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

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

Abstract

Policy search methods in reinforcement learning have demonstrated success in scaling up to larger problem sizes 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 sample-efficient 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.

1. 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 (Mnih et al., 2015), the deterministic tree search problem in Go (Silver et al., 2016), robotic manipulation skills (Levine et al., 2016b), and locomotion tasks (Lillicrap et al., 2015).

Imagine an autonomous robot making a parcel 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 which alters the policy behavior only slightly so as to evade detection but get the parcel delivered to himself at an unintended location resulting in endemic losses. 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 system itself. Machine learning research has shown that a spectrum of models, including RL algorithms, are vulnerable to malicious attacks (Barreno et al., 2006; Biggio et al., 2013; Behzadan & Munir; Huang et al.). Recent works have studied the existence of adversarial examples (Szegedy et al.; Goodfellow et al.; Papernot et al.); and showed that such instances are not only easy to construct but are also effective against different models trained for the same task.

While successful, policy search algorithms, such as variants of Q-Learning (Mnih et al., 2015) and Policy Gradient methods (Lillicrap et al., 2015; Schulman et al., 2015), are highly data intensive. Furthermore, 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.

This paper is a step towards addressing the problem of robustness to malicious attacks while maintaining data efficiency in training. 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 without an increase in data-requirement.* In particular, we explore training in simulated continuous control tasks for evaluation across varied simulated target domains. We analyze the effect of perturbations on the performance via three kinds of modeling errors: model parameter uncertainty, 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.

[1]Department of Computer Science, Stanford University. Correspondence to: Ajay Mandlekar <amandlek@stanford.edu>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 2017. JMLR: W&CP.

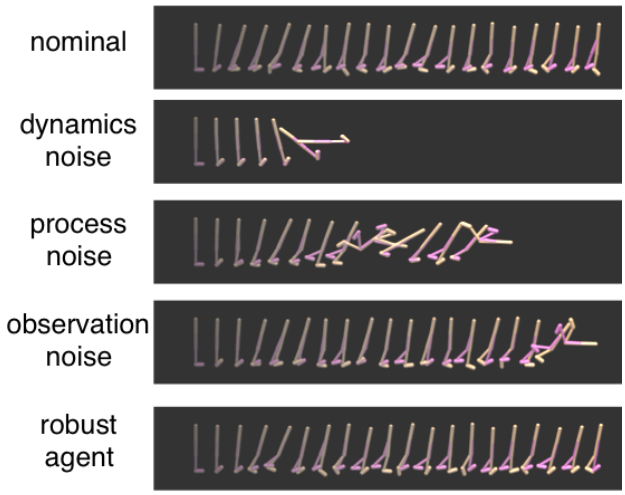


Figure 1. A `walker-2d` agent trained on a fixed set of parameters would be sensitive to noises in dynamics, process, and observation. We propose a robust training method, ARPL, based on adversarial examples to enhance the robustness of our model against uncertainty.

2. We propose a method to synthesize physically-plausible adversarial perturbations for a white-box model. Further, we present Adversarially Robust Policy Learning method to use actively chosen adversarial perturbations for robust policy training.

3. We evaluate our method in 4 simulated environments with many policy training variations in each and observe that training with ARPL results in improving cumulative reward.

2. Related Work

2.1. Deep Reinforcement Learning

Recent interest in reinforcement learning has resulted in the resurgence of many classical algorithms that now use neural network function approximators, enabling problems with larger state and action spaces (Mnih et al.; 2015; Lillicrap et al., 2015). However, these methods are often sample-inefficient and used mainly in simulated domains. Policy rollouts on target domain with real world robot control are expensive and hence require sample efficient policy learning. Despite recent results on robot reinforcement learning (Levine et al., 2016b; Rusu et al., 2016b), the data requirement for these studies has been very large, prompting cynicism towards application to tasks with dynamic motor skills.

A common approach to circumvent the sample complexity problem of model-free methods, is utilizing sample-efficient model-based reinforcement learning approaches (Kober et al., 2013). Source 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 (Taylor & Stone, 2009).

