In *Proceedings of the 34th Conference on Neural Information Processing Systems* (NeurIPS 2020),
Virtual, December 6-12th, 2020

An Imitation from Observation Approach to Transfer Learning with Dynamics Mismatch

Siddharth Desai[§]
Department of Mechanical Engineering
The University of Texas at Austin
sidrdesai@utexas.edu

Ishan Durugkar [§]
Department of Computer Science
The University of Texas at Austin
ishand@cs.utexas.edu

Haresh Karnan [§]
Department of Mechanical Engineering
The University of Texas at Austin
haresh.miriyala@utexas.edu

Garrett Warnell
Army Research Laboratory
garrett.a.warnell.civ@mail.mil

Josiah P. Hanna [*]
School of Informatics
The University of Edinburgh
josiah.hanna@ed.ac.uk

Peter Stone
Department of Computer Science
The University of Texas at Austin
and Sony AI
pstone@cs.utexas.edu

Abstract

We examine the problem of transferring a policy learned in a source environment to a target environment with different dynamics, particularly in the case where it is critical to reduce the amount of interaction with the target environment during learning. This problem is particularly important in sim-to-real transfer because simulators inevitably model real-world dynamics imperfectly. In this paper, we show that one existing solution to this transfer problem—*grounded action transformation*—is closely related to the problem of *imitation from observation* (IfO): learning behaviors that mimic the observations of behavior demonstrations. After establishing this relationship, we hypothesize that recent state-of-the-art approaches from the IfO literature can be effectively repurposed for grounded transfer learning. To validate our hypothesis we derive a new algorithm—generative adversarial reinforced action transformation (GARAT)—based on adversarial imitation from observation techniques. We run experiments in several domains with mismatched dynamics, and find that agents trained with GARAT achieve higher returns in the target environment compared to existing black-box transfer methods.

1 Introduction

Transfer learning with dynamics mismatch refers to using experience in a source environment to more efficiently learn control policies that perform well in a target environment, where the two environments differ only in their transition dynamics. For example, if the friction coefficient in the source and target environments is sufficiently different it might cause the action of placing a foot on the ground to work well in one environment, but cause the foot to slip in the other. One possible application of such transfer is where the source environment is a simulator and the target environment

[*] to be joining the Computer Sciences department at the University of Wisconsin – Madison
[§] Equal contribution

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

is a robot in the real world, called sim-to-real. In sim-to-real scenarios, source environment (simulator) experience is readily available, but target environment (real world) experience is expensive. Sim-to-real transfer has been used effectively to learn a fast humanoid walk [15], dexterous manipulation [29, 22, 38, 26, 6, 24, 23], and agile locomotion skills [32]. In this work, we focus on the paradigm of simulator grounding [10, 15, 8], which modifies the source environment's dynamics to more closely match the target environment dynamics using a relatively small amount of target environment data. Policies then learned in such a grounded source environment transfer better to the target environment.

Separately, the machine learning community has also devoted attention to imitation learning [5], i.e. the problem of learning a policy to mimic demonstrations provided by another agent. In particular, recent work has considered the specific problem of *imitation from observation* (IfO) [25], in which an imitator mimics the expert's behavior without knowing which actions the expert took, only the outcomes of those actions (i.e. state-only demonstrations). While the lack of action information presents an additional challenge, recently-proposed approaches have suggested that this challenge may be addressable [48, 50].

In this paper, we show that a particular grounded transfer technique that has been shown to successfully accomplish sim-to-real transfer, called *grounded action transformation* (GAT) [15], can be seen as a form of IfO. We therefore hypothesize that recent, state-of-the-art approaches for addressing the IfO problem might also be effective for grounding the source environment, leading to improved transfer. Specifically, we derive a distribution-matching objective similar to ones used in adversarial approaches for generative modeling [14], imitation learning [18], and IfO [49] with considerable empirical success. Based on this objective, we propose a novel algorithm, Generative Adversarial Reinforced Action Transformation (GARAT), to ground the source environment by reducing the distribution mismatch between the source and target environments.

Our experiments confirm our hypothesis by showing that GARAT reduces the difference in the dynamics between two environments more effectively than GAT. Moreover, our experiments show that, in several domains, this improved grounding translates to better transfer of policies from one environment to the other.

The contributions of this paper are as follows: *(1)* we show that learning the grounded action transformation can be seen as an *IfO* problem, *(2)* we derive a novel adversarial imitation learning algorithm, GARAT, to learn an action transformation policy for transfer learning with dynamics mismatch, and *(3)* we experimentally evaluate the efficacy of GARAT for transfer with dynamics mismatch.

2 Background

We begin by introducing notation, reviewing the transfer learning with dynamics mismatch problem formulation, and describing the action transformation approach for sim-to-real transfer. We also provide a brief overview of imitation learning and imitation from observation.

2.1 Notation

We consider here sequential decision processes formulated as Markov decision processes (MDPs) [42]. An MDP $\mathcal{M}$ is a tuple $\langle \mathcal{S}, \mathcal{A}, R, P, \gamma, \rho_0 \rangle$ consisting of a set of states, $\mathcal{S}$; a set of actions, $\mathcal{A}$; a reward function, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longmapsto \Delta([r_{\min}, r_{\max}])$ (where $\Delta([r_{\min}, r_{\max}])$ denotes a distribution over the interval $[r_{\min}, r_{\max}] \subset \mathbb{R}$); a discount factor, $\gamma \in [0, 1)$; a transition function, $P : \mathcal{S} \times \mathcal{A} \longmapsto \Delta(\mathcal{S})$; and an initial state distribution, $\rho_0 : \Delta(\mathcal{S})$. An RL agent uses a policy $\pi : \mathcal{S} \longmapsto \Delta(\mathcal{A})$ to select actions in the environment. In an environment with transition function $P \in \mathcal{T}$, the agent aims to learn a policy $\pi \in \mathbf{\Pi}$ to maximize its expected discounted return $\mathbb{E}_{\pi,P}[G_0] = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]$, where $R_t \sim R(s_t, a_t, s_{t+1})$, $s_{t+1} \sim P(s_t, a_t)$, $a_t \sim \pi(s_t)$, and $s_0 \sim \rho_0$.

Given a fixed π and a specific transition function P_q, the marginal transition distribution is $\rho_q(s, a, s') := (1-\gamma)\pi(a|s)P_q(s'|s, a) \sum_{t=0}^{\infty} \gamma^t p(s_t = s|\pi, P_q)$ where $p(s_t = s|\pi, P_q)$ is the probability of being in state s at time t. The marginal transition distribution is the probability of being in state s marginalized over time t, taking action a under policy π, and ending up in state s' under transition function P_q (laid out more explicitly in Appendix A). We can denote the expected return

under a policy π and a transition function P_q in terms of this marginal distribution as:

$$\mathbb{E}_{\pi,q}[G_0] = \frac{1}{(1-\gamma)} \sum_{s,a,s'} \rho_q(s,a,s') R(s'|s,a) \tag{1}$$

2.2 Transfer Learning with Dynamics Mismatch and Grounded Action Transformation

Let $P_s, P_t \in \mathcal{T}$ be the transition functions for two otherwise identical MDPs, $\mathcal{M}_s$ and $\mathcal{M}_t$, representing the source and target environments respectively. Transfer learning with dynamics mismatch, as opposed to transfer learning in general, aims to train an agent policy to maximize return in $\mathcal{M}_t$ with limited trajectories from $\mathcal{M}_t$, and as many as needed in $\mathcal{M}_s$.

The work presented here is specifically concerned with a particular class of approaches used in sim-to-real transfer known as simulator grounding approaches [1, 8, 10]. Here the source environment is the simulator and the target environment is the real world. These approaches use some interactions with the target environment to *ground* the source environment dynamics to more closely match the target environment dynamics. Because it may sometimes be difficult or impossible to modify the source environment itself (when the source environment is a black-box simulator, for example), the recently-proposed grounded action transformation (GAT) approach [15] seeks to instead induce grounding by modifying the agent's actions before using them in the source environment. This modification is accomplished via an action transformation function $\pi_g : \mathcal{S} \times \mathcal{A} \longmapsto \Delta(\mathcal{A})$ that takes as input the state and action of the agent, and produces an action to be presented to the source environment. From the agent's perspective, composing the action transformation with the source environment changes the source environment's transition function. We call this modified source environment the *grounded* environment, and its transition function is given by

$$P_g(s'|s,a) = \sum_{\tilde{a} \in \mathcal{A}} P_s(s'|s,\tilde{a}) \pi_g(\tilde{a}|s,a) \tag{2}$$

The action transformation approach aims to learn function $\pi_g \in \mathbf{\Pi}_g$ such that the resulting transition function P_g is as close as possible to P_t. We denote the marginal transition distributions in the source and target environments by ρ_s and ρ_t respectively, and $\rho_g \in \mathcal{P}_g$ for the grounded environment.

