Adversarially Robust Policy Learning: Active Construction of Physically-Plausible Perturbations

Ajay Mandlekar*[1], Yuke Zhu*[1], Animesh Garg*[1], Li Fei-Fei[1], Silvio Savarese[1]

***Abstract*— Policy search methods in reinforcement learning have demonstrated success in scaling up to larger problems beyond toy examples. However, deploying these methods on real robots remains challenging due to the large sample complexity required during learning and their vulnerability to malicious intervention. We introduce Adversarially Robust Policy Learning (ARPL), an algorithm that leverages active computation of physically-plausible adversarial examples during training to enable robust policy learning in the source domain and robust performance under both random and adversarial input perturbations. We evaluate ARPL on four continuous control tasks and show superior resilience to changes in physical environment dynamics parameters and environment state as compared to state-of-the-art robust policy learning methods. Code, data, and additional experimental results are available at: `stanfordvl.github.io/ARPL`**

I. Introduction

Renewed research focus on policy learning methods in reinforcement learning have enabled autonomy in many problems considered difficult until recently, such as video games with visual input [24], the deterministic tree search problem in Go [33], robotic manipulation skills [19], and locomotion tasks [20].

Imagine an autonomous robot making a delivery. Although an all-out malicious attack can take down the robot, the robot might also be vulnerable to more subtle adversarial attacks. For example, a smart attacker could create a *man-in-the-middle* style attack. They could make a small, imperceptible change to the policy to get the parcel delivered to himself. While this is a hypothetical scenario, with increasingly pervasive autonomy in both personal and public spaces, it is a real threat.

As we move towards deploying learned controllers on physical systems around us, robust performance is not only a desired property but also a design requirement to ensure the safety of both users and the system itself. Research has shown that a spectrum of machine learning models, including RL algorithms, are vulnerable to malicious attacks [2, 3, 5, 15]. Recent works have studied the existence of adversarial examples [10, 28, 34]; and showed that such instances are not only easy to construct but are also effective against different machine learning models trained for the same task.

While successful, policy search algorithms, such as variants of Q-Learning [24] and Policy Gradient methods [20, 32], are highly data intensive due to the large amount of experience needed to train the agent. Furthermore, complex non-linear function approximators such as deep neural networks can worsen the data dependence. Hence, a naïve approach that utilizes joint training over an ensemble of domains to achieve robustness can quickly become intractable.

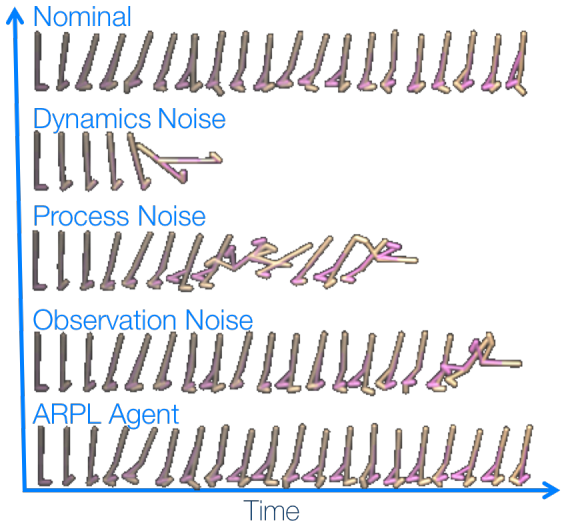

Fig. 1: A `walker-2d` agent trained on a fixed set of parameters would be sensitive to noises in dynamics, process, and observation. We propose a robust training method, ARPL, based on adversarial examples to enhance the robustness of our agent against uncertainty.

Moreover, model-free algorithms require black-box access to a simulator for training. This is often a limitation of policy learning for physical systems. A natural solution to this problem is to train on a source simulator that resembles the target domain with a bounded model mismatch. Although this is a reasonable approach, systematic discrepancies between the source and the target models can result in controller instability. When the target domain is a complex dynamical system, policy learning methods that use gradient descent based optimization procedures often yield marginally stable solutions that are not robust to modeling errors and may fail on a target with a minor model mismatch.

This paper is a step towards addressing the problem of adversarial attacks against autonomous agents in physical domains and the problem of model mismatch when training policies in a simulated domain but deploying the learned policies in a physical target domain. The key intuition of this study is that – *adversarial examples can be used to actively choose perturbations during training in the source domain. This procedure exhibits robustness to both modeling errors and adversarial perturbations on the target domain.* In particular, we explore training in simulated continuous control

* These authors contributed equally to the paper
[1] All authors are affiliated to AI Lab, Stanford University, CA US; {amandlek, yukez, animesh.garg, feifeili, ssilvio}@stanford.edu

tasks for evaluation across varied simulated target domains. We analyze the effect of perturbations on the performance via three kinds of modeling errors: dynamics noise, process noise, and observation noise.

Summary of Contributions:

1) We demonstrate that Deep RL policies are susceptible to both adversarial perturbations and model-mismatch errors.
2) We propose a method to synthesize physically-plausible adversarial perturbations for a white-box model. Further, we present Adversarially Robust Policy Learning (ARPL), a method to use actively chosen adversarial perturbations for robust policy training.
3) We also relate the distance to uncontrollability in simple dynamical systems to control inputs, hence providing a potential justification for ARPL leading to improved robustness in more complex environments where analytic calculations are impractical.
4) We extensively evaluate our method in 4 simulated environments and observe that training with ARPL leads to improved resistance to changes in observations and system dynamics.

