-Arm System. In *IEEE International Conference on Robotics and Automation*.

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

[13] Garud N Iyengar. 2005. Robust dynamic programming. *Mathematics of Operations Research* (2005).

[14] Shengyi Jiang, Jingcheng Pang, and Yang Yu. 2020. Offline Imitation Learning with a Misspecified Simulator. In *Advances in Neural Information Processing Systems*.

[15] Parameswaran Kamalaruban, Yu-Ting Huang, Ya-Ping Hsieh, Paul Rolland, Cheng Shi, and Volkan Cevher. 2020. Robust reinforcement learning via adversarial training with langevin dynamics. In *Advances in Neural Information Processing Systems*.

[16] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

[17] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* (2016).

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

[19] Michael L Littman. 1994. Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning*.

[20] Fangchen Liu, Zhan Ling, Tongzhou Mu, and Hao Su. 2020. State Alignment-based Imitation Learning. In *International Conference on Learning Representations*.

[21] Daniel J Mankowitz, Nir Levine, Rae Jeong, Abbas Abdolmaleki, Jost Tobias Springenberg, Yuanyuan Shi, Jackie Kay, Todd Hester, Timothy Mann, and Martin Riedmiller. 2020. Robust Reinforcement Learning for Continuous Control with Model Misspecification. In *International Conference on Learning Representations*.

[22] Jun Morimoto and Kenji Doya. 2005. Robust reinforcement learning. *Neural Computation* (2005).

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

[24] Arnab Nilim and Laurent El Ghaoui. 2005. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research* (2005).

[25] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE International Conference on Robotics and Automation*.

[26] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. 2017. Robust Adversarial Reinforcement Learning. In *International Conference on Machine Learning*.

[27] Ilija Radosavovic, Xiaolong Wang, Lerrel Pinto, and Jitendra Malik. 2020. State-only imitation learning for dexterous manipulation. *arXiv preprint arXiv:2004.04650* (2020).

[28] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. 2017. EPOpt: Learning Robust Neural Network Policies Using Model Ensembles. In *International Conference on Learning Representations*.

[29] Stefan Schaal. 1999. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences* (1999).

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

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[32] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*.

[33] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation.. In *Advances in Neural Information Processing Systems*.

[34] Chen Tessler, Yonathan Efroni, and Shie Mannor. 2019. Action Robust Reinforcement Learning and Applications in Continuous Control. In *International Conference on Machine Learning*.

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

[36] Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158* (2018).

[37] Luca Viano, Yu-Ting Huang, Parameswaran Kamalaruban, Adrian Weller, and Volkan Cevher. 2021. Robust Inverse Reinforcement Learning under Transition Dynamics Mismatch. In *Advances in Neural Information Processing Systems*.

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

[39] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. 2020. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. In *IEEE Symposium Series on Computational Intelligence*.

[40] Brian D Ziebart. 2010. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. (2010).

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

CODE REPOSITORY

https://github.com/lviano/robust_gaifo

A DETAILS ON THE EQUIVALENCE BETWEEN ACTION ROBUST MDP AND ROBUST MDP

In the following we prove the last equality of Eq. (7).

Theorem 1. *Given the set*

$$\mathcal{T}^{\alpha} := \left\{ T : T(s'|s,a) = \alpha T^{\text{sim}}(s'|s,a) + (1-\alpha)\bar{T}(s'|s),\ \bar{T}(s'|s) = \sum_a \pi(a|s) T^{\text{sim}}(s'|s,a), \quad \forall \pi \in \Pi \right\}$$

and a cost function depending only on states, i.e. $r : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$, define $G_c = \sum_{t=0}^{\infty} \gamma^t r(s_t, s_{t+1})$. Then, the following holds:

$$\min_{\pi^{\text{pl}} \in \Pi} \max_{T^{\alpha} \in \mathcal{T}^{\alpha}} \mathbb{E}\left[G_c \mid \pi^{\text{pl}}, P_0, T^{\alpha} \right] = \min_{\pi^{\text{pl}} \in \Pi} \max_{\pi^{\text{op}} \in \Pi} \mathbb{E}\left[G_c \mid \alpha\pi^{\text{pl}} + (1-\alpha)\pi^{\text{op}}, M^{\text{sim}} \right].$$

In particular, the result in Eq. (7) follows from the choice: $r(s_t, s_{t+1}) = c(s_t, s_{t+1}) + H^{\pi^{\text{pl}}}(A|S = s_t)$.

Proof. Let us define $P^{\pi,T}(s_0, \ldots, s_N) := P_0(s_0) \prod_{t=0}^{N-1} \sum_a \pi(a|s_t) T(s_{t+1}|s_t, a)$. We need to show equality between the distributions $P^{\pi^{\text{pl}}, \alpha T^{\text{sim}} + (1-\alpha)\bar{T}}$ and $P^{\alpha\pi^{\text{pl}} + (1-\alpha)\pi^{\text{op}}, T^{\text{sim}}}$. Due to the Markov property, this is equivalent to show:

$$\sum_a \pi^{\text{pl}}(a|s_t) \left[\alpha T^{\text{sim}}(s_{t+1}|s_t, a) + (1-\alpha)\bar{T}(s_{t+1}|s_t) \right] = \sum_a \left[\alpha\pi^{\text{pl}}(a|s_t) + (1-\alpha)\pi^{\text{op}}(a|s_t) \right] T^{\text{sim}}(s_{t+1}|s_t, a), \tag{9}$$

that implies:

$$\underbrace{\sum_a \pi^{\text{pl}}(a|s_t)}_{=1} \bar{T}(s_{t+1}|s_t) = \sum_a \pi^{\text{op}}(a|s_t) T^{\text{sim}}(s_{t+1}|s_t, a).$$