GAT learns a model of the target environment dynamics $\hat{P}_t(s'|s,a)$, an inverse model of the source environment dynamics $\hat{P}_s^{-1}(a|s,s')$, and uses the composition of the two as the action transformation function, i.e. $\pi_g(\tilde{a}|s,a) = \hat{P}_s^{-1}(\tilde{a}|s, \hat{P}_t(s'|s,a))$.

2.3 Imitation Learning

In parallel to advances in sim-to-real transfer, the machine learning community has also made considerable progress on the problem of imitation learning. Imitation learning [5, 36, 39] is the problem setting where an agent tries to mimic trajectories $\{\xi_0, \xi_1, \ldots\}$ where each ξ is a demonstrated trajectory $\{(s_0,a_0),(s_1,a_1),\ldots\}$ induced by an expert policy π_{exp}.

Various methods have been proposed to address the imitation learning problem. Behavioral cloning [4] uses the expert's trajectories as labeled data and uses supervised learning to recover the maximum likelihood policy. Another approach instead relies on reinforcement learning to learn the policy, where the required reward function is recovered using inverse reinforcement learning (IRL) [28]. IRL aims to recover a reward function under which the demonstrated trajectories would be optimal.

A related setting to learning from state-action demonstrations is the imitation from observation (IfO) [25, 30, 48, 49] problem. Here, an agent observes an expert's state-only trajectories $\{\zeta_0, \zeta_1, \ldots\}$ where each ζ is a sequence of states $\{s_0, s_1, \ldots\}$. The agent must then learn a policy $\pi(a|s)$ to imitate the expert's behavior, without being given labels of which actions to take.

3 GAT as Imitation from Observation

We now show that the underlying problem of GAT—i.e., learning an action transformation for sim-to-real transfer—can also been seen as an IfO problem. Adapting the definition by Liu et al. [25], an IfO problem is a sequential decision-making problem where the policy imitates state-only trajectories

$\{\zeta_0, \zeta_1, \ldots\}$ produced by a Markov process, with no information about what actions generated those trajectories. To show that the action transformation learning problem fits this definition, we must show that it *(1)* is a sequential decision-making problem and *(2)* aims to imitate state-only trajectories produced by a Markov process, with no information about what actions generated those trajectories.

Starting with *(1)*, it is sufficient to show that the action transformation function is a policy in an MDP [34]. This action transformation MDP can be seen clearly if we combine the target environment MDP and the fixed agent policy π. Let the joint state and action space $\mathcal{X} := \mathcal{S} \times \mathcal{A}$ with $x := (s, a) \in \mathcal{X}$ be the state space of this new MDP. The combined transition function is $P_s^x(x'|x, \tilde{a}) = P_s(s'|s, \tilde{a})\pi(a'|s')$, where $x' = (s', a')$, and initial state distribution is $\rho_0^x(x) = \rho_0(s)\pi(a|s)$. For completeness, we consider a reward function $R^x : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \longmapsto \Delta([r_{\min}, r_{\max}])$ and discount factor $\gamma_x \in [0, 1)$, which are not essential for an IfO problem. With these components, the action transformation environment is an MDP $\langle \mathcal{X}, \mathcal{A}, R^x, P_s^x, \gamma_x, \rho_0^x \rangle$. The action transformation function $\pi_g(\tilde{a}|s, a)$, now $\pi_g^x(\tilde{a}|x)$, is then clearly a mapping from states to a distribution over actions, i.e. it is a policy in an MDP. Thus, the action transformation learning problem is a sequential decision-making problem.

We now consider the action transformation objective to show *(2)*. When learning the action transformation policy, we have trajectories $\{\tau_0, \tau_1, \ldots\}$, where each trajectory $\tau = \{(s_0, a_0 \sim \pi(s_0)), (s_1, a_1 \sim \pi(s_1)), \ldots\}$ is obtained by sampling actions from agent policy π in the target environment. Re-writing τ in the above MDP, $\tau = \{x_0, x_1, \ldots\}$. If an expert action transformation policy $\pi_g^* \in \mathbf{\Pi}_g$ is capable of mimicking the dynamics of the target environment, $P_t^x(x'|x) = \sum_{\tilde{a} \in \mathcal{A}} P_s^x(x'|x, \tilde{a})\pi_g^*(\tilde{a}|x)$, then we can consider the above trajectories to be produced by a Markov process with dynamics $P_s^x(x'|x, \tilde{a})$ and policy $\pi_g^*(\tilde{a}|x)$. The action transformation aims to imitate the state-only trajectories $\{\tau_0, \tau_1, \ldots\}$ produced by a Markov process, with no information about what actions generated those trajectories.

The problem of learning the action transformation thus satisfies the conditions we identified above, and so it is an IfO problem.

4 Generative Adversarial Reinforced Action Transformation

The insight above naturally leads to the following question: *if learning an action transformation for transfer learning is equivalent to IfO, might recently-proposed IfO approaches lead to better transfer learning approaches?* To investigate the answer, we derive a novel generative adversarial approach inspired by GAIfO[49] that can be used to train the action transformation policy using IfO. A source environment grounded with this action transformation policy can then be used to train an agent policy which can be expected to transfer effectively to a given target environment. We call our approach generative adversarial reinforced action transformation (GARAT), and Algorithm 1 lays out its details.

The rest of this section details our derivation of the objective used in GARAT. First, in Section 4.1, we formulate a procedure for action transformation using a computationally expensive IRL step to extract a reward function and then learning an action transformation policy based on that

Algorithm 1 GARAT

Input: Target environment with P_t, source environment with P_s, number of update steps N
Agent policy π with parameters η , pretrained in source environment;
Initialize action transformation policy π_g with parameters θ
Initialize discriminator D_ϕ with parameters ϕ
while *performance of policy π in target environment not satisfactory* **do**
  Rollout policy π in target environment to obtain trajectories $\{\tau_{t,1}, \tau_{t,2}, \ldots\}$
  for $i = 0, 1, 2, \ldots N$ **do**
    Rollout Policy π in grounded source environment and obtain trajectories $\{\tau_{g,1}, \tau_{g,2}, \ldots\}$
    Update parameters ϕ of D_ϕ using gradient descent to minimize
    $-\left(\mathbb{E}_{\tau_g}[\log(D_\phi(s, a, s'))] + \mathbb{E}_{\tau_t}[\log(1 - D_\phi(s, a, s'))\right)$
    Update parameters θ of π_g using policy gradient with reward $-[\log D_\phi(s, a, s')]$
  end
  Optimize parameters η of π in source environment grounded with action transformer π_g
end

reward. Then, in Section 4.2, we show that this entire procedure is equivalent to directly reducing the marginal transition distribution discrepancy between the target environment and the grounded source environment. This is important, as recent work [14, 18, 49] has shown that adversarial approaches are a promising algorithmic paradigm to reduce such discrepancies. Thus, in Section 4.3, we explicitly formulate a generative adversarial objective upon which we build the proposed approach.

