Robust Zero-Sum Deep Reinforcement Learning.

Olalekan Ogunmolu, Nicholas Gans, Tyler Summers.

Abstract—This paper presents a methodology for evaluating the sensitivity of deep reinforcement learning policies. This is important when agents are trained in a simulated environment and there is a need to quantify the sensitivity of such policies before exposing agents to the real world where it is hazardous to employ RL policies. In addition, we provide a framework, inspired by $H_{\infty}$ control theory, for building maximum robustness into trained deep reinforcement centric policies. This robust framework for training deep policies involve a two player zero-sum iterative dynamic game in a concave-convex environment, where the agents’ goal is to drive the dynamics to a saddle region. By formalizing an MPC trajectory optimization framework for this two-player system, we evaluate hypothesis on the guided policy search algorithm; without loss of generality, we posit that deep RL policies trained in this fashion will be maximally robust to a “worst” possible disturbance.

I. INTRODUCTION

The strategic goal of deep reinforcement learning (RL) for autonomous agent behavior in realistic environments is to combine function approximation techniques with robust learning-based control. A good RL controller should guarantee fulfillment of performance specifications under external disturbances, or modeling errors. Quite often in practice, however, this is not the case. Typical research in deep RL often involve separate pipelines of function approximation, learning-control and robust control. This makes learned deep RL policies not to often generalize well to real-world scenarios. This can be attributed to the inherent differences between the training and testing environments. Recently, there have been efforts at integrating function approximation techniques with learning-based control, in an end-to-end fashion, in order to have compact systems that fulfill objectives while guaranteeing robustness to environmental uncertainties. Examples include trajectory-based optimization for known dynamics ([1], [2]), or trajectory optimization for unknown dynamics such as guided policy search algorithms [3]–[5].

While these methods produce improvements in performance when transferring the learned policy from the simulation environment to the real world, there are sensitivity questions of such policies that need to be addressed namely, how can we guarantee maximally robust deep RL policies in the presence of external disturbances, or modeling errors? A typical approach employed in minimizing the effect of sample inefficiency is to engineer an agent’s policy in a simulated environment, and later transfer such policies to physical environments. However, questions of robustness persist in such scenarios as the agent often has to cope with modeling errors and new sensory inputs from a different environment. For continuous control tasks, learned policies may become brittle in the presence of external perturbations, or a slight change in the system dynamics may significantly affect the performance of the learned controller [6] – defeating the purpose of having a robust policy that is learned through environmental interaction.

Contributions: The contribution of this paper is two-fold:

- first, we provide a framework that demonstrates the brittleness of a state-of-the-art deep RL policy; specifically, given a trained RL policy, we pose an adversarial agent against the fixed trained policy, whose goal is to perturb the parameter space of the learned policy. We demonstrate that the most sophisticated deep policies fail in the presence of adversarial perturbations.

- second, we formulate an iterative dynamic zero-sum, two player game, where each agent executes an opposite reaction to its pair: a concave-convex problem follows explicitly, and our goal is to achieve a saddle point equilibrium, where the state is everywhere defined but possibly infinite-valued).

Noting that lack of generalization of learned reward functions to the real-world can be thought of as external disturbance that perturb the system dynamics, we formulate the learning of robust control policies as a zero-sum two player Markov game – an iterative dynamic game (iDG) – that pits an adversarial agent against a protagonist agent.

The controller aims to minimize a given cost while the second agent, an adversary aims to maximize the given cost in the presence of an additive disturbance. We run the algorithm in finite episodic settings and show a dynamic game approach aimed at generating policies that are maximally robust.

The content of this paper is thus organized: we review relevant literature to our contribution in Sec. II; we then provide an $H_{\infty}$ background in Sec. III. This $H_{\infty}$ technical introduction will be used in formulating the design of perturbation signals in Sec. IV. Without loss of generality, we provide a formal treatment of the iDG algorithm within the guided policy search framework in Sec. V. Experimental evaluation on multiple robots is provided in Sec. VI followed by conclusions in Sec. VII.