Hence, it follows that equality between $P^{\pi^{\text{pl}}, \alpha T^{\text{sim}} + (1-\alpha)\bar{T}}$ and $P^{\alpha\pi^{\text{pl}} + (1-\alpha)\pi^{\text{op}}, T^{\text{sim}}}$ holds for:

$$\bar{T}(s_{t+1}|s_t) = \sum_a \pi^{\text{op}}(a|s_t) T^{\text{sim}}(s_{t+1}|s_t, a),$$

as we used in the definition of the set $\mathcal{T}^{\alpha}$. □

B ADDITIONAL DETAILS ON ALGORITHM 1

By interpreting $R_w(s, s') = -\log D_w(s, s')$ as the reward function, we have (for a fixed w):

$$J(\theta, \phi) := \mathbb{E}_{\substack{\pi^{\text{mix}}_{\theta,\phi} \\ \rho_{M^{\text{sim}}}}} \left[R_w(s, s') \right] + \lambda H_{\substack{\pi^{\text{mix}}_{\theta,\phi} \\ \rho_{M^{\text{sim}}}}} \left(\pi^{\text{pl}}_{\theta} \right) = J_1(\theta, \phi) + J_2(\theta, \phi),$$

where

$$J_1(\theta, \phi) := \mathbb{E}\left[\sum_t \gamma^t R_w(s_t, s_{t+1}) \middle| \pi^{\text{mix}}_{\theta,\phi}, M^{\text{sim}} \right]$$

$$J_2(\theta, \phi) := \lambda \mathbb{E}\left[\sum_t \gamma^t H^{\pi^{pl}_{\theta}}(A|S = s_t) \middle| \pi^{\text{mix}}_{\theta,\phi}, M^{\text{sim}} \right].$$

By the policy gradient theorem, the derivatives of the first term w.r.t the player and the opponent policy parameters are given by:

$$\nabla_{\theta} J_1(\theta, \phi) = \sum_{s \in \mathcal{S}} \sum_t \gamma^t \mathbb{P}\left[S_t = s \mid \pi^{\text{mix}}_{\theta,\phi}, M^{\text{sim}} \right] \sum_a \nabla_{\theta} \pi^{\text{mix}}_{\theta,\phi}(a|s) Q_{\pi^{\text{mix}}_{\theta,\phi}}(s, a)$$

$$\nabla_{\phi} J_1(\theta, \phi) = \sum_{s \in \mathcal{S}} \sum_t \gamma^t \mathbb{P}\left[S_t = s \mid \pi^{\text{mix}}_{\theta,\phi}, M^{\text{sim}} \right] \sum_a \nabla_{\phi} \pi^{\text{mix}}_{\theta,\phi}(a|s) Q_{\pi^{\text{mix}}_{\theta,\phi}}(s, a),$$

where

$$Q_{\pi^{\text{mix}}_{\theta,\phi}}(s, a) = \sum_{s'} T^{\text{sim}}(s' \mid a, s) \left(R_w(s, s') + \gamma V_{\pi^{\text{mix}}_{\theta,\phi}}(s') \right)$$

$$V_{\pi^{\text{mix}}_{\theta,\phi}}(s) = \mathbb{E}\left[\sum_t \gamma^t R_w(s_t, s_{t+1}) \middle| \pi^{\text{mix}}_{\theta,\phi}, M^{\text{sim}}, s_0 = s \right].$$

For the second term, we introduce the following quantities:

$$Q^{\log}_{\pi^{\text{mix}}_{\theta,\phi}}(s,a) = \sum_{s'} T^{\text{sim}}(s' \mid s,a)\left(\lambda H^{\pi^{pl}_{\theta}}(A|S=s_t) + \gamma V^{\log}_{\pi^{\text{mix}}_{\theta,\phi}}(s')\right)$$

$$V^{\log}_{\pi^{\text{mix}}_{\theta,\phi}}(s) = \mathbb{E}\left[\sum_t \lambda\gamma^t H^{\pi^{pl}_{\theta}}(A|S=s_t)\middle|\pi^{\text{mix}}_{\theta,\phi}, M^{\text{sim}}, s_0 = s\right]$$

Then, we obtain the following derivatives of the second term:

$$\nabla_\theta J_2(\theta,\phi) = \sum_{s\in\mathcal{S}}\sum_t \gamma^t \mathbb{P}\left[S_t = s \mid \pi^{\text{mix}}_{\theta,\phi}, M^{\text{sim}}\right]\sum_a \nabla_\theta \pi^{\text{mix}}_{\theta,\phi}(a|s) Q^{\log}_{\pi^{\text{mix}}_{\theta,\phi}}(s,a)$$

$$\nabla_\phi J_2(\theta,\phi) = \sum_{s\in\mathcal{S}}\sum_t \gamma^t \mathbb{P}\left[S_t = s \mid \pi^{\text{mix}}_{\theta,\phi}, M^{\text{sim}}\right]\sum_a \nabla_\phi \pi^{\text{mix}}_{\theta,\phi}(a|s) Q^{\log}_{\pi^{\text{mix}}_{\theta,\phi}}(s,a).$$