4.1 Action Transformation Inverse Reinforcement Learning

We first lay out a procedure to learn the action transformation policy by extracting the appropriate cost function, which we term action transformation IRL (ATIRL). We use the cost function formulation in our derivation, similar to previous work [18, 49]. ATIRL aims to identify a cost function such that the observed target environment transitions yield higher return than any other possible transitions. We consider the set of cost functions $\mathcal{C}$ as all functions $\mathbb{R}^{\mathcal{S}\times\mathcal{A}\times\mathcal{S}} = \{c : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \longmapsto \mathbb{R}\}$.

$$\texttt{ATIRL}_{\psi}(P_t) := \operatorname*{argmax}_{c\in\mathcal{C}} -\psi(c) + \left(\min_{\pi_g\in\mathbf{\Pi}_g} \mathbb{E}_{\rho_g}[c(s,a,s')]\right) - \mathbb{E}_{\rho_t}[c(s,a,s')] \tag{3}$$

where $\psi : \mathbb{R}^{\mathcal{S}\times\mathcal{A}\times\mathcal{S}} \longmapsto \overline{\mathbb{R}}$ is a (closed, proper) convex reward function regularizer, and $\overline{\mathbb{R}}$ denotes the extended real numbers $\mathbb{R}\bigcup\{\infty\}$. This regularizer is used to avoid overfitting the expressive set $\mathcal{C}$. Note that π_g influences ρ_g (Equation 10 in Appendix A) and P_t influences ρ_t. Similar to GAIfO, we do not use causal entropy in our ATIRL objective due to the surjective mapping from $\mathbf{\Pi}_g$ to $\mathcal{P}_g$.

The action transformation then uses this per-step cost function as a reward function in an RL procedure: $\texttt{RL}(c) := \operatorname{argmin}_{\pi_g\in\mathbf{\Pi}_g} \mathbb{E}_{\rho_g}[c(s,a,s')]$. We assume here for simplicity that there is an action transformation policy that can mimic the target environment dynamics perfectly. That is, there exists a policy $\pi_g \in \mathbf{\Pi}_g$, such that $P_g(s'|s,a) = P_t(s'|s,a)\forall s \in \mathcal{S}, a \in \mathcal{A}$. We denote the RL procedure applied to the cost function recovered by ATIRL as $\texttt{RL}\circ\texttt{ATIRL}_{\psi}(P_t)$.

4.2 Characterizing the Policy Induced by ATIRL

This section shows that it is possible to bypass the ATIRL step and learn the action transformation policy directly from data. We show that ψ-regularized $\texttt{RL}\circ\texttt{ATIRL}_{\psi}(P_t)$ implicitly searches for policies that have a marginal transition distribution close to the target environment's, as measured by the convex conjugate of ψ, which we denote as ψ^*. As a practical consequence, we will then be able to devise a method for minimizing this divergence through the use of generative adversarial techniques in Section 4.3. But first, we state our main theoretical claim:

Theorem 1. $\texttt{RL}\circ\texttt{ATIRL}_{\psi}(P_t)$ *and* $\operatorname{argmin}_{\pi_g} \psi^*(\rho_g - \rho_t)$ *induce policies that have the same marginal transition distribution,* ρ_g.

To reiterate, the agent policy π is fixed. So the only decisions affecting the marginal transition distributions are of the action transformation policy π_g. We can now state the following proposition:

Proposition 4.1. *For a given* ρ_g *generated by a fixed policy* π, P_g *is the only transition function whose marginal transition distribution is* ρ_g.

Proof in Appendix B.1. We can also show that if two transition functions are equal, then the optimal policy in one will be optimal in the other.

Proposition 4.2. *If* $P_t = P_g$, *then* $\operatorname{argmax}_{\pi\in\mathbf{\Pi}} \mathbb{E}_{\pi,P_g}[G_0] = \operatorname{argmax}_{\pi\in\mathbf{\Pi}} \mathbb{E}_{\pi,P_t}[G_0]$.

Proof in Appendix B.2. We now prove Theorem 1, which characterizes the policy learned by $\texttt{RL}(\tilde{c})$ on the cost function $\tilde{c}$ recovered by $\texttt{ATIRL}_{\psi}(P_t)$.

Proof of Theorem 1. To prove Theorem 1, we prove that $\texttt{RL}\circ\texttt{ATIRL}_{\psi}(P_t)$ and $\operatorname{argmin}_{\pi_g} \psi^*(\rho_g - \rho_t)$ result in the same marginal transition distribution. This proof has three parts, two of which are proving that both objectives above can be formulated as optimizing over marginal transition distributions. The third is to show that these equivalent objectives result in the same distribution.

The output of both $\texttt{RL}\circ\texttt{ATIRL}_{\psi}(P_t)$ and $\operatorname{argmin}_{\pi_g} \psi^*(\rho_g - \rho_t)$ are policies. To compare the marginal distributions, we first establish a different $\overline{\texttt{RL}}\circ\overline{\texttt{ATIRL}}_{\psi}(P_t)$ objective that we argue has the same

marginal transition distribution as $\texttt{RL} \circ \texttt{ATIRL}_\psi(P_t)$. We define

$$\overline{\texttt{ATIRL}}_\psi(P_t) := \operatorname*{argmax}_{c \in \mathcal{C}} -\psi(c) + \left(\min_{\rho_g \in \mathcal{P}_g} \mathbb{E}_{\rho_g} \left[c(s, a, s') \right] \right) - \mathbb{E}_{\rho_t} \left[c(s, a, s') \right] \tag{4}$$

with the same ψ and $\mathcal{C}$ as Equation 3, and similar except the internal optimization for Equation 3 is over $\pi_g \in \mathbf{\Pi}_g$, while it is over $\rho_g \in \mathcal{P}_g$ for Equation 4. We define an RL procedure $\overline{\texttt{RL}}(\bar{c}) := \operatorname{argmin}_{\rho_g \in \mathcal{P}_g} \mathbb{E}_{\rho_g} c(s, a, s')$ that returns a marginal transition distribution $\rho_g \in \mathcal{P}_g$ which minimizes the given cost function $\bar{c}$. $\overline{\texttt{RL}}(\bar{c})$ will output the marginal transition distribution $\overline{\rho}_g$.

Lemma 4.1. $\overline{\texttt{RL}} \circ \overline{\texttt{ATIRL}}_\psi(P_t)$ *outputs a marginal transition distribution* $\overline{\rho}_g$ *which is equal to* $\tilde{\rho}_g$ *induced by* $\texttt{RL} \circ \texttt{ATIRL}_\psi(P_t)$.

Proof in Appendix B.3. The mapping from $\mathbf{\Pi}_g$ to $\mathcal{P}_g$ is not injective, and there could be multiple policies π_g that lead to the same marginal transition distribution. The above lemma is sufficient for proof of Theorem 1, however, since we focus on the effect of the policy on the transitions.

Lemma 4.2. $\overline{\texttt{RL}} \circ \overline{\texttt{ATIRL}}_\psi(P_t) = \operatorname{argmin}_{\rho_g \in \mathcal{P}_g} \psi^*(\rho_g - \rho_t)$.

The proof in Appendix B.4 relies on the optimal cost function and the optimal policy forming a saddle point, ψ^* leading to a minimax objective, and these objectives being the same.

Lemma 4.3. *The marginal transition distribution of* $\operatorname{argmin}_{\pi_g} \psi^*(\rho_g - \rho_t)$ *is equal to* $\operatorname{argmin}_{\rho_g \in \mathcal{P}_g} \psi^*(\rho_g - \rho_t)$.

Proof in appendix B.5. With these three lemmas, we have proved that $\texttt{RL} \circ \texttt{ATIRL}_\psi(P_t)$ and $\operatorname{argmin}_{\pi_g} \psi^*(\rho_g - \rho_t)$ induce policies that have the same marginal transition distribution. □

Theorem 1 thus tells us that the objective $\operatorname{argmin}_{\pi_g} \psi^*(\rho_g - \rho_t)$ is equivalent to the procedure from Section 4.1. In the next section, we choose a function ψ which leads to our adversarial objective.

4.3 Forming the Adversarial Objective

Section 4.2 laid out the objective we want to minimize. To solve $\operatorname{argmin}_{\pi_g} \psi^*(\rho_g - \rho_t)$ we require an appropriate regularizer ψ. GAIL [18] and GAIfO [49] optimize similar objectives and have shown a regularizer similar to the following to work well:

$$\psi(c) = \begin{cases} \mathbb{E}_t[g(c(s, a, s'))] & \text{if } c < 0 \\ +\infty & \text{otherwise} \end{cases} \text{ where } g(x) = \begin{cases} -x - log(1 - e^x) & \text{if } x < 0 \\ +\infty & \text{otherwise} \end{cases} \tag{5}$$

It is closed, proper, convex and has a convex conjugate leading to the following minimax objective:

$$\min_{\pi_g \in \mathbf{\Pi}_g} \psi^*(\rho_g - \rho_t) = \min_{\pi_g \in \mathbf{\Pi}_g} \max_D \mathbb{E}_{P_g}[\log(D(s, a, s'))] + \mathbb{E}_{P_t}[\log(1 - D(s, a, s'))] \tag{6}$$

where the reward for the action transformer policy π_g is $-[\log(D(s, a, s'))]$, and $D : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longmapsto (0, 1)$ is a discriminative classifier. These properties have been shown in previous works [18, 49]. Algorithm 1 lays out the steps for learning the action transformer using the above procedure, which we call generative adversarial reinforced action transformation (GARAT).