II. Related Work and Background

A. Deep Reinforcement Learning

Recent interest in reinforcement learning (RL) has resulted in the resurgence of many classical algorithms that now use neural network function approximators, enabling problems with larger state and action spaces [20, 23, 24]. However, these methods are often sample-inefficient, requiring extensive interaction with the environment. Policy rollouts in target domains that involve real world robot control are expensive. As a result, these methods are used primarily in simulated domains. Despite recent results on robot RL [19, 31], the data requirement for these studies has been very large, prompting cynicism towards the use of these methods in real world tasks with dynamic motor skills.

A common approach to circumvent the sample complexity problem of model-free methods is to use model-based RL [16]. Simulated models approximate the target real-world domain and provide a computationally cheap way to learn policies. However, the main challenge in a source to target transfer is the systematic discrepancy between the two domains [35].

B. Transfer in Reinforcement Learning

RL algorithms that can achieve transfer across domains despite modeling errors are highly desirable due to the inherent differences between simulators and real-world tasks. Consequently, the development of these algorithms is an important and active area of current research. The transfer problem has been examined from different perspectives, such as changing dynamics [13, 29, 36], varying targets [39], and multiple tasks [7, 30]. Taylor et al. provide an excellent treatise on the transfer learning problem [35]. Many approaches focused on reducing the number of rollouts performed on a physical robot by alternating between policy improvement in simulation and physical rollouts [1, 18]. After each physical rollout, a time-dependent term is added to the dynamics to account for unmodeled error. However, this approach is susceptible to failure due to potential robustness issues with the initial transfer. There are no guarantees when the policy is deployed for the first few physical rollouts, and the system could sustain or cause damage before the online learning model converges.

In our work, we show that using actively chosen perturbations of the environment dynamics and observations can result in a more robust policy in the target domain.

C. Adversarial Examples in Supervised Learning

Another concern that is increasingly taking center-stage is resilience and robustness of the learned policies when deployed on critical systems. Machine learning researchers have been exploring the effect of adversarial attacks in general machine learning models [2] and investigating both the robustness and security of the models. Sequential decision making is inherently vulnerable because of the ability of an adversary to intervene through changing the observations or the underlying dynamics in a system [5]. Szegedy et al. [34] introduced the notion of using visually-imperceptible perturbations on images at test-time to cause misclassification in Deep Neural Network (DNN) models used in computer vision applications. This has kickstarted research in both the detection and synthesis of adversarial examples [10, 17, 25, 28], and efforts to safeguard against such malicious attacks [8, 12, 26]. It has since been discovered that adversarial inputs for computer vision models can be computed with minimal compute effort [10], can be applied to physical print-outs of pictures [17], and can be found almost universally for any given machine learning model [25]. Furthermore, predesigned adversarial attacks to reliably quantify the robustness of these classifiers have been explored in [28] and [26].

Additionally, it is worth noting that a majority of the studies in adversarial perturbations have been for supervised learning models. Recent works by Behzadan et al. [3] and Huang et al. [15] have illustrated the existence of perturbations in RL. The study in [3] shows that perturbations can be constructed to prevent training convergence, and Huang et al. [15] demonstrate the ability of an adversary to interfere with the policy operation during test time.

D. Robust and Risk-Sensitive Control

Robust control with respect to modeling errors has been widely studied in control theory. An excellent overview of methods is provided in the book by Green & Limebeer [11]. In the problem of robust transfer, we are interested in parametric uncertainty between source and target models. Nilim et al. [27] and Mastin et al. [22] have produced bounds on the performance of transfer in the presence of bounded disturbances in dynamics. Risk-sensitive and safe RL methods have been proposed to handle uncertainty while enabling safety, as reviewed in [9]. These methods model belief priors over the target domain and preserve safety guarantees similar to robust control. However scaling both robust control methods and risk-sensitive RL methods beyond very simple examples has been a challenge.

Recent studies on robust policy learning for transfer across domains have adapted ideas from robust control and risk-sensitive RL to propose simplified sampling-based methods for training. In particular, Rajeswaran et al. [29] propose a method of sampling dynamics parameters over a prior during training to improve policy robustness to a similar but unseen target setting. Further, Yu et al. [38] propose a similar robust policy learning method through adding parameters to the system state and with the additional option of performing an online estimate of dynamics.

E. Controllability and Stability

Controllability and stability analysis forms an important part of analytical controller design. Mailybaev et al. [21] study how far a system is from the nearest uncontrollable state. He [14] examines the problem of calculating the distance to uncontrollability and the distance to instability for a given controller as a function of the perturbation to the system. Boyce [6] proposes a method to calculate this distance through linear matrix inequalities. In Section V, we analyze how the policies resulting from ARPL increase the distance to uncontrollability as compared to a vanilla policy under the same state-action constraints.

This work is, to the best of our knowledge, one of the first studies to examine physically plausible perturbations to observations and systematic shifts in environment dynamics that result in deteriorated policy performance. Furthermore, we propose an algorithm to leverage adversarial perturbations to train policies that are robust to a wide range of perturbations in dynamics and observations.