II. RELATED WORK

Robustness studies in classical control have witnessed the formalization of algorithms and computation necessary to carry out stable feedback control and dynamic game tasks.
(e.g. [7]–[9]). There now exist closed-form and iterative-based algorithms to quantify the sensitivity of a control system and design robust feedback controllers. These methods are well-studied in classical $H_{\infty}$ control theory. While questions of robustness of policies have existed for long in connectionist RL settings [10], only recently have researchers started addressing the question of incorporating robustness guarantees into deep RL controllers.

Heess et. al [11] posit that rich, robust performance will emerge if an agent is simulated in a sufficiently rich and diverse environment. [11] proposed a learning framework for agents in locomotion tasks which involved choosing simple reward functions, but exposing the agent to various levels of difficult environments as a way of achieving ostensibly sophisticated performance objectives. Incorporating various levels of difficulty in obstacles, height and terrain smoothness to an agent’s environment for every episodic task, they achieved robust behaviors for difficult locomotion tasks after many episodes ($\approx 10^6$) of training. However, this strategy defeats one of the primary objectives of RL namely, to make an agent discover good policies with finite data based on little interaction with the environment. An ideal robust RL controller must come from data-efficient samples or imitations. Furthermore, this approach takes a qualitative measure at building robust signals into the reward function via means such as locomotion hurdles with variations in height, slopes, and slalom walls. We reckon that building such physical barriers for an agent is expensive in the real-world and learning such emergent locomotion behaviors takes a long training time.

Pinto et. al. [12], posed the learning of robust RL rewards in a zero-sum, two-player markov decision process (MDP) defined by the standard RL tuple $\{S, A_1, A_2, P, R, \gamma, s_0\}$, where $A_1$ and $A_2$ denote the continuous action spaces of the two players. Both players share a joint transition probability $P$ and reward $R$. Pinto’s approach assumed a knowledge of the underlying dynamics so that an adversarial policy, $\pi_{\text{adv}}^\theta(u_i|x_t)$, can exploit the weakness in a protagonist agent’s policy, $\pi_{\text{prot}}^\theta(u_i|x_t)$. This relied on a minimax alternating optimization process: optimizing for one set of actions while holding the other fixed, to the end of ensuring robustness of the learned reward function. While it introduced $H_{\infty}$ control as a robustness measure for classical RL problems, it falls short of adapting $H_{\infty}$ for complex agent tasks and policy optimizations. Moreover, there are no theoretical analyses of saddle-/pareto-point or Nash equilibrium guarantees and the global optimum that assures maximal robustness at $\pi_{\text{prot}}^\theta(u_i|x_t) = \pi_{\text{adv}}^\theta(u_i|x_t)$ is left unaddressed.

From a methodical perspective, we formulate the robustness of RL controllers within an $H_{\infty}$ framework (see [9]) for deep robot motor tasks. Similar to a matrix game with two agents, we let both agents play a zero-sum, two person game where each agent’s action strategy or security level never falls below that of the other. The ordering according to which the players act so that each player acts optimally in a “min max” fashion does not affect the convergence to saddle point in our formulation. We consider the case where the security levels of both players coincide so that the strategy pair of both agents constitute a saddle-point pure strategy [13, p. 19].

### III. BACKGROUND AND PRELIMINARIES

We consider the standard episodic motor task where an agent interacts with an environment $E$ over a finite time horizon, $T$. As a consequence of actions $u_t \in U$ at time $t$, which are governed by a stochastic policy $\pi_{\text{prot}}(u_t|x_t,t)$, the agent enters a state $x_{t+1}$, where $\pi_{\text{adv}} : X \times U \rightarrow R$. $R$ is the accumulated reward for a trajectory defined as $R(\tau) = r_{t_{\text{end}}} + \sum_{t=0}^{T-1} r_t(x_t, u_t)$, for an instantaneous reward function, $r_t$, and a final reward, $r_{t_{\text{end}}}$. The goal is to choose an optimal control policy, $\pi_{\text{opt}}^\theta$, that maximizes the expected sum of rewards $E[R(\tau|\pi)] = \sum_{\tau} R(\tau)p_{\pi_{\text{adv}}}(\tau)d\tau$ over the trajectories $\tau = (x_t, u_t)$. Specifically, for a stochastic policy $\pi_{\text{adv}}(u_t|x_t)$, the marginals over the trajectory distribution $\tau$, is defined as $p_{\pi_{\text{adv}}} = p(x_1) \prod_{t=0}^{T-1} \pi_{\text{adv}}(u_t|x_t)p(x_{t+1}|x_t, u_t)$, where $p(x_{t+1}|x_t, u_t)$ represents the system dynamics. Many tasks in robot learning domains can be formulated in the reinforcement learning paradigm above. We are then faced with questions surrounding managing high-dimensional continuous state and action spaces, and the high cost of robot interactions with the environment.