5 Related Work

While our work lies in the space of transfer learning with dynamics mismatch, the eventual goal of this research is to enable effective sim-to-real transfer. In this section, we discuss the variety of sim-to-real methods, work more closely related to GARAT, and some related methods in the IfO literature. Sim-to-real transfer can be improved by making the agent's policy more robust to variations in the environment or by making the simulator more accurate w.r.t. the real world. The first approach, which we call policy robustness methods, encompasses algorithms that train a robust policy that performs well on a range of environments [20, 31, 32, 33, 35, 37, 45, 46]. Robust adversarial reinforcement learning (RARL) [33] is such an algorithm that learns a policy robust to adversarial perturbations

[43]. While primarily focused on training with a modifiable simulator, a version of RARL treats the simulator as a black-box by adding the adversarial perturbation directly to the protagonist's action. Additive noise envelope (ANE) [21] is another black-box robustness method which adds an envelope of Gaussian noise to the agent's action during training.

The second approach, known as domain adaption or system identification, grounds the simulator using real world data to make its transitions more realistic. Since hand engineering accurate simulators [44, 52] can be expensive and time consuming, real world data can be used to adapt low-fidelity simulators to the task at hand. Most simulator adaptation methods [1, 8, 10, 19] rely on access to a parameterized simulator.

GARAT, on the other hand, does not require a modifiable simulator and relies on an action transformation policy applied in the source environment to bring its transitions closer to the target environment. GAT[15] learns an action transformation function similar to GARAT. It was shown to have successfully learned and transferred one of the fastest known walk policies on the humanoid robot, Nao.

GARAT draws from recent generative adversarial approaches to imitation learning (GAIL [18]) and IfO (GAIfO [49]). AIRL[11], FAIRL[13], and WAIL[51] are related approaches which use different divergence metrics to reduce the marginal distribution mismatch. GARAT can be adapted to use any of these metrics, as we show in the appendix.

One of the insights of this paper is that grounding the simulator using action transformation can be seen as a form of IfO. BCO [48] is an IfO technique that utilizes behavioral cloning. I2L [12] is an *IfO* algorithm that aims to learn in the presence of transition dynamics mismatch in the expert and agent's domains, but requires millions of real world interactions to be competent.

6 Experiments

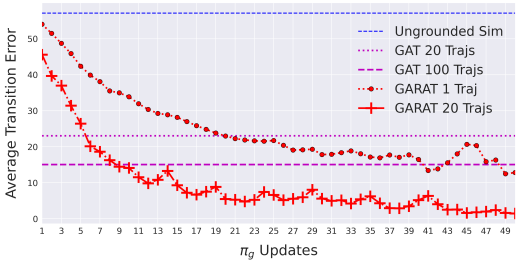
In this section, we conduct experiments to verify our hypothesis that GARAT leads to improved transfer in the presence of dynamics mismatch compared to previous methods. We also show that it leads to better source environment grounding compared to the previous action transformation approach, GAT.

We validate GARAT for transfer by transferring the agent policy between Open AI Gym [7] simulated environments with different transition dynamics. We highlight the Minitaur domain (Figure 2) as a particularly useful test since there exist two simulators, one of which has been carefully engineered for high fidelity to the real robot [44]. For other environments, the target environment is the source environment modified in different ways such that a policy trained in the source environment does not transfer well to the target environment. Details of these modifications are provided in Appendix C.1. Apart from a thorough evaluation across multiple different domains, this setup also allows us to compare GARAT and other algorithms against a policy trained directly in the target environment with millions of interactions, which is otherwise prohibitively expensive on a real robot. This setup also allows us to perform a thorough evaluation of sim-to-real algorithms across multiple different domains. We focus here on answering the following questions :

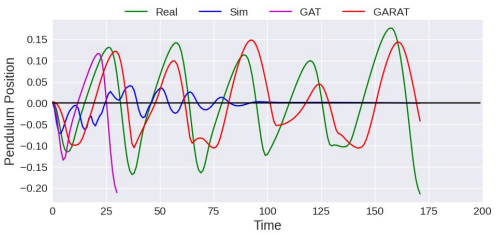
1. How well does GARAT ground the source environment with respect to the target environment?
2. Does GARAT lead to improved transfer with in the presence of dynamics mismatch, compared to other related methods?

6.1 Source Environment Grounding

In Figure 1, we evaluate how well GARAT grounds the source environment to the target environment both quantitatively and qualitatively. This evaluation is in the *InvertedPendulum* domain, where the target environment has a heavier pendulum than the source; implementation details are in Appendix C.1. In Figure 1a, we plot the average error in transitions in source environments grounded with GARAT and GAT with different amounts of target environment data, collected by deploying π in the target environment. In Figure 1b we deploy the same policy π from the same start state in the different environments (source, target, and grounded source). From both these figures it is evident that GARAT leads to a grounded source environment with lower error on average, and responses



(a) L2 norm of per step transition errors (lower is better) between different source environments and the target environment, shown over number of action transformation policy updates for GARAT.



(b) Example trajectories of the same agent policy deployed in different environments, plotted using the pendulum angle across time. Response of GARAT grounded source environment is the most like target environment.

Figure 1: Evaluation of source environment grounding with GARAT in *InvertedPendulum* domain

qualitatively closer to the target environment compared to GAT. Details of how we obtained these plots are in Appendix C.2.

6.2 Transfer Experiments

We now validate the effectiveness of GARAT at transferring a policy from source environment to target environment. For various MuJoCo [47] environments, we pretrain the agent policy π in the ungrounded source environment, collect target environment data with π, use GARAT to ground the source environment, re-train the agent policy until convergence in these grounded source environments, and then evaluate mean return across 50 episodes for the updated agent policy in the target environment.

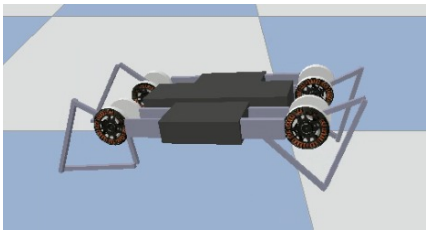


Figure 2: The Minitaur Domain

The agent policy π and action transformation policy π_g are trained with TRPO [40] and PPO [41] respectively. The specific hyperparameters used are provided in Appendix C. We use the implementations of TRPO and PPO provided in the stable-baselines library [17]. For every π_g update, we update the GARAT discriminator D_ϕ once as well. Results here use the losses detailed in Algorithm 1. However, we find that GARAT is just as effective with other divergence measures [11, 13, 51] (Appendix C).

GARAT is compared to GAT [15], RARL [33] adapted for a black-box simulator, and action-noise-envelope (ANE) [21]. π_t and π_s denote policies trained in the target environment and source environment respectively until convergence. We use the best performing hyperparameters for these methods, specified in Appendix C.

Figure 3 shows that, in most of the domains, GARAT with just a few thousand transitions from the target environment facilitates transfer of policies that perform on par with policies trained directly in the target environment using 1 million transitions. GARAT also consistently performs better than previous methods on all domains, except *HopperHighFriction*, where most of the methods perform well. The shaded envelope denotes the standard error across 5 experiments with different random seeds for all the methods. Apart from the MuJoCo simulator, we also show successful transfer in the PyBullet simulator [9] using the *Ant* domain. Here the target environment has gravity twice that of the source environment, resulting in purely source environment-trained policies collapsing ineffectually in the target environment. In this relatively high dimensional domain, as well as in *Walker*, we see GARAT still transfers a competent policy while the related methods fail.

In the Minitaur domain [44] we use the high fidelity simulator as our target environment. Here as well, a policy trained in the source environment does not directly transfer well to the target environment [53]. We see in this realistic setting that GARAT learns a policy that obtains more than 80% of the optimal target environment performance with just 1000 target environment transitions while the next best baseline (GAT) obtains at most 50%, requiring ten times more target environment data.