III. Preliminaries

A. Reinforcement Learning: Policy Optimization

Consider an infinite horizon discounted Markov Decision Process ($\mathcal{M}$) defined as a tuple of the form $\mathcal{M}: \langle \mathcal{S}, \mathcal{A}, \mathcal{T}(\cdot,\cdot,\cdot), R(\cdot,\cdot), \gamma \rangle$. Here $\mathcal{S}$ and $\mathcal{A}$ represent the state and action space for the agent. $\mathcal{T}(\cdot,\cdot,\cdot)$ is an transition function $\mathcal{T}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ that captures the state transition dynamics in the environment. $R: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. is the reward function that maps a state-action tuple to a scalar, and $\gamma \in [0,1]$ is the discount factor to allow devaluation of future reward. The goal is to find a policy $\pi: \mathcal{S} \to \mathcal{A}$, the maximizes the expected cumulative reward $\eta(\pi)$ over the choice of policy:

$$\pi^*(s) = \underset{\pi}{\operatorname{argmax}}\, \eta(\pi) = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

Now to perform policy gradient optimization, we can define the gradient of the cumulative reward with respect to current policy parameters θ over an observed trajectory of state-action pairs τ and the corresponding return over the trajectory $R(\tau)$,

$$\begin{aligned} \eta(\pi) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] = \sum_{\tau} \mathcal{T}(\tau;\theta) R(\tau), \\ \nabla_\theta \eta(\pi) &= \mathbb{E}_{\tau \sim \mathcal{T}}\left[\nabla_\theta \mathcal{T}(\tau;\theta) R(\tau)\right] \end{aligned}$$

We use sampled policy gradient to perform gradient-based parameter update over the function class of policies π parametrized through θ. In this work, we train all our agents using Trust Region Policy Optimization [32]. TRPO is an on-policy batch learning algorithm which uses a constrained policy gradient update to perform policy improvement, and is state-of-the-art for continuous control.

B. Adversarial Perturbation in Deep Neural Networks

Szegedy et al. [34] discovered that deep learning models are highly vulnerable to adversarial examples. Furthermore, these adversarial examples exhibit a remarkable generalization property - a variety of models with different parametrizations are often fooled by the same adversarial example, even when these models are trained on different subsets of the training data.

Methods of creating adversarial examples rely upon maximizing the prediction error, such that the perturbation is bounded. In image classification, a common objective is to perturb an image by a small, imperceptible amount such that its prediction class changes. In reinforcement learning, the objective is to misguide the policy to output incorrect actions, while minimizing the change in either the input state, the dynamics model, or the observation model. One of the most prevalent methods for generating adversarial examples is the Fast Gradient Sign Method (FGSM) by Goodfellow et al. [10]. FGSM offers computationally efficiency at the cost of a slight decrease in attack success rate.

FGSM focuses on adversarial perturbations of an image input where the change in each pixel is limited by ε and makes a linear approximation of the Deep Neural Network, resulting in the following perturbation

$$\delta = \varepsilon \operatorname{sign}\big(\nabla_s \eta(\pi_\theta(s))\big) \qquad \texttt{(FGSM)} \tag{1}$$

where η is a loss function over the policy π, which is parametrized by parameters θ.

IV. Adversarially Robust Policy Learning

This section will first describe the various threat models that can be used to generate perturbations, and then detail the ARPL algorithm that leverages these perturbations during policy training for robust performance.

A. Physically Plausible Threat Model

Consider a physical dynamical system:

$$\begin{aligned} s_{t+1} &= f(s_t, a_t; \mu) + \nu \qquad \texttt{(Dynamics)} \\ o_t &= g(s_t) + \omega \qquad \texttt{(Observation)} \end{aligned} \tag{2}$$

where the `Dynamics` model updates the state s with action a according to a state transition function f, parametrized by dynamics parameters μ (such as mass), and process noise ν. The `Observation` model maps the current state s to the observed state o with the observation function g and observation noise ω.

To generate a perturbation on a state s we use an isometrically scaled version of the full gradient

$$\delta = \varepsilon\, \nabla_s \eta(\pi_\theta(s)) \qquad \texttt{(ARPL)} \tag{3}$$

where η is a loss function over the policy π, which is parametrized by parameters θ, and ε is a scalar that corresponds to the perturbation strength. Notice that we use the full gradient to generate the perturbation instead of using the popular FGSM. FGSM is primarily designed for images where the state is high dimensional and the dimensions are approximately IID. However, for dynamical models, the dimensions in the state space correspond to different physical quantities; hence a fixed unit step size can result in scaling issues.

Type of Perturbation: A malicious adversary can change any of the three quantities μ, ν, or ω. A change in μ can be equated to dynamics noise, i.e. uncertainty in physical parameters such as mass, friction, and inertia, while ν and ω correspond to direct perturbations of state and observation. This contrasts with prior work in [3, 15] that only examine perturbations to the current state in image space, i.e. ν, which is often not physically plausible.

We perform perturbations that correspond to process noise ν by adding gradient based perturbation to the state of the system. Similarly, we add observation noise ω by changing the observation while preserving the system state. Adversarial perturbation on dynamics noise through agent parameters μ requires state augmentation $\bar{s} = [s, \mu]^T$, and only the latter component of the gradient is used $\nabla_{\bar{s}} = [0, \nabla_{\mu}]$.

We maintain physical plausibility of all perturbations through the projection of the perturbed state to its respective domain, i.e. the state space for s and bounded variation in $\mu \in [0.5\mu_0, 1.5\mu_0]$, where μ_0 is nominal (source) dynamics.