For a practical algorithm, we need to compute gradient estimates from a data-set of sampled trajectories $\mathcal{D} = \{\tau_i\}_i$ with $\tau_i = (s^i_0, a^i_0, \ldots, s^i_T, a^i_T)$. The gradient estimates are given by:

$$\widehat{\nabla}_\theta J_1(\theta,\phi) = \sum_{\tau_i\in\mathcal{D}}\sum_t \gamma^t \nabla_\theta \log \pi^{\text{mix}}_{\theta,\phi}(a^i_t|s^i_t)\widehat{Q}_{\pi^{\text{mix}}_{\theta,\phi}}(s^i_t,a^i_t)$$

$$\widehat{\nabla}_\phi J_1(\theta,\phi) = \sum_{\tau_i\in\mathcal{D}}\sum_t \gamma^t \nabla_\phi \log \pi^{\text{mix}}_{\theta,\phi}(a^i_t|s^i_t)\widehat{Q}_{\pi^{\text{mix}}_{\theta,\phi}}(s^i_t,a^i_t)$$

$$\widehat{\nabla}_\theta J_2(\theta,\phi) = \sum_{\tau_i\in\mathcal{D}}\sum_t \gamma^t \nabla_\theta \log \pi^{\text{mix}}_{\theta,\phi}(a^i_t|s^i_t)\widehat{Q}^{\log}_{\pi^{\text{mix}}_{\theta,\phi}}(s^i_t,a^i_t)$$

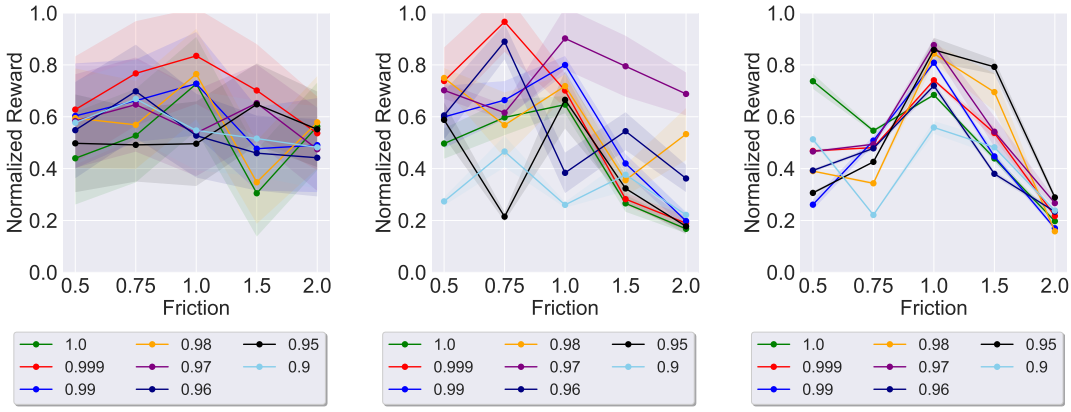
$$\widehat{\nabla}_\phi J_2(\theta,\phi) = \sum_{\tau_i\in\mathcal{D}}\sum_t \gamma^t \nabla_\phi \log \pi^{\text{mix}}_{\theta,\phi}(a^i_t|s^i_t)\widehat{Q}^{\log}_{\pi^{\text{mix}}_{\theta,\phi}}(s^i_t,a^i_t),$$

where the estimator $\widehat{Q}_{\pi^{\text{mix}}_{\theta,\phi}}(s^i_t,a^i_t)$ is the future return observed for the trajectory i after time t, i.e., $\widehat{Q}_{\pi^{\text{mix}}_{\theta,\phi}}(s^i_t,a^i_t) = \sum_{k=t+1}^{T}\gamma^{k-t-1}R_w(s^i_k, s^i_{k+1}) = G^i_t$. Similarly, for the entropy term we have $\widehat{Q}^{\log}_{\pi^{\text{mix}}_{\theta,\phi}}(s^i_t,a^i_t) = \sum_{k=t+1}^{T} -\gamma^{k-t-1}H^{\pi^{\text{pl}}_{\theta}}(A|S=s^i_k) = G^{\log,i}_t$. The trajectory sampling process is given in Algorithm 2.

C TRANSFER PERFORMANCE: MUJOCO

We present the following results:

- The ablation study on the transfer performance of Algorithm 1 with different values of α under the relative friction mismatches (see Figure 7).
- The ablation study on the transfer performance of Algorithm 1 with different values of α under the relative mass mismatches (see Figure 8).
- The transfer performance of Algorithm 1 with different (best) values of α for each relative friction mismatch of a task (see Figure 9).
- The transfer performance of Algorithm 1 with different (best) values of α for each relative mass mismatch of a task (see Figure 10).



(a) HalfCheetah (b) Walker (c) Hopper

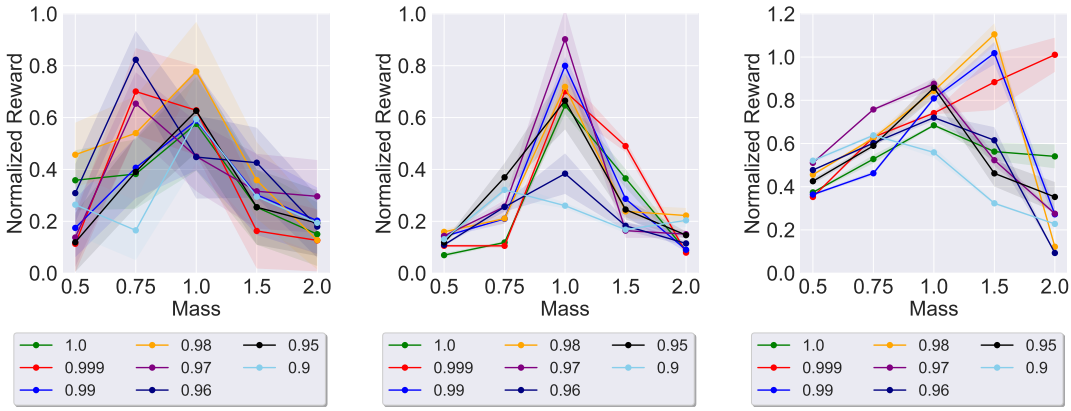
Figure 7: The average (over 3 seeds) transfer performance of Algorithm 1 with different values of α. The ablation shown here is used to choose α in Figure 1. The x-axis denotes the relative friction of the learner environment M^{sim}. The policies are evaluated in $M^{\text{real}}_{c^*}$ over $1e5$ steps truncating the last episode if it does not terminate. Note that robust-GAILfO with $\alpha = 1$ corresponds to GAILfO.