Given the inadequacy of value function approximation methods in managing high-dimensional continuous state and action spaces as well as the difficulty of carrying out arbitrary exploration given hardware constraints [14], we resolve to use policy search (PS) methods as they operate in the parameter space of parameterized policies. However, direct PS are often specialized algorithms that produce optimal policies for a particular task (often using policy gradient methods), and they come with the negative effects of not generalizing well to flexible trajectory optimizations and large representations e.g. using neural network policies.

Guided policy search (GPS) algorithms [3]–[5] are able to guide the search for parameterized policies from poor local minima using an alternating block coordinate ascent of the optimization problem, made up of a C-Step and an S-Step. In the C-step, a well-posed cost function is minimized with respect to the trajectory samples, generating guiding distributions $p_i(u_t|x_t)$; and in the S-step, the locally learned time-varying control laws, $p_i(u_t|x_t)$, are parameterized by a nonlinear, neural network policy using supervised learning. The S-step fits policies of the form $\pi_{\text{adv}}(u_t|x_t) = \mathcal{N}(\mu^\tau(x_t), \Sigma^\tau(x_t))$ to the local controllers $p_i(u_t|x_t)$, where $\mu^\tau(x_t)$, and $\Sigma^\tau(x_t)$ are functions that are estimated. Our goal is to select a suitable policy parameterization, so as to assure robustness and stability guarantees [15]. In this paper, we specifically use the mirror descent version of GPS [5].

### IV. CASE FOR ROBUSTNESS IN PS ALGORITHMS

In this section, we show why guided policy search algorithms are non-robust to even the simplest form of perturbations – additive disturbance. Before we proceed, we note that policy search algorithms are popular among the RL tools available because they have a “modest” level of robustness built into them e.g.
Fig. 1: Representation of Robust Controllers

- by requiring the learning controller to start the policy parameterization from multiple initial states,
- adding a Kullback-Leibler (KL) constraint term to the reward function e.g. [3], [4],
- solving a motor task in multiple ways where more than one solution exist [14],
- or by introducing additive noise into the system as white noise e.g. DDP methods or their iLQG variants [16].

A fundamental drawback of these robustness mechanisms, however, is that the learned policy can only tolerate disturbance for slightly changing conditions; parametric uncertainty is not suitably modeled as white noise, and treating the error as an extra input might be relative to the size of the inputs (drawn from the environment) – necessitating the need for a formal treatment of robustness in PS algorithms.

A methodical way of solving the robustness problem in deep RL would be to consider techniques formalized in $H_\infty$ control theory, where controller sensitivity and robustness are solved within the framework of a differential game. We conjecture that the lack of robustness of a RL trained policy arises from the difference between a plant model and the real system, or the difference in learning environments. If we can measure the sensitivity of a system, $\gamma$, then we can aim for policy robustness by ensuring that $\gamma$ is sufficiently small to reject disturbance arising from the training environment or modeling errors if the gain of mapping from the error space to the disturbance is less than $\gamma^{-1}$ [17].

Fig. 1 depicts the standard $H_\infty$ control problem. Suppose $G$ in Fig. 1a is a plant for which we can find an internally stabilizing controller, $\Sigma_K$, that ensures stable transfer of input $u$ to measurement $y$, the $H_\infty$ control objective is to find the “worst” possible disturbance, $v$, which produces an undesired output, $z$; we want to minimize the effect of $z$. In Fig. 1b, we treat unmodeled dynamics, transfer errors and other uncertainty as an additional feedback to which we would like to adapt with respect to the worst possible disturbance in a prescribed range. $\Sigma_\Delta$ in the right inset represents these uncertainties; our goal is to find the closed-loop optimal policy for which the plant, $G$, will satisfy performance requirements and maintain robustness for a large range of systems $\Sigma_\Delta$. We focus on conditions under which we can make the $H_\infty$ norm of the system less than a given prior, $\gamma$. Specifically, we want to design a controller $\Sigma_K$ that minimizes the $H_\infty$ norm of the closed-loop transfer function $T_{ZV}$ from disturbance $v$ to output $z$ defined as $\|T_{ZV}\|_\infty = \sup_v \|z\|_2 / \|v\|_2$.