Modes of Perturbation: We build two threat modes: *Adversarial* and *Random*. For noise in states (ν) and observation (ω), adversarial states are calculated using δ, while random perturbations are uniformly sampled from $[-\delta, \delta]$. For dynamics noise, μ is set to be a uniform sample in $[0.5\mu_0, 1.5\mu_0]$ at each time iteration. For adversarial dynamics noise, we first get a random sample as before, then add a gradient δ evaluated at $\mu_{adv} \sim U(0.5\mu_0, 1.5\mu_0)$.

Frequency of Perturbation: The parameter $\phi \in [0,1]$ determines the frequency of applying adversarial (or random) updates. At each time step, an update is applied with prob. $Bernoulli(\phi)$. When $\phi = 0$, only the initial time step is perturbed in each episode.

B. Robust Training with Adversarial Perturbations

Policy Optimization methods utilize batch trajectory sampling for gradient estimates as in [32]. ARPL operates by modifying the trajectory rollouts to include trajectory perturbation. This procedure is outlined in Algorithm 1. In the most general setting, at each iteration, a trajectory is rolled out, and an adversarial perturbation is added to the model with probability ϕ at each time step along the trajectory. The exact operation in `add_perturbation` depends on the choice of threat model, and ϕ controls the frequency of perturbation. A policy gradient update is made after rolling out a batch of k trajectories. We used a curriculum learning approach [4] in order to train our agents on increasing ϕ values. This was necessary to help our policies converge - the agent first learns to perform well in the nominal environment, and then perturbations are added with higher frequency as training progresses.

Algorithm 1: Adversarially Robust Policy Learning

Data: hyperparamaters: ϕ, ε, k
Initialize π_0
foreach $i = 0, 1, 2, \ldots,$ *max_iter* **do**
  `// Perform k-Batch Rollout(s) with Adv. Perturb.`
  foreach $t = 1, \ldots, T_k, \forall k$ **do**
    Sample trajectory with policy $\pi(\theta_i)$: $\tau_{ik} = \{s_t, a_t\}_{t=0}^{T_k - 1}$
    Compute adv. perturb. $\delta = \varepsilon\big(\nabla_{s_t} \eta(\pi_{\theta_i})\big)$
    `add_perturbation`(δ) with prob. $Bernoulli(\phi)$
  `// Batch Update to Policy Network`
  $\theta_{i+1} \leftarrow$ `policy_grad`$(\theta_i, \{\tau_{i1}, ..., \tau_{ik}\})$
Result: Robust Policy π

ARPL achieves robustness by adding adversarial model variation in each rollout. However, training on adversarial examples is different from other data augmentation schemes. In supervised models, the data is augmented with *a priori* transformations such as translation and rotation which are expected to occur in the test set. By contrast, adversarial perturbations rely on the online generation of scenarios that not only expose flaws in the ways that the agent conceptualizes its decision function, but also are likely to occur naturally.

V. Robustness as Distance to Uncontrollabilty

A dynamical system is said to be controllable if for every state x (previously called state s), there exists a sequence of control inputs that can transfer the system to state x. Model perturbations can turn a controllable dynamical system into one that is uncontrollable. For a controllable linear system $\dot{x} = Ax + Bu$ with state x and control u (previously called action a), and matrices A and B of appropriate dimension, the distance to uncontrollability d_{uc} can be characterized by the smallest singular value of the matrix $[A - \lambda I, B]$,

$$d_{uc} = \min_{\lambda} \ \sigma_{\min}\big([A - \lambda I, B]\big) \tag{4}$$

Consider an infinite-horizon time-invariant Linear Quadratic Regulator (with standard notation). The form of the optimal controller is $u = -Kx$. Here K is given by $K = R^{-1}(B^T P)$, and P is the solution of a Continuous-time Algebraic Riccati Equation (CARE):

$$A^T P + A - (PB)R^{-1}(B^T P) + Q = 0$$

Proposition 1: Let $(A;B)$ be controllable, $(A;Q)$ be observable, P be a positive definite solution to CARE, with Q positive semi-definite, and R positive definite. Then, we have:

$$d_{uc} \geq \frac{\lambda_{min}(Q)}{||P|| \sqrt{1 + \frac{\lambda_{min}(Q)}{\lambda_{min}(R)}})} \tag{5}$$

where $\lambda_{min}(\cdot)$ denotes the smallest eigenvalue. We omit the proof and refer the reader to Theorem 2 of the analysis in [14], from which the above can be derived.

Proposition 1 implies that the distance to uncontrollability under perturbations to the matrices $(A;B)$ is inversely proportional to the matrix norm of P, and by extension the total control effort. Intuitively, if the control approximately saturated in order to achieve or maintain stability, then the system can be destabilized by small perturbations to the model, and vice-versa. While this result is hard to derive for non-linear systems, we can use this result as a guideline to compare robustness of controllers under the same set of perturbations on the same task by directly comparing the control effort needed to achieve stability. An experiment along this line is presented in Section VI-C.