2.2. Transfer in Reinforcement Learning

Transfer in RL algorithms to adapt across variations in modeling errors is both important and an active area of current research. The transfer problem has been examined from different perspectives, such as changing dynamics (Rajeswaran et al.), varying targets (Zhu et al., 2016), and multiple tasks (Devin et al., 2016; Rusu et al., 2016a). Taylor et al. provide an excellent treatise on the transfer learning problem (Taylor & Stone, 2009). A series of approaches focused on reducing the number of rollouts performed on a physical robot by alternating between policy improvement in simulation and physical rollouts (Abbeel et al., 2006; Levine et al., 2016a). After each physical rollout, a time-dependent term is added to the dynamics to account for unmodeled error. This approach, however, does not address robustness in the initial transfer, and the system could sustain or cause damage before the online learning model converges.

In contrast to previous work, we show that using actively chosen perturbations of the environment dynamics and observations noise models can result in a more robust policy in target domain during testing than randomly sampled source domains.

2.3. Adversarial Examples in Supervised Learning

Concurrent with RL, 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 (Barreno et al., 2006) 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 both changing the underlying dynamics or the observations (Biggio et al., 2013). For instance, in an autonomous driving scenario, an adversary may disrupt the controller by adding structured jitter to the camera images or by changing the paint color on a STOP sign.

The notion of minimal perturbation of test-time input, imperceptible to the human eye, to lead to misclassification in Deep Neural Network (DNN) models was first shown for computer vision applications in Szegedy et al. (Szegedy et al.). This has fueled an exciting direction in both detection and synthesis of adversarial examples (Goodfellow et al.; Kurakin et al.; Papernot et al.; Moosavi-Dezfooli et al.), and efforts to safeguard against such malicious at-

tacks (Gao et al.; Grosse et al.; Moosavi-Dezfooli et al., 2016). It has since been discovered that adversarial inputs for computer vision models can be computed with minimal compute effort (Goodfellow et al.), can be applied to physical print-outs of pictures (Kurakin et al.), and can be found almost universally for any given model (Moosavi-Dezfooli et al.). Furthermore, pre-designed adversarial attacks to reliably quantify the robustness of these classifiers have been explored in (Papernot et al.) and (Moosavi-Dezfooli et al., 2016). And finally, researchers have also explored methods to secure models against malicious attacks. Grosse et al. (Grosse et al.) show that adversarial examples are not drawn from the same distribution than the original data, and can thus be detected using statistical tests. Gao et al. (Gao et al.) presents a method that limits the capacity an attacker can use generating adversarial samples and therefore increase the robustness against such inputs.

2.4. Robust and Adversarial Learning in Robotics

Robust control for modeling errors has been widely studied in control theory. An excellent overview of methods is provided in the book by Green & Limebeer (Green & Limebeer, 2012). In the problem of robust transfer, we are interested in parametric uncertainty between source and target models. Nilim et al. (Nilim & El Ghaoui, 2005) and Mastin et al. (Mastin & Jaillet, 2012) have analyzed the 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 (Garcıa & Fernández, 2015). 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. (Rajeswaran et al.) 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. (Yu et al.) 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. Both of these methods use random sampling methods which can be computationally challenging for realistic domains where training a single policy on a stationary MDP is hard enough.

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

This work is, to the best of our knowledge, one of the first studies to examine physically plausible perturbations to not only observations but also to cause a systematic shift in dynamics that result in a predictably worse policy performance. Furthermore, we also propose an algorithm to leverage adversarial perturbations to train policies that are robust to a wide range of perturbations in dynamics and observations.

3. Approach

3.1. Adversarial Perturbations in Deep Neural Networks

Szegedy et al. (Szegedy et al.) discovered the insightful fact 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 parametrization 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 subject to a constraint on the perturbation size. In image classification, the objective is switching prediction classes, while minimizing the perceived image perturbation. 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. (Goodfellow et al.). FGSM offers computationally efficiency at the cost of a slight decrease in attack success rate. The FGSM method makes a linear approximation of a Deep Neural Network and maximizes the objective in a closed form.