Table 2: Best value for $\alpha < 1$ chosen independently for each mismatch based on the ablation in Figure 7. The performance of this configuration is reported by the red line in Figure 9. We add a 1 in brackets when standard GAILfO outperforms the robust version. The value outside brackets denotes the best value found for the robust version.

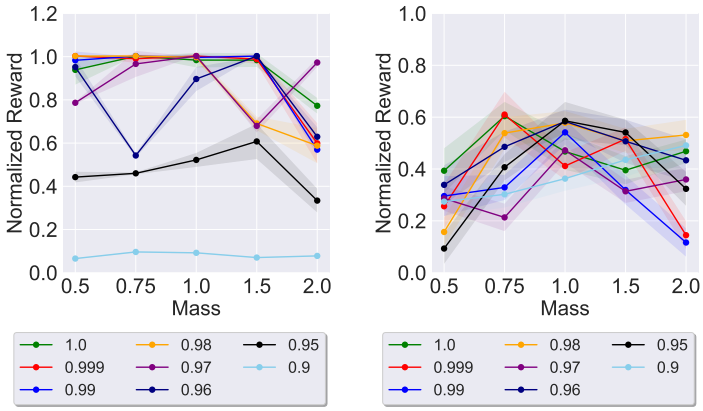
	Relative Friction				
	0.5	0.75	1.0	1.5	2.0
HalfCheetah	0.999	0.999	0.999	0.999	0.999
Walker	0.98	0.999	0.97	0.97	0.97
Hopper	0.9 (1)	0.99 (1)	0.97	0.95	0.95

Table 3: Best value for $\alpha < 1$ chosen independently for each mismatch based on the ablation in Figure 8. The performance of this configuration is reported by the red line in Figure 10. We add a 1 in brackets when standard GAILfO outperforms the robust version. The value outside brackets denotes the best value found for the robust version.

	Relative Mass				
	0.5	0.75	1.0	1.5	2.0
HalfCheetah	0.96	0.97	0.98	0.96	0.97
Walker	0.98	0.95	0.97	0.999	0.98
Hopper	0.9	0.97	0.97	0.98	0.999
InvDoublePendulum	0.98	0.99	0.97	0.96	0.97
Swimmer	0.96 (1)	0.999 (1)	0.95	0.95	0.98