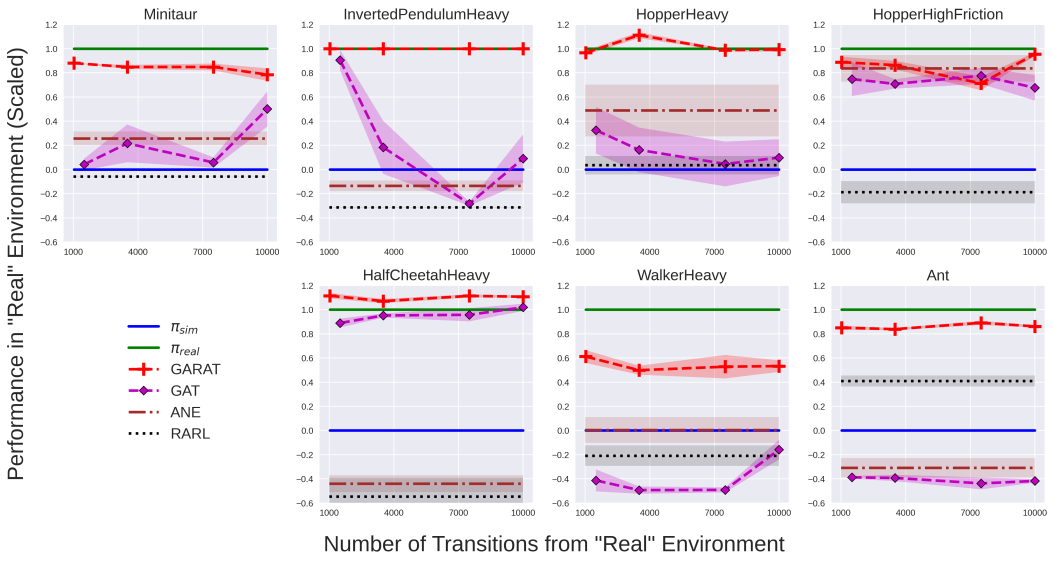


Figure 3: Performance of different techniques evaluated in target environment. Environment return on the y-axis is scaled such that π_t achieves 1 and π_s achieves 0.

7 Conclusion

In this paper, we have shown that grounded action transformation, a particular kind of grounded transfer technique, can be seen as a form of imitation from observation. We use this insight to develop GARAT, an adversarial imitation from observation algorithm for grounded transfer. We hypothesized that such an algorithm would lead to improved grounding of the source environment as well as better transfer compared to related techniques. This hypothesis is validated in Section 6 where we show that GARAT leads to better grounding of the source environment as compared to GAT, and improved transfer to the target environment on various mismatched environment transfers, including the realistic Minitaur domain.

Acknowledgements and Disclosure of Funding

This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (CPS-1739964, IIS-1724157, NRI-1925082), the Office of Naval Research (N00014-18-2243), Future of Life Institute (RFP2-000), Army Research Office (W911NF-19-2-0333), DARPA, Lockheed Martin, General Motors, and Bosch. The views and conclusions contained in this document are those of the authors alone. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

Broader Impact

Reinforcement learning [42] is being considered as an effective tool to train autonomous agents in various important domains like robotics, medicine, etc. A major hurdle to deploying learning agents in these environments is the massive exploration and data requirements [16] to ensure that these agents learn effective policies. Real world interactions and exploration in these situations could be extremely expensive (wear and tear on expensive robots), or dangerous (treating a patient in the medical domain).

Sim-to-real transfer aims to address this hurdle and enables agents to be trained mostly in simulation and then transferred to the real world based on very few interactions. Reducing the requirement for

real world data for autonomous agents might open up the viability for autonomous agents in other fields as well.

Improved sim-to-real transfer will also reduce the pressure for high fidelity simulators, which require significant engineering effort [8, 44]. Simulators are also developed with a task in mind, and are generally not reliable outside their specifications. Sim-to-real transfer might enable simulators that learn to adapt to the task that needs to be performed, a potential direction for future research.

Sim-to-real research needs to be handled carefully, however. Grounded simulators might lead to a false sense of confidence in a policy trained in such a simulator. However, a simulator grounded with real world data will still perform poorly in situations outside the data distribution. As has been noted in the broader field of machine learning [3], out of training distribution situations might lead to unexpected consequences. Simulator grounding must be done carefully in order to guarantee that the grounding is applied over all relevant parts of the environment.

Improved sim-to-real transfer could increase reliance on compute and reduce incentives for sample efficient methods. The field should be careful in not abandoning this thread of research as the increasing cost and impact of computation used by machine learning becomes more apparent [2].

References

[1] Adam Allevato, Elaine Schaertl Short, Mitch Pryor, and Andrea L Thomaz. Tunenet: One-shot residual tuning for system identification and sim-to-real robot task transfer. In *Conference on Robot Learning (CoRL)*, 2019.

[2] Dario Amodei and Danny Hernandez. AI and compute. *openai.com*, May 2018. URL `https://openai.com/blog/ai-and-compute/`.

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

[4] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[5] Paul Bakker and Yasuo Kuniyoshi. Robot see, robot do: An overview of robot imitation. In *AISB96 Workshop on Learning in Robots and Animals*, pages 3–11, 1996.

[6] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, Sergey Levine, and Vincent Vanhoucke. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. *CoRR*, abs/1709.07857, 2017. URL `http://arxiv.org/abs/1709.07857`.

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

[8] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[9] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. *GitHub repository*, 2016.

[10] Alon Farchy, Samuel Barrett, Patrick MacAlpine, and Peter Stone. Humanoid robots learning to walk faster: From the real world to simulation and back. In *Proc. of 12th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2013.

[11] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rkHywl-A-`.

[12] Tanmay Gangwani and Jian Peng. State-only imitation with transition dynamics mismatch. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HJgLLyrYwB`.

[13] Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods, 2019.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[15] Josiah P Hanna and Peter Stone. Grounded action transformation for robot learning in simulation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[16] Josiah Paul Hanna. *Data efficient reinforcement learning with off-policy and simulated data*. PhD thesis, University of Texas at Austin, 2019.

[17] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. `https://github.com/hill-a/stable-baselines`, 2018.

[18] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4565–4573. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6391-generative-adversarial-imitation-learning.pdf`.

[19] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[20] Nick Jakobi. Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive behavior*, 6(2):325–368, 1997.

[21] Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In Federico Morán, Alvaro Moreno, Juan Julián Merelo, and Pablo Chacón, editors, *Advances in Artificial Life*, pages 704–720, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. ISBN 978-3-540-49286-3.

[22] Stephen James, Andrew J. Davison, and Edward Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. *CoRR*, abs/1707.02267, 2017. URL `http://arxiv.org/abs/1707.02267`.

[23] Stephen James, Michael Bloesch, and Andrew J. Davison. Task-embedded control networks for few-shot imitation learning. *CoRR*, abs/1810.03237, 2018. URL `http://arxiv.org/abs/1810.03237`.

[24] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. *CoRR*, abs/1812.07252, 2018. URL `http://arxiv.org/abs/1812.07252`.

[25] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1118–1125. IEEE, 2018.

[26] Jan Matas, Stephen James, and Andrew J. Davison. Sim-to-real reinforcement learning for deformable object manipulation. *CoRR*, abs/1806.07851, 2018. URL `http://arxiv.org/abs/1806.07851`.

[27] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3481–3490, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/mescheder18a.html`.

[28] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 663–670, 2000.

[29] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand, 2019.

[30] Brahma S Pavse, Faraz Torabi, Josiah P Hanna, Garrett Warnell, and Peter Stone. Ridm: Reinforced inverse dynamics modeling for learning from a single observed demonstration. *arXiv preprint arXiv:1906.07372*, 2019.

[31] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1–8. IEEE, 2018.

[32] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[33] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2817–2826. JMLR. org, 2017.

[34] Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

[35] Aravind Rajeswaran, Sarvjeet Ghotra, Sergey Levine, and Balaraman Ravindran. Epopt: Learning robust neural network policies using model ensembles. *CoRR*, abs/1610.01283, 2016. URL `http://arxiv.org/abs/1610.01283`.