VI. Experimental Evaluation

A. Experimental Setup

We evaluated our proposed ARPL algorithm on four continuous control tasks – Inverted Pendulum, Half Cheetah, Hopper, and Walker using the MuJoCo physics simulator [37]. These tasks involve complex non-linear dynamics and direct torque control on the actuated joints. Under-actuation and discontinuous contact render these tasks a challenging benchmark for reinforcement learning methods. To understand our agent's robustness with physical plausibility, we use a low-dimensional state representation that captures the joint angles and velocities in these tasks. In such cases, perturbations on the state vectors naturally lead to a new state that is physically realizable in the environments. The objective of the Inverted Pendulum task is to keep a pole upright by applying a force (control) to the base of the pole. The agent keeps accumulating reward while the pole is upright and fails the task if the pole tilts by too much. The objective of the Half Cheetah, Hopper, and Walker tasks is to apply torque control to the joints in order to move right as fast as possible until the body falls over.

We use the state-of-the-art trust region policy optimization (TRPO) method [32] to learn a stochastic policy using neural networks. The policy is parametrized by a Gaussian distribution, where its mean is predicted by a network that takes a state as input and consists of two hidden layers with 64 units per layer, and tanh as the non-linearity, and the standard deviation is an additional learned parameter that is independent of the state. For the loss function η that ARPL uses to generate adversarial perturbations, we use $\eta(\mu_\theta) = \|\mu_\theta\|_2^2$, where μ_θ is the output of the mean network with parameters θ. We used this loss function to create perturbations that would move the state or dynamics such that the policy is encouraged to apply actions with larger norms, which can lead to instability.

As mentioned in Sec. IV-B, we used a curriculum learning approach to train our agents on increasing perturbation frequency (ϕ). All agents were trained for 2000 iterations - the large number of iterations was necessary to guarantee convergence for curriculum learning. We define the curriculum by uniformly increasing ϕ between 0 and ϕ_{max}. For process noise perturbations, we set $\phi_{max} = 0.1$, and for dynamics noise perturbations we set $\phi_{max} = 0.5$. We update the curriculum every 200 iterations. Note that we omitted results on observation noise perturbations due to space constraints. It is likely that observation noise will become much more important in the context of real world robot experiments.

B. Improving Agent Robustness using Adversarial Training

Here we evaluate the effectiveness of our robust training method proposed in Sec. IV-B. For every agent type, we trained 15 agents and selected the agent with the best learning curve. This is necessary since our method also tends to produce agents with poor performance due to the high variance of the training process. Nominal agents were trained with vanilla TRPO, random agents were trained using ARPL with random perturbations, and adversarial agents were trained using ARPL with adversarial perturbations. These results are for two sets of agents - one that were trained on process noise and another that were trained on dynamics noise.

Figures 2(a) and 2(b) show the effect of process noise in the Inverted Pendulum task. As Fig. 2(a) demonstrates, the nominal agent is highly susceptible to process noise, but both the random and adversarial agents are more robust. It is interesting to note that the adversarial agent performs better in the region where it was trained, but the random agent seems to generalize outside of that region. However, Fig. 2(b) shows that the adversarial agent is incredibly robust to random process noise, much more so than the random agent. This is a very promising result since random process noise perturbations are much more likely to be encountered in practice (for example, on a robot, with noise in sensor measurements).

Figures 3(a) and 3(b) show the effect of dynamics noise in the Inverted Pendulum task. We see that the adversarial agent is robust across both adversarial and random dynamics perturbations across all perturbation frequencies, while the random agent is significantly more robust than the nominal agent.

Figures 4(a) and 4(b) show the effect of dynamics noise in the Walker task. We see that the adversarial and random agents are robust across both adversarial and random dynamics perturbations across all perturbation frequencies.

Figures 5(a)-(d) show the agents' performance under different combinations of dynamics configurations. The center of the grid corresponds to the original values of the dynamics parameters. We can see that nominal performance tends to suffer farther from the center of the grid, as expected, while both random and adversarial agents are robust to the changed mass and friction values. In general, the adversarial agents tend to obtain higher reward, but the random agents are still much more robust than the nominal agents. It is not clear whether the use of adversarial training with respect to dynamics noise results in substantial benefits. Additional investigation is necessary.

C. Robustness as Controllability

We also investigated the control effort exerted by the dynamics noise agents on the nominal environments to test our hypothesis from Section V. We define control effort as the mean square control norm exerted by an agent over an