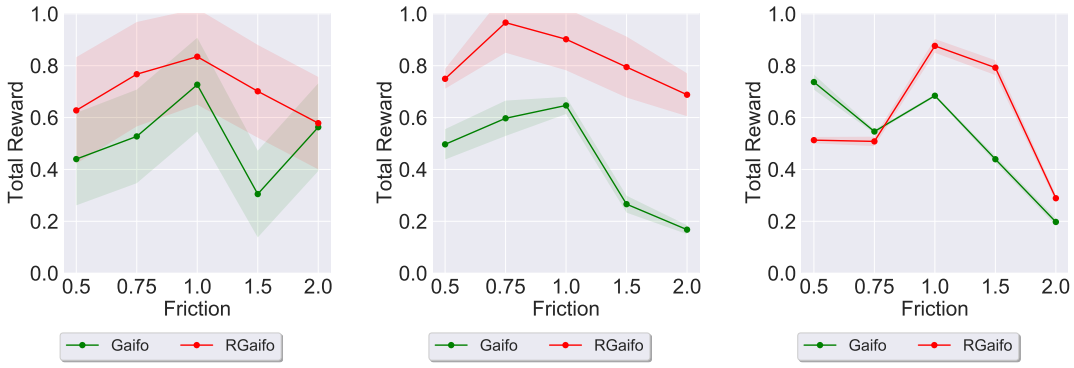


(a) HalfCheetah (b) Walker (c) Hopper



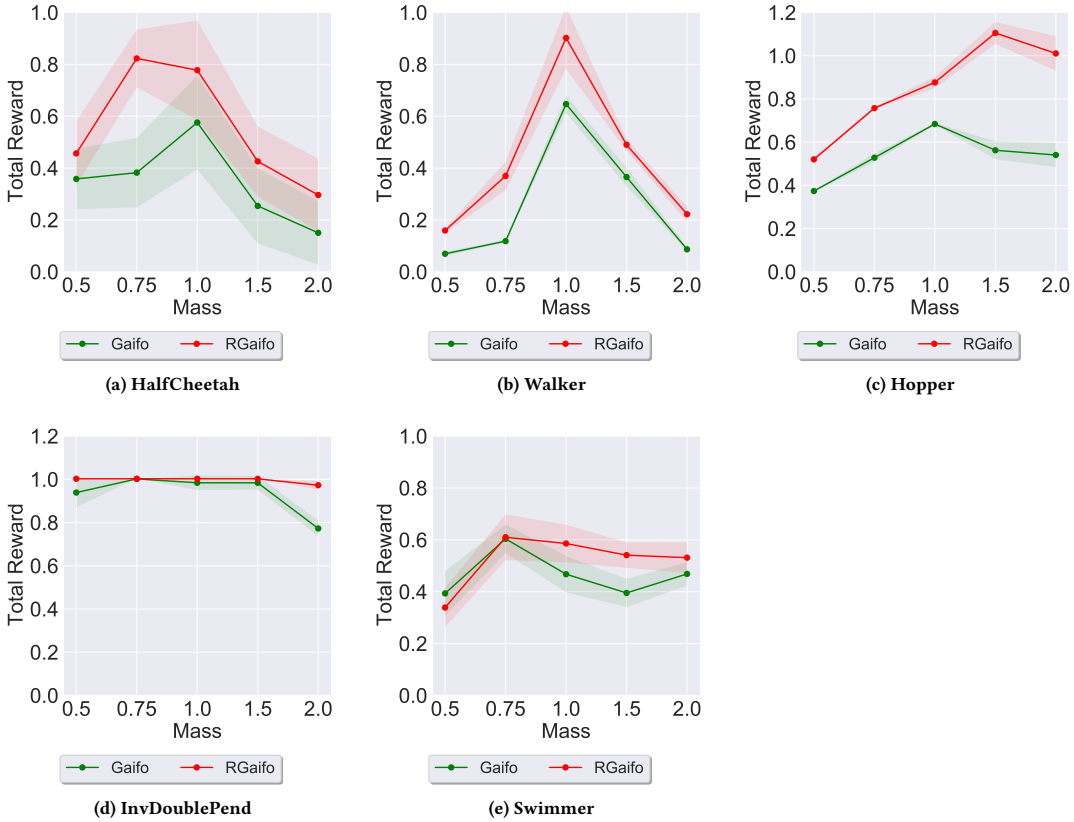
(d) InvDoublePend (e) Swimmer

Figure 8: The average (over 3 seeds) transfer performance of Algorithm 1 with different values of α. The ablation shown here is used to choose α in Figure 2. The x-axis denotes the relative mass of the learner environment M^{sim}. The policies are evaluated in $M^{\text{real}}_{c^*}$ over $1e5$ steps truncating the last episode if it does not terminate. Note that robust-GAILfO with $\alpha = 1$ corresponds to GAILfO.



(a) HalfCheetah (b) Walker (c) Hopper

Figure 9: Average performance (over 3 seeds) of Algorithm 1 with the value of α that is chosen indipendently for each mismatch (i.e. each point on the x-axis). The choice is made picking the best performing α for each mismatch in Figure 7. The x-axis reports the relative friction of the learner environment. The policies are evaluated over $1e5$ steps truncating the last episode if it does not terminate.The values chosen for α are given in Table 2.