FGSM focuses on adversarial perturbations of an image input where the change in each pixel is limited by ε. With a linear approximation of a DNN, $\hat{\pi}(s) = w^{\mathrm{T}}s$, the optimal FGSM perturbation is defined as, $\delta = \varepsilon\,\mathrm{sign}(w)$. Since we define δ to be the perturbation, the output of the network on the adversarial example $\hat{s}$ is $\hat{\pi}(\hat{s}) = w^{\mathrm{T}}s + w^{\mathrm{T}}\delta$. Now assuming that the network output $\pi(s;\theta)$ is instead a non-linear function parametrized by θ, then a linearization of the loss function around the current input provides the following perturbation

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

where η is a loss function over the policy π, which is parametrized by parameters θ. Moreover, we note that a key assumption in FGSM is that the attacker has complete access to the target neural network – such as its architecture, weights, and other hyperparameters. This is a *white-box* setting. In contrast, in a *black-box* setting, the attacker does not have access to the parameters of the target network but only the output. Previous works have studied the black-box setting where gradient computation can be done numerically, and the attacker has unlimited query access to the oracle. A common approach is to re-train a separate model for the same input and output space and leverage the transferability of the adversarial examples to attack the target policy. We restrict our analysis to a white-box setting because training the policy with numerical gradient estimates is no more efficient than computing the gradient on the original policy.

3.2. Physically Plausible Threat Model

Consider a physical dynamical system:

$$\begin{aligned} x_{t+1} &= f(x_t, u_t; \mu) + \nu \qquad \texttt{(Dynamics)} \\ z_t &= g(x_t) + \omega \qquad \texttt{(Observation)} \end{aligned} \qquad (2)$$

where the `Dynamics` equation updates the state x with control input u according to a function f, parametrized by model parameters μ, and process noise ν. The `Observation` model maps the current state x to the observed state z with the observation function g and observation noise ω.

We use the full gradient method instead of the popular FGSM. FGSM is primarily designed for images where the state is high dimensional and approx. IID. However, for dynamical models, the state space is structured with different domains; hence a fixed unit step size can result in scaling issues.

We use an isometrically scaled version of the full gradient

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

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

Type of Perturbation: A malicious adversary can change either 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. Prior work in (Huang et al.; Behzadan & Munir) only examines perturbations to the current state in image space, i.e. ν, which is often not physically plausible. We perform perturbations to process noise ν by adding gradient based perturbation to the state of the system. Similarly, we add observation noise to ω by changing the observation while preserving the system state. Adversarial perturbation on dynamics noise through model 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. $Bern(\phi)$. When $\phi = 0$, only the initial time step is perturbed in each episode.

The three perturbation types described above combined with two modes of perturbation and three levels of perturbation frequency result in a total of 18 threat models. Each model is also compared with the nominal source model with no changes as a baseline.

3.3. Robust Training with Adversarial Perturbations

Direct Policy Optimization methods utilize batch trajectory sampling for gradient estimates as in (Schulman et al., 2015). The core of the ARPL operates by modifying the trajectory rollouts to include trajectory perturbation. 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 to compute the perturbation depends on the choice of threat model. A gradient update to the policy parameters is then made after a rolling out a batch of k trajectories.

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, ad-

versarial perturbations rely on the online generation of scenarios that not only expose flaws in the ways that the model conceptualizes its decision function, but also are likely to occur naturally.

4. Experimental Results

We evaluated our proposed ARPL algorithm on four continuous control tasks – `inverted pendulum`, `half-cheetah`, `hopper`, and `walker-2d` using the MuJoCo physics simulator (Todorov et al., 2012). 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 model'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-2d` 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 (Schulman et al., 2015) 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 ARPLuses 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 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.

Improving Model Robustness using Adversarial Training:

Here we evaluate the effectiveness of our robust training method proposed in Sec. 3.3. 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. This is something we would like to address in future work. 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.

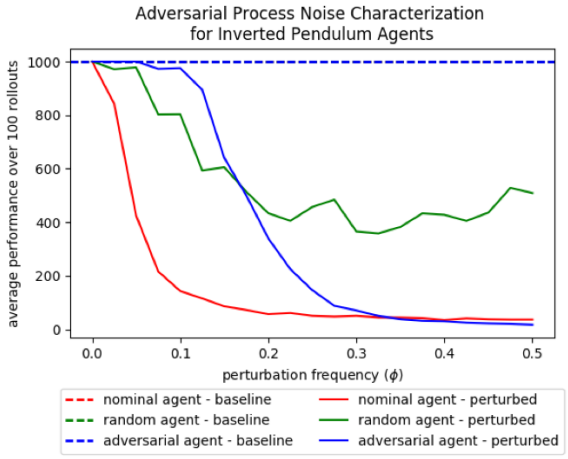


Figure 2. A comparison of agent performance with respect to adversarial process noise on the Inverted Pendulum task. Here, $\varepsilon = 0.01$, and we evaluated agent performance as we increased the perturbation frequency. The baseline performance indicates how each agent performs in an unperturbed environment. Notice that the adversarial agent does the best in the region where is was trained, but the random agent is more resistant in the higher frequency regime.