[36] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[37] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[38] Fereshteh Sadeghi, Alexander Toshev, Eric Jang, and Sergey Levine. Sim2real view invariant visual servoing by recurrent control. *CoRR*, abs/1712.07642, 2017. URL `http://arxiv.org/abs/1712.07642`.

[39] Stefan Schaal. Learning from demonstration. In *Advances in neural information processing systems*, pages 1040–1046, 1997.

[40] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL `http://arxiv.org/abs/1502.05477`.

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

[42] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[43] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[44] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *CoRR*, abs/1804.10332, 2018. URL `http://arxiv.org/abs/1804.10332`.

[45] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[46] Josh Tobin, Lukas Biewald, Rocky Duan, Marcin Andrychowicz, Ankur Handa, Vikash Kumar, Bob McGrew, Alex Ray, Jonas Schneider, Peter Welinder, et al. Domain randomization and generative models for robotic grasping. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3482–3489. IEEE, 2018.

[47] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[48] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4950–4957, 2018.

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

[50] Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, Aug 2019.

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

[52] Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonathan Hurst, and Michiel van de Panne. Learning locomotion skills for cassie: Iterative design and sim-to-real. In *Proc. Conference on Robot Learning (CORL 2019)*, volume 4, 2019.

[53] Wenhao Yu, C. Karen Liu, and Greg Turk. Policy transfer with strategy optimization. *CoRR*, abs/1810.05751, 2018. URL `http://arxiv.org/abs/1810.05751`.

A Marginal Distributions and Returns

We expand the marginal transition distribution (ρ_s) definition to be more explicit below.

$$\rho_{sim,t}(s,a,s') := \rho_{sim,t}(s)\pi(a|s)P_s(s'|s,a) \tag{7}$$

$$\rho_{sim,t}(s') := \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \rho_{sim,t-1}(s,a,s') \tag{8}$$

$$\rho_s(s,a,s') :=(1-\gamma)\sum_{t=0}^{\infty}\gamma^t\rho_{sim,t}(s,a,s') \tag{9}$$

where $\rho_{sim,0}(s) = \rho_0(s)$ is the starting state distribution. Written in a single equation:

$$\rho_s(s,a,s') = (1-\gamma)\sum_{s_0\in\mathcal{S}}\rho_0(s_0)\sum_{t=0}^{\infty}\gamma^t\sum_{a_t\in\mathcal{A}}\sum_{s_{t+1}\in\mathcal{S}}\pi(a_t|s_t)P(s_{t+1}|s_t,a_t)$$

The expected return can be written more explicitly to show the dependence on the transition function. It then makes the connection to 1 more explicit.

$$\begin{aligned}\mathbb{E}_{\pi,P}\left[G_0\right] &= \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t,s_{t+1})\right] \\ &= \sum_{s_0\in\mathcal{S}}\rho_0(s_0)\sum_{t=0}^{\infty}\gamma^t\sum_{a_t\in\mathcal{A}}\sum_{s_{t+1}\in\mathcal{S}}\pi(a_t|s_t)P(s_{t+1}|s_t,a_t)R(s_t,a_t,s_{t+1})\end{aligned}$$

In the grounded source environment, the action transformer policy π_g transforms the transition function as specified in Section 2.2. Ideally, such a $\pi_g \in \mathbf{\Pi}_g$ exists. We denote the marginal transition distributions in sim and real by ρ_s and ρ_t respectively, and $\rho_g \in \mathcal{P}_g$ for the grounded source environment. The distribution ρ_g relies on $\pi_g \in \mathbf{\Pi}_g$ as follows:

$$\rho_g(s,a,s') = (1-\gamma)\pi(a|s)\sum_{\tilde{a}\in\mathcal{A}}P_s(s'|s,\tilde{a})\pi_g(\tilde{a}|s,a)\sum_{t=0}^{\infty}\gamma^t p(s_t=s|\pi,P_g) \tag{10}$$

The marginal transition distribution of the source environment after action transformation, $\rho_g(s,a,s')$, differs in Equation 7 as follows:

$$\rho_{g,t}(s,a,s') := \rho_{g,t}(s)\pi(a|s)\sum_{\tilde{a}\in\mathcal{A}}\pi_g(\tilde{a}|s,a)P_g(s'|s,\tilde{a}) \tag{11}$$

B Proofs

B.1 Proof of Proposition 4.1

Proposition 4.1. *For a given ρ_g generated by a fixed policy π, P_g is the only transition function whose marginal transition distribution is ρ_g.*

Proof. We prove the above statement by contradiction. Consider two transition functions P_1 and P_2 that have the same marginal distribution ρ_π under the same policy π, but differ in their likelihood for at least one transition (s,a,s').

$$P_1(s'|s,a) \neq P_2(s'|s,a) \tag{12}$$

Let us denote the marginal distributions for P_1 and P_2 under policy π as ρ_1^π and ρ_2^π. Thus, $\rho_1^\pi(s) = \rho_2^\pi(s)\ \forall s \in \mathcal{S}$ and $\rho_1^\pi(s,a,s') = \rho_2^\pi(s,a,s')\forall s,s' \in \mathcal{S}, a \in \mathcal{A}$.

The marginal likelihood of the above transition for both P_1 and P_2 is:

$$\rho_1^\pi(s,a,s') = \sum_{t=0}^{T-1} \rho_1^\pi(s)\pi(a|s)P_1(s'|s,a)$$
$$\rho_2^\pi(s,a,s') = \sum_{t=0}^{T-1} \rho_2^\pi(s)\pi(a|s)P_2(s'|s,a)$$

Since the marginal distributions match, and the policy is the same, this leads to the equality:

$$P_1(s'|s,a) = P_2(s'|s,a) \forall s, s' \in \mathcal{S}, a \in \mathcal{A} \tag{13}$$

Equation 13 contradicts Equation 12, proving our claim. □

B.2 Proof of Proposition 4.2

Proposition 4.2. *If $P_t = P_g$, then* $\operatorname{argmax}_{\pi \in \mathbf{\Pi}} \mathbb{E}_{\pi, P_g}[G_0] = \operatorname{argmax}_{\pi \in \mathbf{\Pi}} \mathbb{E}_{\pi, P_t}[G_0]$.

Proof. We overload the notation slightly and refer to ρ_t^π as the marginal transition distribution in the target environment while following agent policy π. Proposition 4.1 still holds under this expanded notation.

From Proposition 4.1, if $P_t = P_g$, we can say that $\rho_t^\pi = \rho_g^\pi \forall \pi \in \mathbf{\Pi}$. From Equation 1, $\mathbb{E}_{\pi,g}[G_0] = \mathbb{E}_{\pi,real}[G_0] \forall \pi \in \mathbf{\Pi}$, and $\operatorname{argmax}_{\pi \in \mathbf{\Pi}} \mathbb{E}_{\pi,g}[G_0] = \operatorname{argmax}_{\pi \in \mathbf{\Pi}} \mathbb{E}_{\pi,real}[G_0]$. □

B.3 Proof of Lemma 4.1

Lemma 4.1. $\overline{\texttt{RL}} \circ \overline{\texttt{ATIRL}}_\psi(P_t)$ *outputs a marginal transition distribution $\overline{\rho}_g$ which is equal to $\tilde{\rho}_g$ induced by* $\texttt{RL} \circ \texttt{ATIRL}_\psi(P_t)$.

Proof. For every $\rho_g \in \mathcal{P}_g$, there exists at least one action transformer policy $\pi_g \in \mathbf{\Pi}_g$, from our definition of $\mathcal{P}_g$. Let $\texttt{RL} \circ \texttt{ATIRL}_\psi(P_t)$ lead to a policy $\tilde{\pi}_g$, with a marginal transition distribution $\tilde{\rho}_g$. The marginal transition distribution induced by $\overline{\texttt{RL}} \circ \overline{\texttt{ATIRL}}_\psi(P_t)$ is $\overline{\rho}_g$.

We need to prove that $\tilde{\rho}_g = \overline{\rho}_g$, and we do so by contradiction. We assume that $\tilde{\rho}_g \neq \overline{\rho}_g$. For this inequality to be true, the marginal transition distribution of the result of $\texttt{RL}(\tilde{c})$ must be different than the result of $\overline{\texttt{RL}}(\overline{c})$, or the cost functions $\tilde{c}$ and $\overline{c}$ must be different.