(a) HalfCheetah (b) Walker (c) Hopper

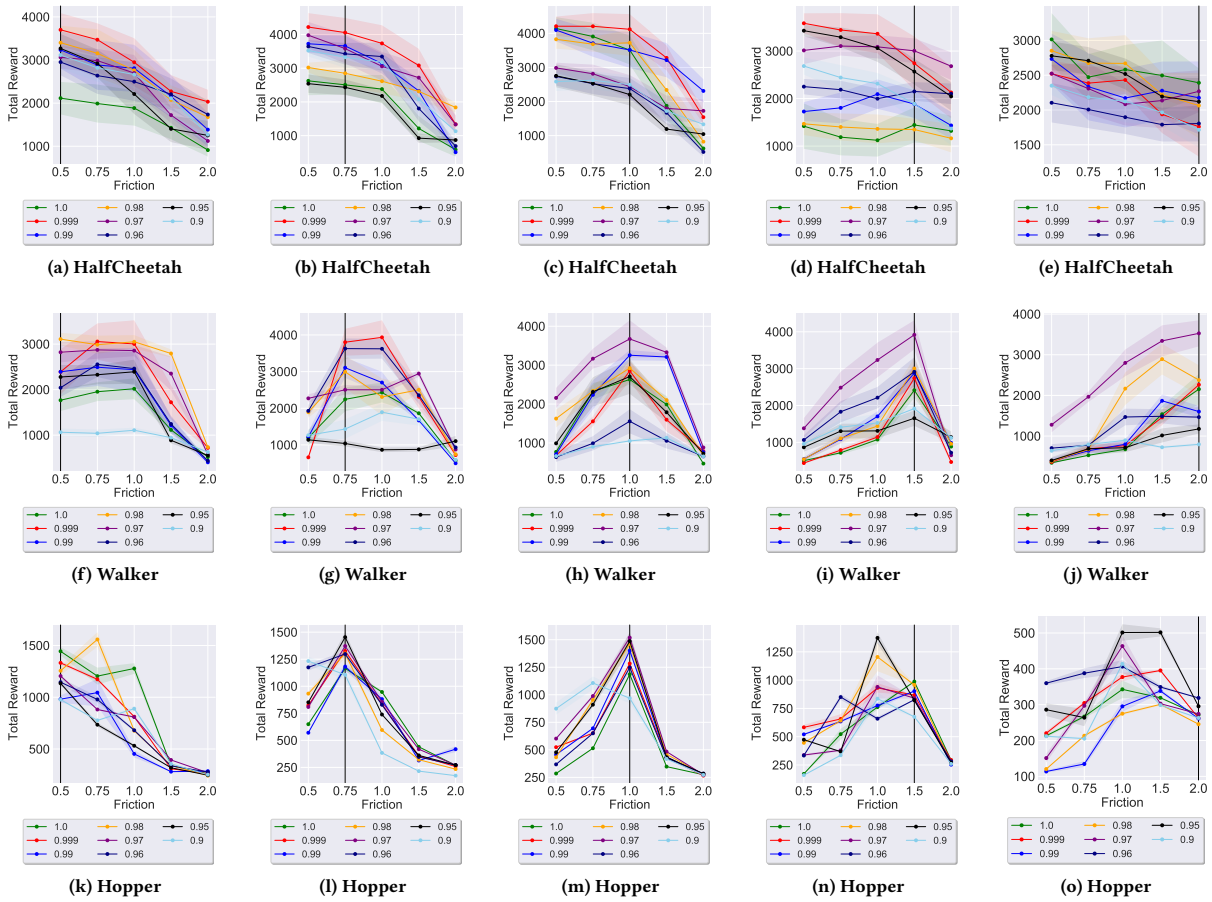
(d) InvDoublePend (e) Swimmer

Figure 10: Average performance (over 3 seeds) of Algorithm 1 with the value of α that is chosen indipendently for each mismatch (i.e. each point on the x-axis). The choice is made picking the best performing α for each mismatch in Figure 8. The x-axis reports the relative mass of the learner environment. The policies are evaluated over $1e5$ steps truncating the last episode if it does not terminate. The values chosen for α are given in Table 3.

D ROBUST PERFORMANCE: MUJOCO

We present the following results:

- The ablation study on the robust performance of Algorithm 1 with different values of α under the relative friction variations (see Figure 11).
- The ablation study on the robust performance of Algorithm 1 with different values of α under the relative mass variations (see Figure 12).