From the small-gain theorem, Fig. 1b will be stable for any stable mapping $\Delta : z \to v$ for $\|\Delta\|_\infty < \gamma^{-1}$ [9]. In a differential game setting, we can consider a min-max solution to the $H_\infty$ problem for the plant $G$ with dynamics given by $\dot{x} = f(x, u, w)$ so that we solve an $H_\infty$ problem that satisfies the constraint $\|T_{ZV}\|_\infty = \sup_v \|z\|_2 / \|v\|_2 \leq \gamma^2$, or find a control input $u$ that satisfies the constraint $V = \int_{t=0}^T (z^T \dot{z} - \gamma^2 v^T v) dt \leq 0$ for all possible disturbances $v$ for which $x_0 = 0$.

We consider a differential game for which the best control $u$ that minimizes $V$, and the worst disturbance $v$ that maximizes $V$ are derived from

$$V^* = \min_u \max_v \int_{t=0}^T [z^T \dot{z} - \gamma^2 v^T v] dt.$$  \hspace{1cm} (1)

The optimal value function is determined from the Hamilton-Jacobi-Isaacs (HJI) equation,

$$\min_u \max_v \left[ z^T \dot{z} - \gamma^2 v^T v + \frac{\partial V^*}{\partial x} f(x, u, v) \right] = 0$$

from which the optimal $u$ and $v$ can be computed.

A. Problem Formulation

In order to ensure the learned policy for a dynamical system is robust to external uncertainties, modeling and transfer learning errors, we propose an iterative dynamic game consisting of an agent within an environment, and an adversarial agent, interacting with the original agent in the closed-loop environment $E$, over a finite horizon, $T$ (it could also be extended to the infinite horizon case). The adversary could represent a spoofing agent in the world or modeling errors between the plant and dynamics. The states evolve according to the following stochastic dynamics $p(x_{t+1} | x_t, u_t, v_t), \forall t = 0, ..., T$ where $x_t \in X_t$ is a markovian system state, $u_t \in U_t$ is the action taken by the agent (henceforth called the protagonist), $v_t \in V_t$ is the action taken by an adversarial agent. The subscripts denote time steps $t \in [1, T]$, allowing for a simpler structuring of the individual policies per time step [14]. The problems we consider are control tasks with complex dynamics, having continuous state and action spaces, and with trajectories defined as $T = \{x_1, u_1, v_1, x_2, u_2, v_2, ..., x_T, u_T, v_T \}$. At time $t$, the system’s controller visits states with high rewards while the adversarial agent wants to visit states with low rewards. The solution to this zero-sum game follows with an equilibrium at the origin – a saddle-point solution ensues.

The policies that govern the behavior of the agents are defined as $\pi_{\theta}(u_t | x_t)$ and $\pi_{\phi}(v_t | x_t)$ respectively, and the learned local linear time-varying Gaussian controllers are defined as $p(u_t | x_t)$, $p(v_t | x_t)$. In the next subsection, we show that learned motor policies, $p(u_t | x_t)$, are sensitive to minute additive disturbances; we later on propose how to build robustness to such trained neural network policies.
is important in learning tasks where the robustness margins of a trained controller need to be known in advance before being introduced to a new execution environment.