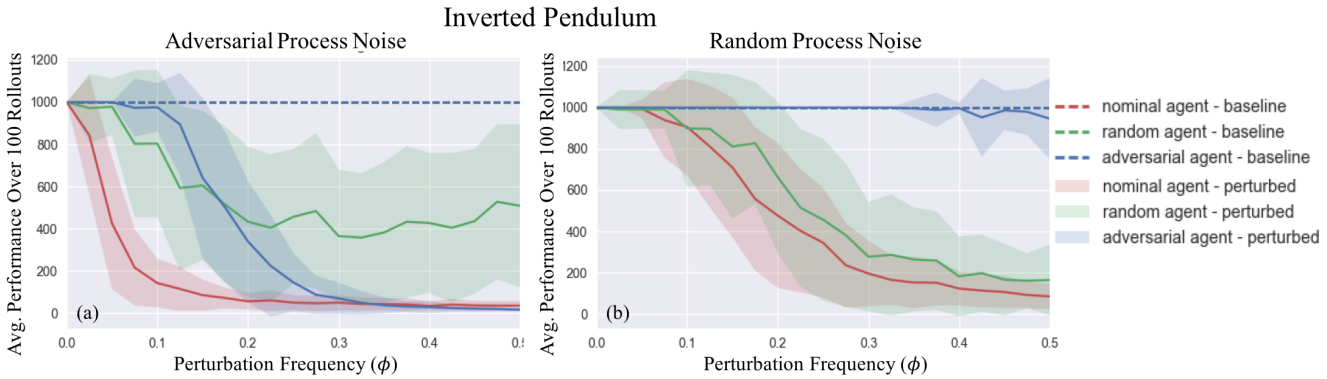

Fig. 2: **Inverted Pendulum** - A comparison of agent performance with respect to Adversarial and Random Process Noise. The baseline performance indicates how each agent performs in an unperturbed environment (all three perform optimally here). Here, $\varepsilon = 0.01$, and we evaluated agent performance as we increased the perturbation frequency. (a) **Adversarial Process Noise**: We note that the adversarial agent does the best in the region where it was trained, but the random agent is more resistant in the higher frequency regime. (b)**Random Process Noise**: We note that the adversarial agent is robust across all perturbation frequencies while the random agent suffers with higher frequency noise.

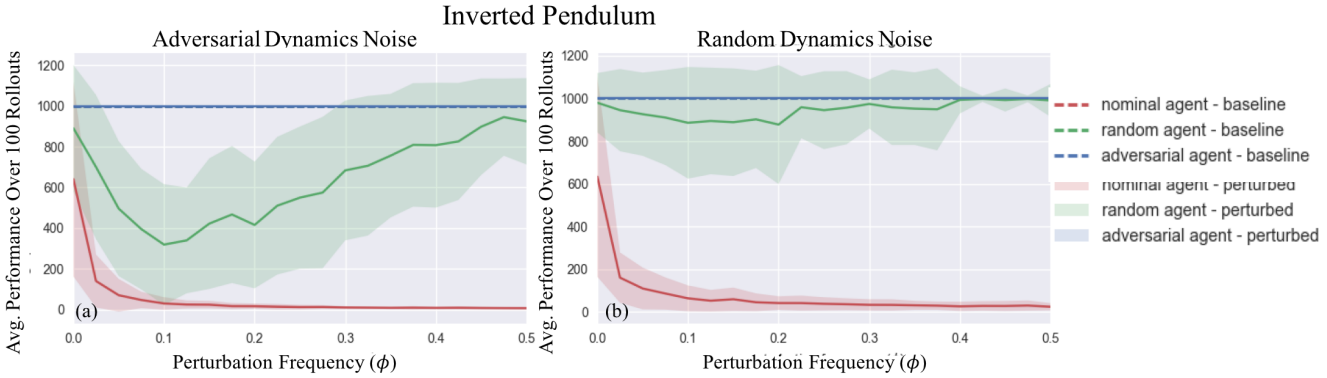

Fig. 3: **Inverted Pendulum** - A comparison of agent performance with respect to Adversarial and Random Dynamics Noise. The baseline performance indicates how each agent performs in an unperturbed environment (all three perform optimally here). Here, $\varepsilon = 10.0$, and we evaluated agent performance as we increased the perturbation frequency. We note that the adversarial agent is robust across all perturbation frequencies both for (a) **Adversarial Dynamics Noise** and (b) **Random Dynamics Noise**.

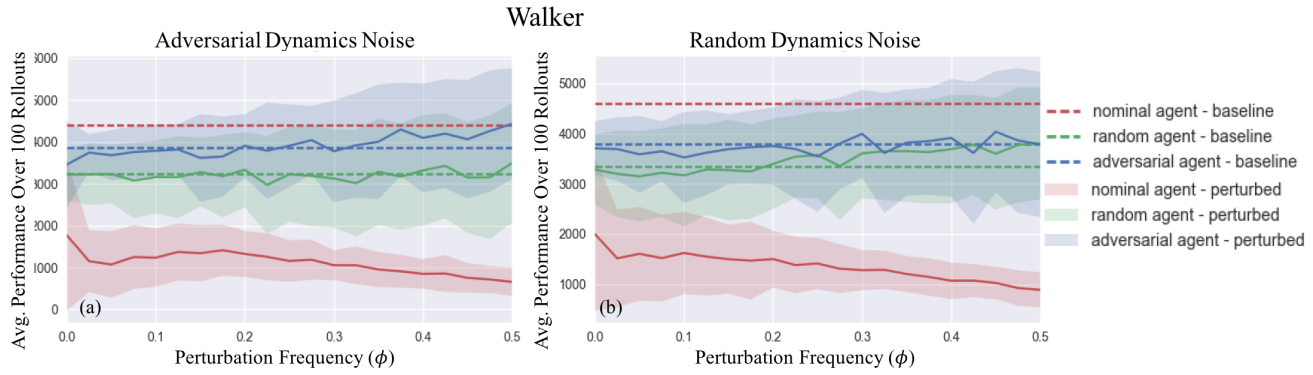

Fig. 4: **Walker** - A comparison of agent performance with respect to Adversarial and Random Dynamics Noise. Here, $\varepsilon = 10.0$, and we evaluated agent performance as we increased the perturbation frequency. The baseline performance indicates how each agent performs in an unperturbed environment. We note that the both adversarial and random agent are robust across all perturbation frequencies and maintain zero noise performance in case of both (a) **Adversarial Dynamics Noise** and (b) **Random Dynamics Noise**.

episode. The results in Table I for the Inverted Pendulum and Walker tasks support our hypothesis that the adversarial agents are more robust because they exert less control effort than the nominal policy. However, the results for the Half Cheetah and Hopper tasks demonstrate that there are other factors besides control effort that determine policy robustness.