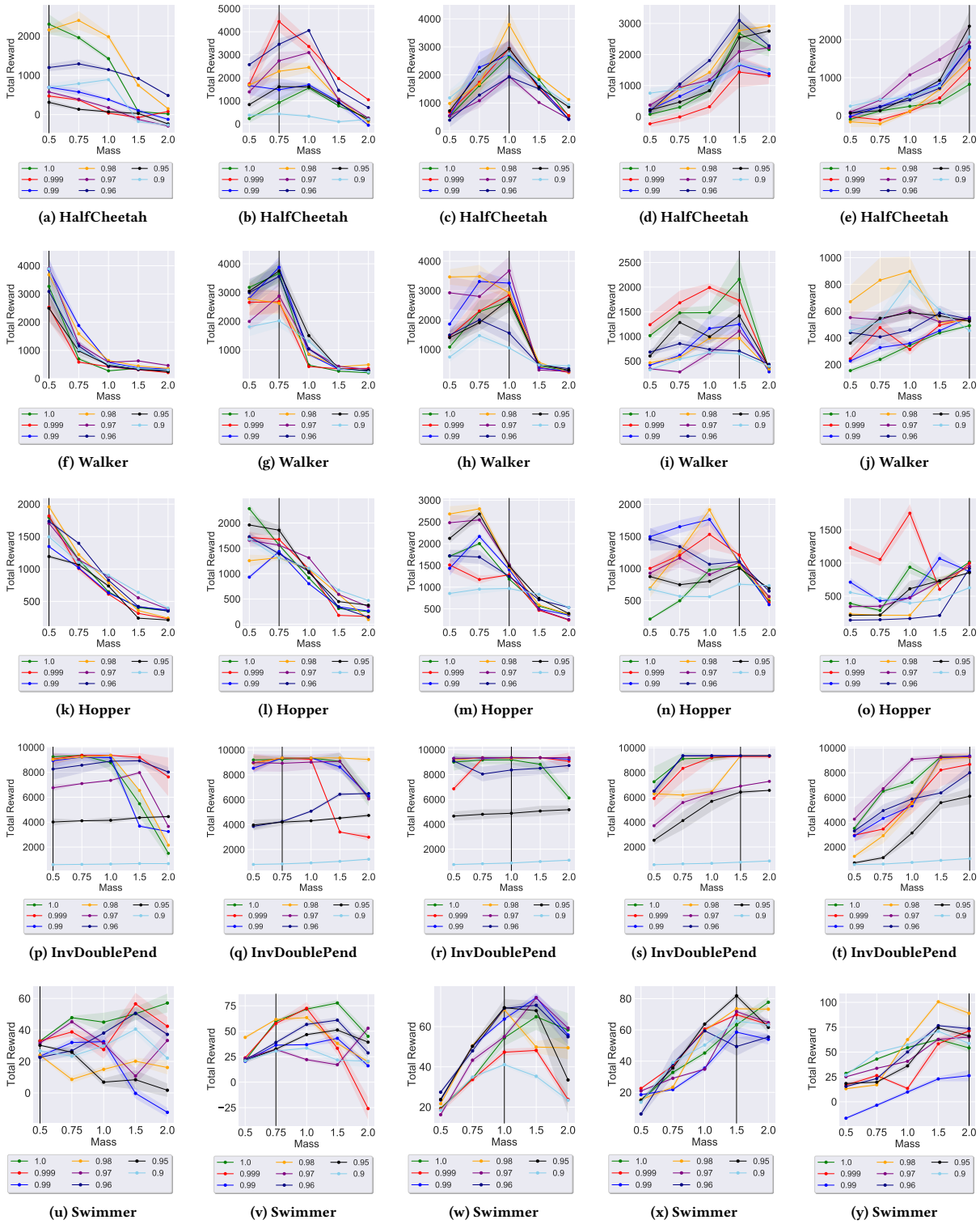


(a) HalfCheetah (b) HalfCheetah (c) HalfCheetah (d) HalfCheetah (e) HalfCheetah

(f) Walker (g) Walker (h) Walker (i) Walker (j) Walker

(k) Hopper (l) Hopper (m) Hopper (n) Hopper (o) Hopper

Figure 11: The average (over 3 seeds) robust performance of Algorithm 1 with different values of α. The ablation shown here is used to choose α in Figure 3. The expert environment M^{real}, in which the demonstrations are collected, has relative friction 1.0. In each plot, the black vertical line corresponds to the relative friction of the learner environment M^{sim} where we trained the policy with Algorithm 1. The x-axis denotes the relative friction of the test environment M^{test} in which the policies are evaluated. The policies are evaluated over $1e5$ steps truncating the last episode if it does not terminate. Note that robust-GAILfO with $\alpha = 1$ corresponds to GAILfO.



(a) HalfCheetah (b) HalfCheetah (c) HalfCheetah (d) HalfCheetah (e) HalfCheetah

(f) Walker (g) Walker (h) Walker (i) Walker (j) Walker

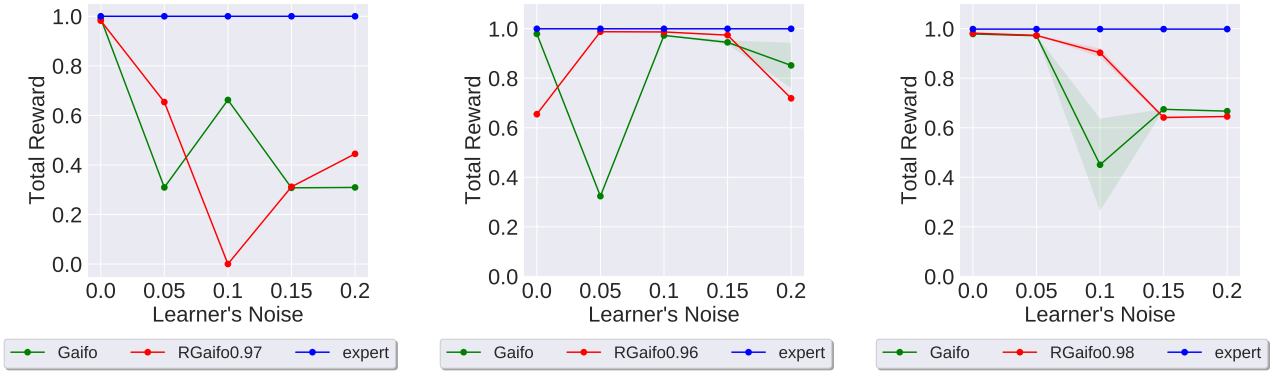
(k) Hopper (l) Hopper (m) Hopper (n) Hopper (o) Hopper

(p) InvDoublePend (q) InvDoublePend (r) InvDoublePend (s) InvDoublePend (t) InvDoublePend

(u) Swimmer (v) Swimmer (w) Swimmer (x) Swimmer (y) Swimmer

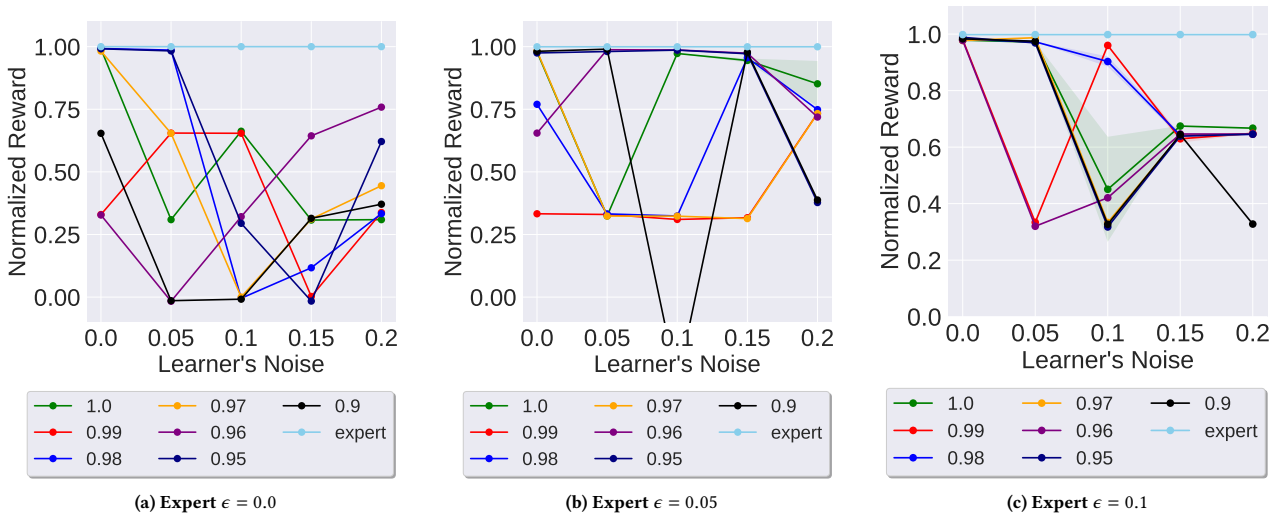
Figure 12: The average (over 3 seeds) robust performance of Algorithm 1 with different values of α. The ablation shown here is used to choose α in Figure 4. The expert environment M^{real}, in which the demonstrations are collected, has relative mass 1.0. In each plot, the black vertical line corresponds to the relative mass of the learner environment M^{sim} where we trained the policy with Algorithm 1. The x-axis denotes the relative mass of the test environment M^{test} in which the policies are evaluated. The policies are evaluated over $1e5$ steps truncating the last episode if it does not terminate. Note that robust-GAILfO with $\alpha = 1$ corresponds to GAILfO.