Fig. 2 and Fig. 3 show the effect of process noise in the Inverted Pendulum task. As Fig. 2 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. 3 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).

Fig. 4 and Fig. 5 show the effect of dynamics noise in

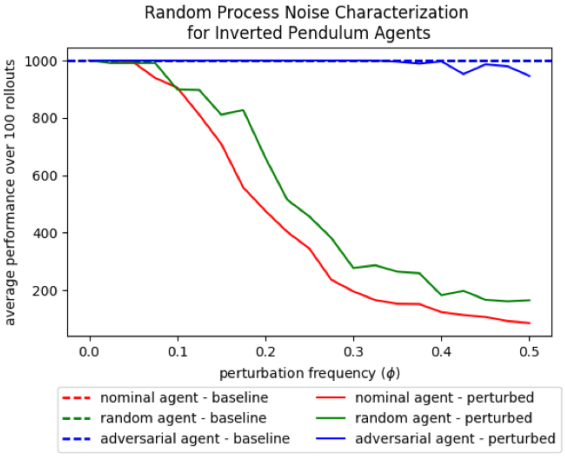


Figure 3. A comparison of agent performance with respect to random process noise on the Inverted Pendulum task. Here, $\epsilon = 0.01$, and we evaluated agent performance as we increased the perturbation frequency. The baseline performance indicates how each agent performs in an unperturbed environment. Notice that the adversarial agent is robust across all perturbation frequencies while the random agent suffers with higher frequency noise.

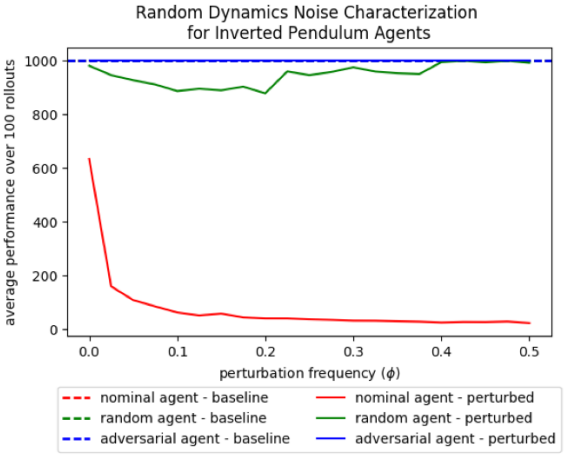


Figure 5. A comparison of agent performance with respect to random dynamics noise in the Inverted Pendulum task. Here, $\epsilon = 10.0$, and we evaluated agent performance as we increased the perturbation frequency. The baseline performance indicates how each agent performs in an unperturbed environment. Notice that the adversarial agent is robust across all perturbation frequencies.

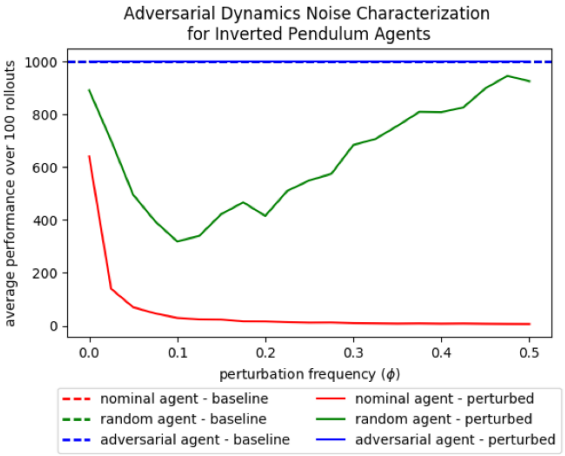


Figure 4. A comparison of agent performance with respect to adversarial dynamics noise in the Inverted Pendulum task. Here, $\epsilon = 10.0$, and we evaluated agent performance as we increased the perturbation frequency. The baseline performance indicates how each agent performs in an unperturbed environment. Notice that the adversarial agent is robust across all perturbation frequencies.