VII. Discussion and Future Work

We motivated and presented ARPL, an algorithm for using adversarial example during the training of RL agents to make them more robust to changes in the environment. We trained and evaluated policies on 4 continuous control MuJoCo tasks, and showed that agents trained using vanilla TRPO are vulnerable to changes in the environment state

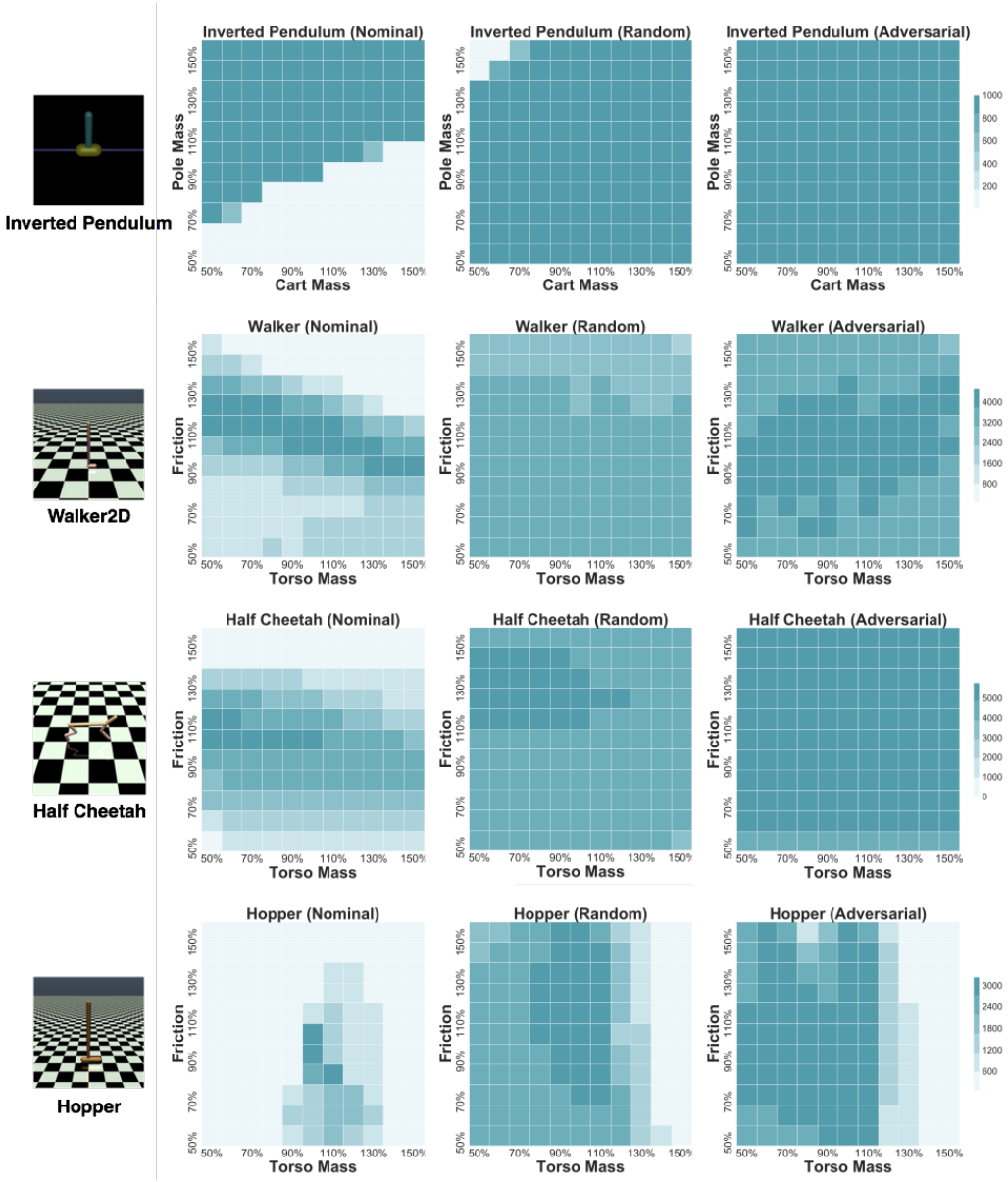

Fig. 5: Policy robustness of dynamics noise agents with respect to various dynamics parameters. Each plot is a heat map that demonstrates the performance of each agent over different environment dynamics. Every grid point corresponds to a particular set of environment dynamics, and darker colors correspond to higher reward. Results were averaged over 100 rollouts. The plots demonstrate that the random and adversarial agents are much more robust to different environment dynamics than the nominal agents.

TABLE I: Comparison of average control effort per rollout across the agents trained with dynamics noise. Results were averaged across 100 rollouts.

Agent Type	Nominal	Random	Adversarial
Inverted Pendulum	1.413	0.028	**0.015**
Walker	15.162	9.247	**8.749**
Half Cheetah	5.613	**4.492**	5.903
Hopper	**2.152**	2.637	4.608

and dynamics. We demonstrated that using ARPL with both random and adversarial dynamics noise leads to policies that are robust with respect to the environment dynamics. We also demonstrated that using ARPL to train with random and adversarial process noise leads to agents that are robust to noise in the environment state.