Let us compare the RL procedures first. Assume that $\tilde{c} = \overline{c}$.

$$\begin{aligned}
\texttt{RL}(\tilde{c}) &= \operatorname*{argmin}_{\pi} \mathbb{E}_{\rho_g}\left[\tilde{c}(s,a,s')\right] \\
&= \operatorname*{argmin}_{\rho_g} \mathbb{E}_{\rho_g}\left[\tilde{c}(s,a,s')\right] \quad \texttt{...(surjective mapping)} \\
&= \overline{\texttt{RL}}(\overline{c})(\tilde{\texttt{c}} = \overline{\texttt{c}})
\end{aligned}$$

which leads to a contradiction.

Now let's consider the cost functions presented by $\texttt{ATIRL}_\psi(P_t)$ and $\overline{\texttt{ATIRL}}_\psi(P_t)$. Since $\texttt{RL}(\tilde{c})$ and $\overline{\texttt{RL}}(\overline{c})$ lead to the same marginal transition distributions, for the inequality we assumed at the beginning of this proof to be true, $\texttt{ATIRL}_\psi(P_t)$ and $\overline{\texttt{ATIRL}}_\psi(P_t)$ must return different cost functions.

$$
\begin{aligned}
\texttt{ATIRL}_\psi(P_t) &= \operatorname*{argmax}_{c\in\mathcal{C}} -\psi(c) + \left(\min_{\pi_g} \mathbb{E}_{P_g}[c(s,a,s')]\right) - \mathbb{E}_{P_t}[c(s,a,s')] \\
&= \operatorname*{argmax}_{c\in\mathcal{C}} -\psi(c) + \left(\min_{\pi_g} \sum_{s,a,s'} \rho_g(s,a,s')c(s,a,s')\right) - \\
&\sum_{s,a,s'} \rho_t(s,a,s')c(s,a,s') \\
&= \operatorname*{argmax}_{c\in\mathcal{C}} -\psi(c) + \left(\min_{\rho_g} \sum_{s,a,s'} \rho_g(s,a,s')c(s,a,s')\right) - \\
&\sum_{s,a,s'} \rho_t(s,a,s')c(s,a,s') \\
&= \overline{\texttt{ATIRL}}_\psi(P_t)
\end{aligned}
$$

which leads to another contradiction. Therefore, we can say that $\overline{\rho}_g = \rho_{\bar{g}}$. □

B.4 Proof of Lemma 4.2

We prove convexity under a particular agent policy π but across AT policies $\pi_g \in \mathbf{\Pi}_g$

Lemma B.1. *$\mathcal{P}_g$ is compact and convex.*

Proof. We first prove convexity of $\rho_{\mathbf{\Pi}_g,t}$ for $\pi_g \in \mathbf{\Pi}_g$ and $0 \le t < \infty$, by means of induction.

Base case: $\lambda\rho_{at_1,0} + (1-\lambda)\rho_{at_2,0} \in \rho_{\mathbf{\Pi}_g,0}$, for $0 \le \lambda \le 1$.

$$
\begin{aligned}
\lambda\rho_{at_1,0}(s,a,s') + (1-\lambda)\rho_{at_2,0}(s,a,s') &= \lambda\rho_0(s)\pi(a|s)\sum_{\tilde{a}\in\mathcal{A}} \pi_{at_1}(\tilde{a}|s,a)P_s(s'|s,\tilde{a}) \\
&+ (1-\lambda)\rho_0(s)\pi(a|s)\sum_{\tilde{a}\in\mathcal{A}} \pi_{at_2}(\tilde{a}|s,a)P_s(s'|s,\tilde{a}) \\
&= \rho_0(s)\pi(a|s)\sum_{\tilde{a}\in\mathcal{A}} \left(\lambda\pi_{at_1}(\tilde{a}|s,a) + (1-\lambda\pi_{at_2}(\tilde{a}|s,a))\right) P_s(s'|s,\tilde{a})
\end{aligned}
$$

$\mathbf{\Pi}_g$ is convex and hence $\rho_0(s)\pi(a|s)\sum_{\tilde{a}\in\mathcal{A}} \left(\lambda\pi_{at_1}(\tilde{a}|s,a) + (1-\lambda\pi_{at_2}(\tilde{a}|s,a))\right) P_s(s'|s,\tilde{a})$ is a valid distribution, meaning $\rho_{\mathbf{\Pi}_g,0}$ is convex.

Induction Step: If $\rho_{\mathbf{\Pi}_g,t-1}$ is convex, $\rho_{\mathbf{\Pi}_g,t}$ is convex.

If $\rho_{\mathbf{\Pi}_g,t-1}$ is convex, $\lambda\rho_{at_1,t}(s) + (1-\lambda)\rho_{at_2,t}(s)$ is a valid distribution. This is true simply by summing the distribution at time $t-1$ over states and actions.

$$
\begin{aligned}
\lambda\rho_{at_1,t}(s,a,s') + (1-\lambda)\rho_{at_2,t}(s,a,s') &= \lambda\rho_{at_1,t}(s)\pi(a|s)\sum_{\tilde{a}\in\mathcal{A}} \pi_{at_1}(\tilde{a}|s,a)P_s(s'|s,\tilde{a}) \\
&+ (1-\lambda)\rho_{at_2,t}(s)\pi(a|s)\sum_{\tilde{a}\in\mathcal{A}} \pi_{at_2}(\tilde{a}|s,a)P_s(s'|s,\tilde{a}) \\
&= \left(\lambda\rho_{at_1,t}(s) + (1-\lambda)\rho_{at_2,t}(s)\right)\pi(a|s) \\
&\sum_{\tilde{a}\in\mathcal{A}} \left(\lambda\pi_{at_1}(\tilde{a}|s,a) + (1-\lambda\pi_{at_2}(\tilde{a}|s,a))\right) P_s(s'|s,\tilde{a})
\end{aligned}
$$

$\lambda\rho^{\pi}_{at_1,t}(s) + (1-\lambda)\rho^{\pi}_{at_1,t}(s)$ is a valid distribution, and $\mathbf{\Pi}_g$ is convex. This proves that the transition distribution at each time step is convex. The normalized discounted sum of convex sets (Equation 9) is also convex. Since the exponential discounting factor $\gamma \in [0,1)$, the sum is bounded as well. □

We now prove Lemma 4.2.

Lemma 4.2. $\overline{\texttt{RL}} \circ \overline{\texttt{ATIRL}}_\psi(P_t) = \text{argmin}_{\rho_g \in \mathcal{P}_g} \psi^*(\rho_g - \rho_t)$.

Proof of Lemma 4.2. Let $\overline{c} = \overline{\texttt{ATIRL}}(P_t)$, $\overline{\rho}^g = \overline{\texttt{RL}}(\overline{c}) = \overline{\texttt{RL}} \circ \overline{\texttt{ATIRL}}(P_t)$ and

$$\hat{\rho}_g = \underset{\rho_g}{\text{argmin}}\, \psi^*(\rho_g - \rho_t) = \underset{\rho_g}{\text{argmin}} \max_c -\psi(c) + \sum_{s,a,s'} (\rho_g(s,a,s') - \rho_t(s,a,s'))c(s,a,s') \tag{14}$$

where $\psi^* : \mathcal{C}^* \longmapsto \bar{\mathbb{R}}$ is the convex conjugate of ψ, defined as $\psi^*(c^*) := \sup_{c \in \mathcal{C}} \langle c^*, c \rangle - \psi(c)$. Applying the above definition to the rightmost term in the above equation gives us the middle term.