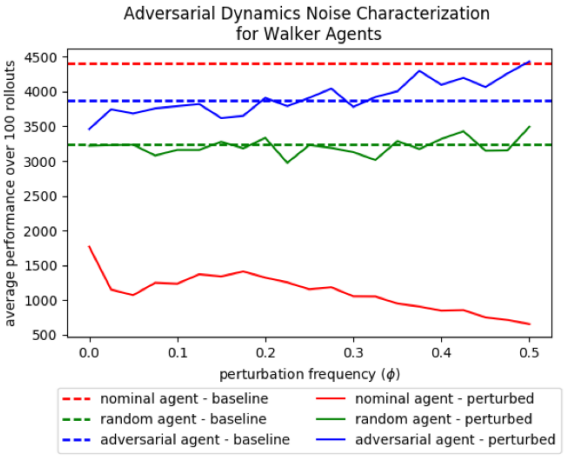


Figure 6. A comparison of agent performance with respect to adversarial dynamics noise in the Walker task. Here, $\epsilon = 10.0$, and we evaluated agent performance as we increased the perturbation frequency. The baseline performance indicates how each agent performs in an unperturbed environment. Notice that the adversarial and random agents are robust across all perturbation frequencies.

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.

Fig. 6 and Fig. 7 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.

Fig. 8, Fig. 9, Fig. 10, and Fig. 11 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

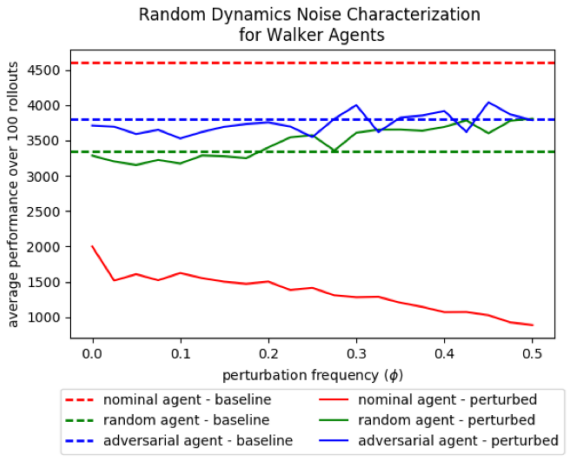


Figure 7. A comparison of agent performance with respect to random dynamics noise in the Walker task. Here, $\epsilon = 10.0$, and we evaluated agent performance as we increased the perturbation frequency. The baseline performance indicates how each agent performs in an unperturbed environment. Notice that the adversarial and random agents are robust across all perturbation frequencies.

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.

5. Conclusion

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 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.

6. Team Member Contribution and Acknowledgments

I originally started this work while working with my mentors Yuke Zhu and Animesh Garg, but this quarter, I continued to work independently on this project. Their guidance and feedback was critical for the completion of this work, and I look forward to their continued help and support. I would also like to thank my advisors Silvio and Fei-Fei.

References

Abbeel, Pieter, Quigley, Morgan, and Ng, Andrew Y. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 1–8. ACM, 2006.

Barreno, Marco, Nelson, Blaine, Sears, Russell, Joseph, Anthony D, and Tygar, J Doug. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pp. 16–25. ACM, 2006.

Behzadan, Vahid and Munir, Arslan. Vulnerability of deep reinforcement learning to policy induction attacks.

Biggio, Battista, Corona, Igino, Maiorca, Davide, Nelson, Blaine, Šrndić, Nedim, Laskov, Pavel, Giacinto, Giorgio, and Roli, Fabio. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 387–402. Springer, 2013.

Devin, Coline, Gupta, Abhishek, Darrell, Trevor, Abbeel, Pieter, and Levine, Sergey. Learning modular neural network policies for multi-task and multi-robot transfer. *arXiv preprint arXiv:1609.07088*, 2016.

Gao, Ji, Wang, Beilun, and Qi, Yanjun. Deepmask: Masking dnn models for robustness against adversarial samples.