B. Sensitivity of a learned RL Policy

This section offers an empirical guidance on testing the sensitivity of a deep neural network policy for an agent. We consider additive disturbance to a deep RL policy. Our goal is to study the degradation of performance of a trained neural network policy in the presence of the "worst" possible disturbance in the parameter space of the policy; if this disturbance cannot alter the performance of the trained policy, we have some value for the policy parameters in the prescribed range that the decision strategy is acceptable. We follow the model in Fig. 1, where $\Sigma_\Delta$ denotes the uncertainty injected by the adversary. We arrive at the nominal system from $u \rightarrow y$ when the transfer matrix of $\Sigma_\Delta$ is zero. We call $\Sigma_\Delta$ the adversary whose control, $v$, effect on the output $z$ is to be minimized. We quantify the effect of $v$ on $z$ in closed loop using a suitable cost function as a min-max criteria. This can be seen as an $H_\infty$ norm on the system. Suppose the local actions, $u_t \in \pi(x_t)$, of the controller belongs to the policy space $\pi = [\pi_0, ..., \pi_T]$ that maximize the expected sum of rewards,

$$R(x_t, u_t) = \max_u \mathbb{E} \sum_{t=0}^{T-1} r(x_t, u_t) + r_T(x_T, u_T),$$

the adversarial close-loop system is therefore given by $x_{t+1} = f_t(x_t, \pi(x_t), v_t)$. Therefore, the augmented reward for the closed-loop protagonist-adversary system becomes

$$\ell(x_t, u_t, v_t) = \min_{v_t} \mathbb{E} \sum_{t=0}^{T} R^*(x_t, u_t) + \gamma \alpha(v_t),$$

where $\alpha(\cdot)$ can be chosen as a function of the adversarial disturbance $v_t$.

C. Robust zero-sum, two-person games

To learn robust policies, we run an alternating optimization algorithm that maximizes the cost function with respect to the adversarial controller (modeled with the worst possible disturbance) and minimizes the cost function with respect to the protagonist’s policy. We consider a two-player, zero-sum Markov game framework for simultaneously learning policies for the protagonist and the adversary. We seek to learn saddle-point equilibrium strategies for the zero-sum game:

$$\min_{u_t \in \pi(x_t)} \max_{v_t \in \pi(x_t)} \mathbb{E} \sum_{t=0}^{T-1} \ell(x_t, u_t, v_t),$$

where we have overloaded notation such that $\pi(x_t) = \pi_\theta(u_t|x_t)$ and $\tilde{\pi}(x_t) = \pi_\theta(u_t|x_t); \ell(x_t, u_t, v_t) = r(x_t, u_t) + \gamma \alpha(v_t)$ is the stage cost. $\pi_\theta(x_t)$ denotes that the adversarial actions are drawn from outside of the action space of the protagonist’s policy. Fixing a value of $\gamma$ is equivalent to an assumption on the capability of the adversary or the magnitude of a worst possible disturbance. To validate this proposal, we develop locally robust controllers for a trajectory optimization problem from multiple initial states using (4) as a guiding cost; a neural network function approximator is then used to parameterize these local controllers using supervised learning. We discuss this procedure in the next section.

D. Robust GPS

GPS adds off-policy guiding samples to a sample set: this guides the policy toward spaces of high rewards. If $p(\tau)$ is the trajectory distribution induced by the locally linear Gaussian controller $p(u_t|x_t)$ and $\tilde{p}(u_t|x_t)$ denotes the previous local controller, GPS algorithms reduce the effect of visiting regions of low entropy by minimizing the KL divergence of the current local policy from the previous one as follows,

$$D_{KL}(p(\tau)||\tilde{p}(\tau)) = \mathbb{E}[-r(\tau)] - \eta \mathcal{H}(\tilde{p})$$

where $\mathcal{H}$ is the entropy term that favors broad distributions, $\eta$ is a Lagrange multiplier and the first term forces the actions $p$ to be high in regions of high reward. The trajectory is optimized using optimal control principles under linear quadratic Gaussian assumptions [16]. GPS minimizes the expected cost, $\mathbb{E}_{\tau \sim (x,u)} r(x_t, u_t)$ over the joint distribution of state and action pairs given by the marginals $\pi_\theta(\tau) = \int \prod_{t=1}^T p(x_{t+1}|x_t, u_t)$. GPS algorithms optimize the cost $J(\theta)$ via a split process of trajectory optimization of local control laws and a standard supervised learning to generalize to high-dimensional policy space settings. A generic GPS algorithm is shown in Algorithm 1. During the C-step, multiple local control laws, $p_i(u_t|x_t)$, are generated for different initial states $x_1 \sim p(x_1)$. The supervised learning stage (S-step) regresses the global policy $\pi_\theta(u|x_t)$ to all the local actions computed in the C-step. For unknown dynamics, one can fit $p(x_{t+1}|x_t, u_t)$ to sampled trajectories from the trajectory distribution under $\tilde{p}(\tau)$. To avoid divergence in dynamics, the difference between the current and previous