We now argue that $\overline{\rho}_g = \hat{\rho}_g$ which are the two sides of the equation we want to prove. Let us consider loss function $L : \mathcal{P}_g \times \mathbb{R}^{\mathcal{S} \times \mathcal{A} \times \mathcal{S}} \longmapsto \mathbb{R}$ to be

$$L(\rho_g, c) = -\psi(c) + \sum_{s,a,s'} (\rho_g(s,a,s') - \rho_t(s,a,s'))c(s,a,s') \tag{15}$$

We can then pose the above formulations as:

$$\hat{\rho}_g \in \underset{\rho_g \in \mathcal{P}_g}{\text{argmin}} \max_c L(\rho_g, c) \tag{16}$$

$$\overline{c} \in \underset{c}{\text{argmax}} \min_{\rho_g \in \mathcal{P}_g} L(\rho_g, c) \tag{17}$$

$$\overline{\rho}_g \in \underset{\rho_g \in \mathcal{P}_g}{\text{argmin}} L(\rho_g, \overline{c}) \tag{18}$$

$\mathcal{P}_g$ is compact and convex (by Lemma B.1) and $\mathbb{R}^{\mathcal{S} \times \mathcal{A} \times \mathcal{S}}$ is convex. $L(\cdot, c)$ is convex over all c and $L(\rho_g, \cdot)$ is concave over all ρ_g. Therefore, based on minimax duality:

$$\min_{\rho_g \in \mathcal{P}_g} \max_c L(\rho_g, c) = \max_c \min_{\rho_g \in \mathcal{P}_g} L(\rho_g, c) \tag{19}$$

From Equations 16 and 17, $(\hat{\rho}_g, \overline{c})$ is a saddle point of L, implying $\hat{\rho}_g = \text{argmin}_{\rho_g \in \mathcal{P}_g} L(\rho_g, \overline{c})$ and so $\overline{\rho}_g = \hat{\rho}_g$.

□

B.5 Proof of Lemma 4.3

Lemma 4.3. *The marginal transition distribution of* $\text{argmin}_{\pi_g} \psi^*(\rho_g - \rho_t)$ *is equal to* $\text{argmin}_{\rho_g \in \mathcal{P}_g} \psi^*(\rho_g - \rho_t)$.

Proof. The proof of equivalence here is simply to prove that optimizing over π_g is the same as optimizing over ρ_g. From Equation 10 and from the fact that agent policy π and source environment transition function P_s are fixed, we can say that the only way to optimize ρ_g is to optimize π_g, which leads to the above equivalence. □

C Experimental Details

To collect expert trajectories from the target environment, we rollout the stochastic initial policy trained in sim for 1 million timesteps, on the target environment. This dataset serves as the expert dataset during the imitation learning step of GARAT. At each GAN iteration, we sample a batch of data from the grounded source environment and expert dataset and update the discriminator. Similarly, we rollout the action transformer policy in its environment and update π_g. We perform 50 such GAN

Name	Value
Hidden Layers	2
Hidden layer size	64
timesteps per batch	5000
max KL constraint	0.01
λ	0.97
γ	0.995
learning rate	0.0004
cg damping	0.1
cg iters	20
value function step size	0.001
value function iters	5

Table 1: Hyperparameters for the TRPO algorithm used to update the Agent Policy

Name	Value
Hidden Layers	2
Hidden layer size	64
nminibatches	2
Num epochs	1
λ	0.95
γ	0.99
clipping ratio	0.1
time steps	5000
learning rate	0.0003

Table 2: Hyperparameters for the PPO algorithm used to update the Action Transformer Policy

updates to ground the source environment using GARAT. The hyperparameters for the PPO algorithm used to update the action transformer policy is provided in Table 2. The hyperparameters used for the TRPO algorithm to update the agent policy can be found in Table 1.

We implemented different IfO algorithms and noticed that there was no significant difference between these backend algorithms in sim-to-real performance. During the discriminator update step in GAIfO-reverseKL (AIRL), GAIfO and GAIfO-W (WAIL), we use two regularizers in its loss function - L2 regularization of the discriminator's weights and a gradient penalty (GP) term, with a coefficient of 10. Adding the GP term has been shown to be helpful in stabilizing GAN training [27].

In our implementation of the AIRL [11] algorithm, we do not use the special form of the discriminator, described in the paper, because our goal is to simply imitate the expert and does not require recovering the reward function as was the objective of that work. We instead use the approach Ghasemipour et al. [13] use with state-only version of AIRL.

GAT uses a smoothing parameter α, which we set to 0.95 as suggested by Hanna and Stone [15]. RARL has a hyperparameter on the maximum action ratio allowed to the adversary, which measures how much the adversary can disrupt the agent's actions. This hyperparameter is chosen by a coarse grid-search. For each domain, we choose the best result and report the average return over five policies trained with those hyperparameters. We used the official implementation of RARL provided by the authors for the MuJoCo environments. However, since their official code does not readily support PyBullet environments, for the Ant and Minitaur domain, we use our own implementation of RARL, which we reimplemented to the best of our ability. When training a robust policy using Action space Noise Envelope (ANE), we do not know the right amount of noise to inject into the agent's actions. Hence, in our analysis, we perform a sweep across zero mean gaussian noise with multiple standard deviation values and report the highest return achieved in the target environment with the best hyperparameter, averaged across 5 different random seeds.

Environment Name	Property Modified	Default Value	Modified Value
InvertedPendulumHeavy	Pendulum mass	4.89	100.0
HopperHeavy	Torso Mass	3.53	6.0
HopperHighFriction	Foot Friction	2.0	2.2
HalfCheetahHeavy	Total Mass	14	20
WalkerHeavy	Torso Mass	3.534	10.0
Ant	Gravity	-4.91	-9.81
Minitaur [44]	Torque vs. Current	linear	non-linear

Table 3: Details of the Modified target environments for benchmarking GARAT against other black-box transfer algorithms.

C.1 Modified environments

We evaluate GARAT against several algorithms in the domains shown in Figure 3. Table 3 shows the source environment along with the specific properties of the environment/agent modified. We modified the values such that a policy trained in the sim environment is unable to achieve similar returns in the modified environment. By modifying an environment, we incur the risk that the environment may become too hard for the agent to solve. We ensure this is not the case by training a policy π_t directly in the target environment and verifying that it solves the task.

C.2 Source Environment Grounding Experimental Details

In Section 6.1, we show results which validate our hypothesis that GARAT learns an action transformation policy which grounds the source environment better than GAT. Here we detail our experiments for Figure 1.

In Figure 1a, we plot the average error in transitions in source environments grounded with GARAT and GAT with different amounts of target environment data, collected by deploying π in the target environment. The per step transition error is calculated by resetting the source environment state to states seen in the target environment, taking the same action, and then measuring the error in the L2-norm with respect to target environment transitions. Figure 1a shows that with a single trajectory from the target environment, GARAT learns an action transformation that has similar average error in transitions compared to GAT with 100 trajectories of target environment data to learn from.

In Figure 1b, we compare GARAT and GAT more qualitatively. We deploy the agent policy π from the same start state in the target environment, the source environment, GAT-grounded source environment, and GARAT-grounded source environment. Their resultant trajectories in one of the domain features (angular position of the pendulum) is plotted in Figure 1b. The trajectories in GARAT-grounded source

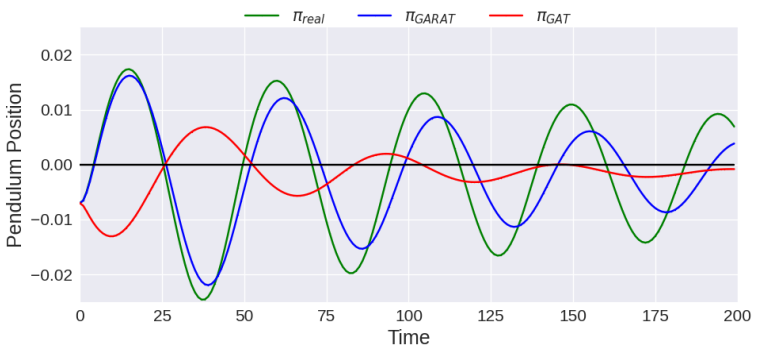


Figure 4: Policies trained in target environment, GAT-grounded source environment, and GARAT-grounded source environment deployed in the target environment from the same starting state

environment keeps close to the target environment, which neither the ungrounded source environment nor the GAT-grounded source environment manage. The trajectory in the GAT-grounded source environment can be seen close to the one in the target environment initially, but since it disregards the sequential nature of the problem, the compounding errors cause the episode to terminate prematurely.

An additional experiment we conducted was to compare the policies trained in the target environment, GAT-grounded source environment and GARAT-grounded source environment. This comparison is done by deploying them in the target environment from the same initial state. As we can see in Figure 4, the policies trained in the target environment and the GARAT-grounded source environment behave similarly, while the one trained in the GAT-grounded source environment acts differently. This comparison is another qualitative one. How well these policies perform in w.r.t. the task at hand is explored in detail in Section 6.2.