Garcıa, Javier and Fernández, Fernando. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

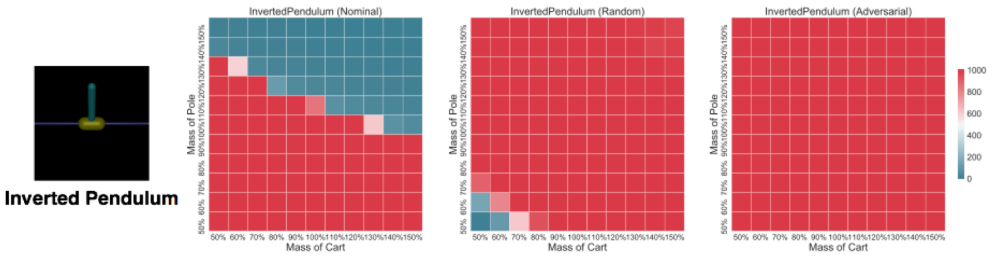


Figure 8. Policy robustness of the `inverted pendulum` agents with respect to varying dynamics configurations.

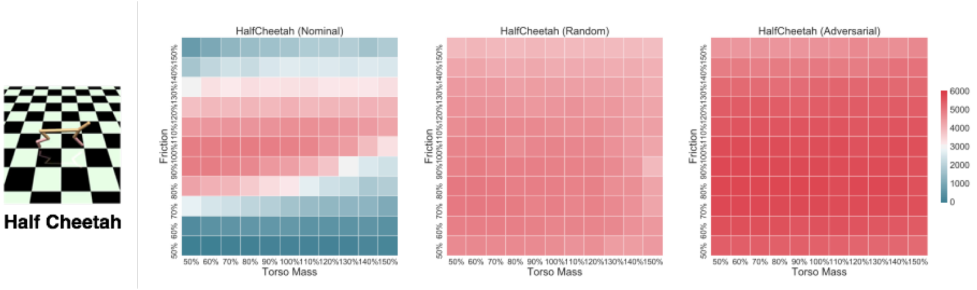


Figure 9. Policy robustness of the `half cheetah` agents with respect to varying dynamics configurations.

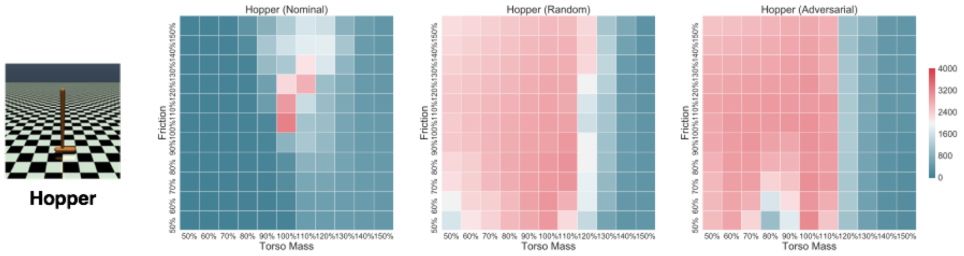


Figure 10. Policy robustness of the `hopper` agents with respect to varying dynamics configurations.

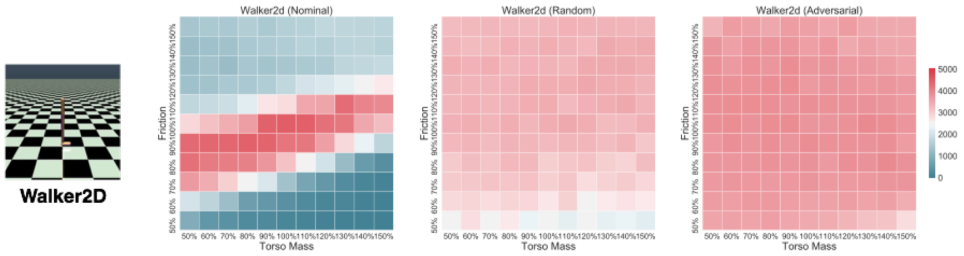


Figure 11. Policy robustness of the `walker` agents with respect to varying dynamics configurations.

Goodfellow, Ian J., Shlens, Jonathon, and Szegedy, Christian. Explaining and harnessing adversarial examples.

Green, Michael and Limebeer, David JN. *Linear robust control*. Courier Corporation, 2012.

Grosse, Kathrin, Manoharan, Praveen, Papernot, Nicolas, Backes, Michael, and McDaniel, Patrick. On the (statistical) detection of adversarial examples.

Huang, Sandy, Papernot, Nicolas, Goodfellow, Ian, Duan, Yan, and Abbeel, Pieter. Adversarial attacks on neural network policies.