1This formulation assumes that $\mathcal{V}_t$ is a vector space, though one can define nonnegative adversary input penalty functions in other settings, e.g. when $\mathcal{V}_1$ is a finite set.
Our approach consists of a zero-sum, two-person dynamic game scenario, where each player is represented by a neural network policy robust to these uncertainties. We begin by augmenting the computing neural network policy robust to these uncertainties, we propose a zero-sum, two-person dynamic game scenario in the sensitivity section in subsection IV-B. To make the changes in environment settings or disturbance as we show in the sensitivity section in subsection IV-B. To make the computed neural network policy robust to these uncertainties, we propose a zero-sum, two-person dynamic game scenario in the next section.

V. TWO-PLAYER ZERO-SUM ITERATIVE DYNAMIC GPS

To guarantee robust performance during the training of policies of a stochastic system, we introduce the “worst” disturbance in the $H_{\infty}$ paradigm to the search for a good guiding distribution problem. We begin by augmenting the reward function with a term that allows for withstanding a disturbing input

$$\ell(x_t, u_t, v_t, t) = r(x_t, u_t, v_t, t) + \gamma^2 v^T v.$$  \hspace{1cm} (6)

where $\gamma^2 v^T v$ allows us to introduce a quadratic weighting term in the disturbing input; $\gamma$ denotes the robustness parameter. A zero-sum game follows explicitly: the protagonist is guided toward regions of high reward regions while adversary pulls in its own favorite direction – yielding a saddle-point solution. This framework facilitates learning control decision strategies that are robust in the presence of disturbances and modeling errors – improving upon the generic optimal control policies that GPS and indeed deep RL algorithms guarantee.

A. MPC-BASED TRAJECTORY OPTIMIZATION

We propose repeatedly solving an MPC-based finite-horizon trajectory optimization problem within the framework of DDP. Specifically, we generalize a DDP variant – the iLQR algorithm of [18], to a two-player, zero-sum dynamic game as follows:

- We iteratively approximate the nonlinear dynamics, $\dot{x} = f(x_t, u_t, v_t)$, starting with nominal control, $u_t, v_t$; $t \in [0, t_f]$, and nominal adversarial input $v_t; t \in [0, t_f]$ which are assumed to be available.
- We run the passive dynamics with $u_t$ and $v_t$ to generate a trajectory $(\dot{x}_t, u_t, v_t)$
- Discretizing time, we linearize the nonlinear system, $\dot{x}_t$, about $(\dot{x}_k, u_k, v_k)$, so that the new state and action pairs become

$$\delta x_k = x_k - \bar{x}_k, \quad \delta u_k = u_k - \bar{u}_k, \quad \delta v_k = v_k - \bar{v}_k$$

and $\delta x_k, \delta u_k,$ and $\delta v_k$ are measured w.r.t. the nominal vectors $\bar{x}_k, \bar{u}_k, \bar{v}_k$ and are not necessarily small. The LQR approximation to the original optimal control problem and reward become

$$\delta x_{k+1} = f(x_k, \delta x_k) + f_x \delta x_k + f_v \delta v_k$$

and

$$\ell(x_k, u_k, v_k) = \frac{1}{2} \delta x_k^T \ell_{xk} + \ell_{uk} \delta u_k + \ell_{vk} \delta v_k + \ell_{xuk} \delta x_k \delta u_k + \ell_{xvk} \delta x_k \delta v_k + \ell_{vuk} \delta v_k \delta u_k + \ell_{vvk} \delta v_k \delta v_k$$