E TRANSFER PERFORMANCE: CONTINUOUS GRIDWORLD



(a) Expert $\epsilon = 0.0$ **(b) Expert $\epsilon = 0.05$** **(c) Expert $\epsilon = 0.1$**

Figure 13: The average (over 3 seeds) transfer performance of Algorithm 1 with fixed value of α for each mismatch (i.e., each point on the x-axis) in the environment shown in Figure 5. The x-axis denotes the ϵ value of the learner environment. The policies are evaluated over $1e5$ steps truncating the last episode if it does not terminate.

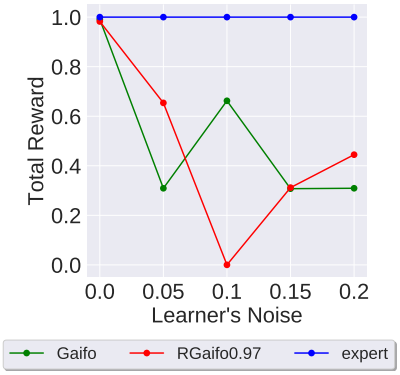


(a) Expert $\epsilon = 0.0$ **(b) Expert $\epsilon = 0.05$** **(c) Expert $\epsilon = 0.1$**

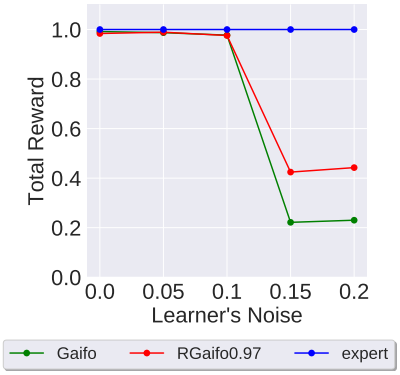
Figure 14: The average (over 3 seeds) transfer performance of Algorithm 1 with different values of α for each mismatch (i.e., each point on the x-axis) in the environment shown in Figure 5. The x-axis denotes the ϵ value of the learner environment. The policies are evaluated over $1e5$ steps truncating the last episode if it does not terminate. The ablation shown here is used to choose α in Figure 13. Note that robust-GAILfO with $\alpha = 1$ corresponds to GAILfO.

F ADDITIONAL EXPERIMENTS ON CHOICE OF α

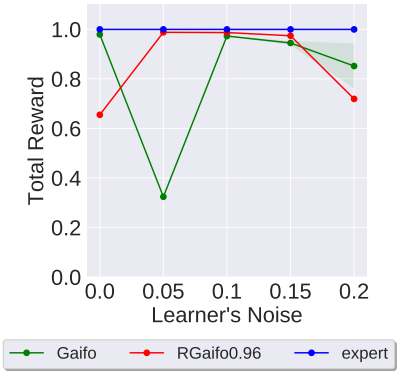
In this section, we aim to understand whether our strategy of choosing suitable α value introduces maximization bias. For example, in Figure 14, the best performing α is chosen, and its performance curve (w.r.t. the original seeds used for training) is presented in Figure 13. To avoid this bias, for the chosen best performing α in Figure 14, we conduct a new set of runs with a new set of seeds. The new results presented in Figure 15 suggest that our α selection process does not introduce maximization bias.



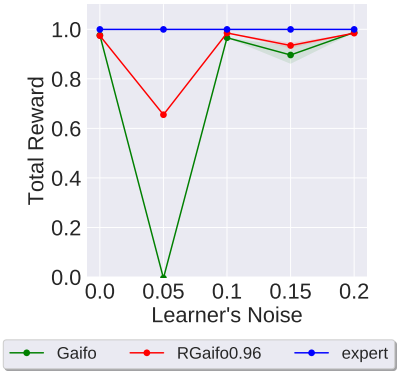
(a) Expert $\epsilon = 0.0$; Original seeds



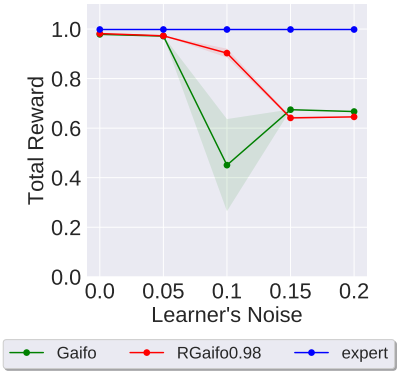
(b) Expert $\epsilon = 0.0$; New seeds



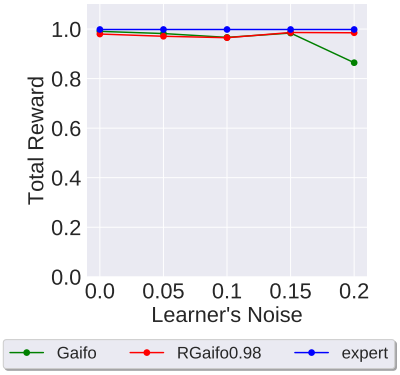
(c) Expert $\epsilon = 0.05$; Original seeds



(d) Expert $\epsilon = 0.05$; New seeds



(e) Expert $\epsilon = 0.1$; Original seeds



(f) Expert $\epsilon = 0.1$; New seeds

Figure 15: Experiments for understanding whether our strategy of choosing suitable α value introduces maximization bias.