Kober, Jens, Bagnell, J Andrew, and Peters, Jan. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, pp. 0278364913495721, 2013.

Kurakin, Alexey, Goodfellow, Ian, and Bengio, Samy. Adversarial examples in the physical world.

Levine, Sergey, Finn, Chelsea, Darrell, Trevor, and Abbeel, Pieter. End-to-end Training of Deep Visuomotor Policies. *Journal of Machine Learning Research*, 17(39): 1–40, 2016a.

Levine, Sergey, Pastor, Peter, Krizhevsky, Alex, and Quillen, Deirdre. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *arXiv preprint arXiv:1603.02199*, 2016b.

Lillicrap, Timothy P, Hunt, Jonathan J, Pritzel, Alexander, Heess, Nicolas, Erez, Tom, Tassa, Yuval, Silver, David, and Wierstra, Daan. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Mastin, Andrew and Jaillet, Patrick. Loss bounds for uncertain transition probabilities in markov decision processes. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pp. 6708–6715. IEEE, 2012.

Mnih, Volodymyr, Badia, Adri Puigdomnech, Mirza, Mehdi, Graves, Alex, Lillicrap, Timothy P., Harley, Tim, Silver, David, and Kavukcuoglu, Koray. Asynchronous methods for deep reinforcement learning.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Moosavi-Dezfooli, Seyed-Mohsen, Fawzi, Alhussein, Fawzi, Omar, and Frossard, Pascal. Universal adversarial perturbations.

Moosavi-Dezfooli, Seyed-Mohsen, Fawzi, Alhussein, and Frossard, Pascal. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Nilim, Arnab and El Ghaoui, Laurent. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

Papernot, Nicolas, McDaniel, Patrick, Goodfellow, Ian, Jha, Somesh, Celik, Z. Berkay, and Swami, Ananthram. Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples.

Rajeswaran, Aravind, Ghotra, Sarvjeet, Ravindran, Balaraman, and Levine, Sergey. Epopt: Learning robust neural network policies using model ensembles.

Rusu, Andrei A, Rabinowitz, Neil C, Desjardins, Guillaume, Soyer, Hubert, Kirkpatrick, James, Kavukcuoglu, Koray, Pascanu, Razvan, and Hadsell, Raia. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016a.

Rusu, Andrei A, Vecerik, Matej, Rothörl, Thomas, Heess, Nicolas, Pascanu, Razvan, and Hadsell, Raia. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*, 2016b.

Schulman, John, Levine, Sergey, Moritz, Philipp, Jordan, Michael, and Abbeel, Pieter. Trust region policy optimization. *ICML*, 2015.

Silver, David, Huang, Aja, Maddison, Chris J, Guez, Arthur, Sifre, Laurent, Van Den Driessche, George, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian, and Fergus, Rob. Intriguing properties of neural networks.

Taylor, Matthew E. and Stone, Peter. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

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

Yu, Wenhao, Liu, C. Karen, and Turk, Greg. Preparing for the unknown: Learning a universal policy with online system identification.

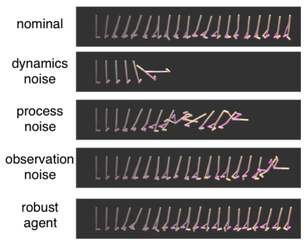
Zhu, Yuke, Mottaghi, Roozbeh, Kolve, Eric, Lim, Joseph J, Gupta, Abhinav, Fei-Fei, Li, and Farhadi, Ali. Target-driven visual navigation in indoor scenes using deep reinforcement learning. *arXiv preprint arXiv:1609.05143*, 2016.

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

Ajay Mandlekar, Yuke Zhu, Animesh Garg, Li Fei-Fei, Silvio Savarese
Department of Computer Science, Stanford University

Introduction

- 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.
- We demonstrate that Deep RL methods are susceptible to adversarial perturbations in states, model parameters, and observations.
- We introduce Adversarially Robust Policy Learning (ARPL) - an algorithm that leverages active computation of physically-plausible adversarial examples during training in order to enable robust performance under both random and adversarial perturbations of the system.



Physically-Plausible Threat Model

Dynamics Noise — Process Noise

Dynamics $x_{t+1} = f(x_t, u_t; \boxed{\mu}) + \boxed{\nu}$

Observation $z_t = g(x_t) + \boxed{\omega}$

Observation Noise

Key Idea: Can we use Adversarial Perturbations?