There are several aspects of this work that we will investigate in the future. First, we want to investigate the use of perturbations with respect to different loss functions over the policy network. This could include using modified policy gradients with respect to the states and using numerical gradients to estimate the reward gradient with respect to the state. We also want to investigate the effect of observation noise further, as well as different forms of random process noise. We want to look into ways to compute dynamics noise perturbations without state augmentation, since this information is not typically available in the real world. This might also lead to more differentiation between random and adversarial dynamics noise agents. Additionally, the agents trained by ARPL demonstrated incredibly high variance in performance (hence why we compared results across the best agents). We want to investigate methods of variance

reduction. We would also like to develop a theoretically sound justification for ARPL. Finally, we want to test an ARPL policy on a robot and investigate the robustness of the policy.

ACKNOWLEDGMENT

This research was performed at the SAIL at Stanford University in affiliation with the Stanford-Toyota AI Center.

Resources: Code, data, and additional experimental results are available at: `stanfordvl.github.io/ARPL`

REFERENCES

[1] P. Abbeel, M. Quigley, and A. Y. Ng, "Using inaccurate models in reinforcement learning", in *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006, pp. 1–8.

[19] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection", *ArXiv preprint arXiv:1603.02199*, 2016.

[2] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?", in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, ACM, 2006, pp. 16–25.

[3] V. Behzadan and A. Munir, "Vulnerability of deep reinforcement learning to policy induction attacks", Jan. 16, 2017. arXiv: 1701.04143v1 [cs.LG].

[4] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning", in *Int'l Conf. on Machine Learning*, ACM, 2009.

[5] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time", in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2013, pp. 387–402.

[6] S. J. Boyce, "The Distance to Uncontrollability via Linear Matrix Inequalities", Master's thesis, Virginia Polytechnic Institute and State University, 2010.

[7] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer", *ArXiv preprint arXiv:1609.07088*, 2016.

[8] J. Gao, B. Wang, and Y. Qi, "Deepmask: Masking dnn models for robustness against adversarial samples", Feb. 22, 2017. arXiv: 1702.06763v1 [cs.LG].

[9] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning", *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples", Dec. 20, 2014. arXiv: 1412.6572v3 [stat.ML].

[11] M. Green and D. J. Limebeer, *Linear robust control*. Courier Corporation, 2012.

[12] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples", Feb. 21, 2017. arXiv: 1702.06280v1 [cs.CR].

[13] J. Harrison*, A. Garg*, B. Ivanovic, Y. Zhu, S. Savarese, L. Fei-Fei, and M. Pavone (* equal contribution), "Adapt: zero-shot adaptive policy transfer for stochastic dynamical systems", 2017. arXiv: 1707.04674 [CS.RO].

[14] C. He, "On the distance to uncontrollability and the distance to instability and their relation to some condition numbers in control", *Numerische Mathematik*, vol. 76, no. 4, 1997.

[15] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies", Feb. 8, 2017. arXiv: 1702.02284v1 [cs.LG].

[16] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey", *The International Journal of Robotics Research*, 2013.

[17] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world", Jul. 8, 2016. arXiv: 1607.02533v4 [cs.CV].

[18] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies", *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[20] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning", *ArXiv preprint arXiv:1509.02971*, 2015.

[21] A. A. Mailybaev, "Uncontrollability for linear autonomous multi-input dynamical systems depending on parameters", *SIAM journal on control and optimization*, 2003.

[22] A. Mastin and P. Jaillet, "Loss bounds for uncertain transition probabilities in markov decision processes", in *Annual Conf. on Decision and Control (CDC)*, IEEE, 2012.

[23] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning", *ICML 2016*, Feb. 4, 2016. arXiv: 1602.01783v2 [cs.LG].

[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning", *Nature*, 2015.

[25] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations", Oct. 26, 2016. arXiv: 1610.08401v2 [cs.CV].

[26] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[27] A. Nilim and L. El Ghaoui, "Robust control of markov decision processes with uncertain transition matrices", *Operations Research*, vol. 53, no. 5, pp. 780–798, 2005.

[28] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples", Feb. 8, 2016. arXiv: 1602.02697v3 [cs.CR].

[29] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, "Epopt: learning robust neural network policies using model ensembles", Oct. 5, 2016. arXiv: 1610.01283v3.

[30] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks", *ArXiv preprint arXiv:1606.04671*, 2016.

[31] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets", *ArXiv preprint arXiv:1610.04286*, 2016.

[32] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust region policy optimization", *ICML*, 2015.

[33] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search", *Nature*, 2016.

[34] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks", Dec. 21, 2013. arXiv: 1312.6199v4 [cs.CV].

[35] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey", *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[36] B. Thananjeyan, A. Garg, S. Krishnan, C. Chen, L. Miller, and K. Goldberg, "Multilateral surgical pattern cutting in 2d orthotropic gauze with deep reinforcement learning policies for tensioning", in *Int'l Conf. on Robotics and Automation (ICRA)*, IEEE, 2017.

[37] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control", in *Int'l Conf. on Intelligent Robots and Systems (IROS)*, IEEE, 2012.

[38] W. Yu, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification", Feb. 8, 2017. arXiv: 1702.02453v2 [cs.LG].

[39] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning", *ArXiv preprint arXiv:1609.05143*, 2016.