ARPL Algorithm

ARPL is a basic augmentation for any policy gradient method. Every iteration consists of policy evaluation and improvement.

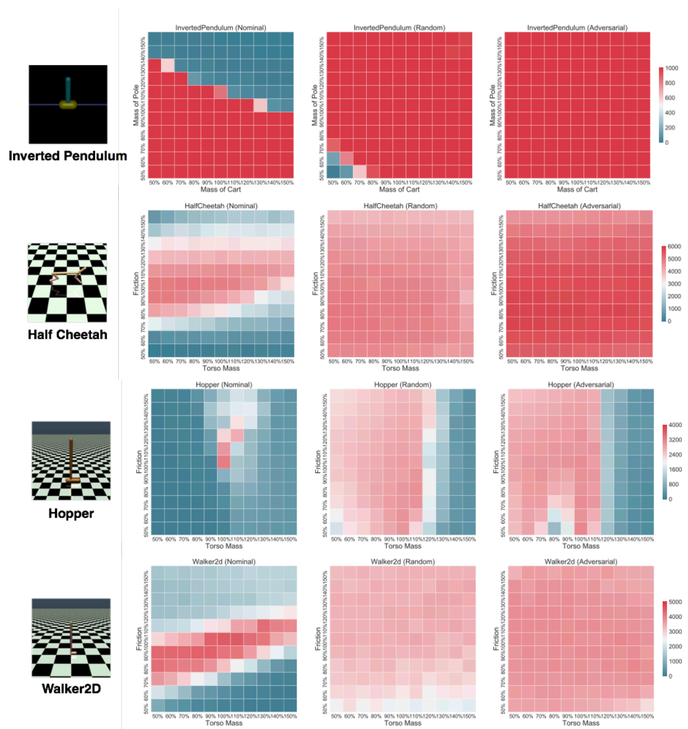
During **policy evaluation**, we collect N trajectories via policy rollouts. For every observation during a rollout, we **adversarially perturb** the state with **probability ϕ** and **magnitude ε** using the following perturbation

$$\delta = \varepsilon \, \nabla_s \eta(\pi_\theta(s))$$

where **η** is the L2 norm of the control produced by the policy **π**.

Then, we run **policy improvement**, as prescribed by the policy gradient method. Our implementation uses TRPO.

Demonstrated Robustness in Physical Dynamics Parameters



Experimental Setup

Process Noise - We perturb the original environment state and explicitly set this as the environment state in the simulator. We also feed this perturbed state to the agent as an observation.

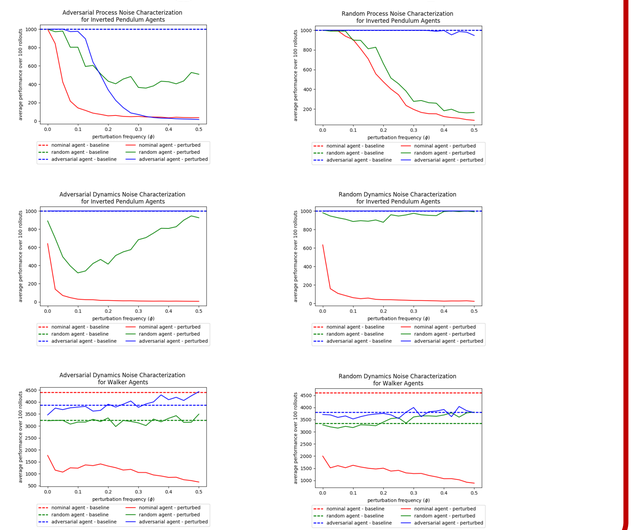
Dynamics Noise - We augment the agent's observation with environment dynamics parameters during training and use this part of the observation vector to compute perturbations for these parameters. They are then updated in the environment.

Observation Noise - Identical to process noise, but the agent receives the unperturbed state as the observation.

We experimented with the **perturbation type** (process, dynamics, observation), the **perturbation frequency**, controlled by **ϕ**, the probability of a perturbation at every time step, and whether the perturbation is generated **randomly** or **adversarially**.

We evaluate ARPL on 4 continuous control tasks using MuJoCo and Gym.

ARPL Agent Examples



References

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]

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

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]

E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control", in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, 2012, pp. 5026–5033