where single and double subscripts in the augmented reward denote first-order and second-order derivatives respectively, and $f_{xk}$ and $f_{vk}$ are the respective Jacobians e.g., $f_{xk} = \frac{\partial f}{\partial x}|_{x_k}$ and $f_{vk} = \frac{\partial f}{\partial v}|_{v_k}$ at time $k$. $\ell$ is an additive random noise term (folded into $v_t$ in our implementation); the value function is the cost-to-go given by the min-max of the control sequence

$$V(x_k) = \min_{u_t, v_t} \max_{\ell_t} \ell_t(x_t, u_t, v_t).$$

Setting $\nu(x_k) = \ell_{xk}(x_k, u_k, v_k)$, where $k_f$ is the final time step, the dynamic programming problem transforms the min-max over an entire control sequence to a series of optimizations over a single control, which proceeds backward in time as

$$V(x_k) = \min_{u_k} \left[ \ell(x_k, u_k, v_k) + V(f(x_{k+1}, u_{k+1}, v_{k+1})) \right].$$

The Hamiltonian term within the min-max argument can be considered as a function of perturbations around the tuple $(x_k, u_k, v_k)$. Given the intractability of solving the Bellman partial differential equation above, we restrict attention to the local neighborhood of the nominal trajectory by expanding a power series about the nominal, nonoptimal trajectory similar to [19]:

- We maintain a second-order local model of the perturbed $Q$-coefficients of the LQR problem, $(Q_k, Q_{ux_k}, Q_{uu_k}, Q_{vx_k}, Q_{uxv}, Q_{vux}, Q_{vxv}, Q_{vuv}, Q_{vvv})$ defined thus

$$Q(x_k - \bar{x}_k, u_k - \bar{u}_k, v_k - \bar{v}_k) = \ell(x_k - \bar{x}_k, u_k - \bar{u}_k, v_k - \bar{v}_k) + V(f(x_k, u_k, v_k)),$$

- a second-order Taylor approximation of $Q(\delta x_k, \delta u_k, \delta v_k)$ in the preceding equation yields

$$\approx \frac{1}{2} \left( \begin{array}{c} \delta x_k \\ \delta u_k \\ \delta v_k \\ \end{array} \right)^T \left( \begin{array}{cccc} 0 & Q_{xk}^T & Q_{uxk}^T & Q_{vxk}^T \\ Q_{xk} & Q_{xxk} & Q_{xuxk} & Q_{xxvk} \\ Q_{uxk} & Q_{uxxk} & Q_{uxuk} & Q_{uxvv} \\ Q_{vxk} & Q_{vxuxk} & Q_{vxuk} & Q_{vvv} \\ \end{array} \right) \left( \begin{array}{c} \delta x_k \\ \delta u_k \\ \delta v_k \end{array} \right)$$

- the best possible (protagonist) action and the worst possible (adversarial) action can be found by performing the respective arg min and arg max operations

$$u^*_k = \arg \min_{u_k} Q(\delta x_k, u_k, \delta v_k),$$

and

$$v^*_k = \arg \max_{v_k} Q(\delta x_k, \delta u_k, \delta v_k)$$

so that we have the fol-

\footnote{where $Q_k = \ell(x_k, u_k, v_k) + V(x_{k+1}, k + 1)$. Vector subscripts indicate partial derivatives.}
following linear controllers that minimize and maximize the quadratic \( Q \)-function respectively:

\[
\delta u_k^* = -Q_{uu}^{-1} [Q u_k + Q u x_k \delta x_k + Q u v_k \delta v_k],
\]
\[
\delta v_k^* = -Q_{vv}^{-1} [Q v_k + Q v x_k \delta x_k + Q v u_k \delta u_k].
\]

For nonlinear systems, the inverse of the 2nd partial derivatives of the Hamiltonian with respect to the controls must be strictly positive definite. When the inverse of the Hessians above are non-positive-definite, we can circumvent the bottleneck by adding a suitably large positive quantity to \( Q_{uu}^{-1} \) and \( Q_{vv}^{-1} \) [20], [21], or by replacing the Hessian with an identity matrix (which gives the steepest descent) [16]. We find that the protagonist and adversary in the above-equations have a local action containing a state feedback term, \( G \), and an open-loop term, \( g \), given by

\[
g_{u_k} = -Q_{uu}^{-1} [Q u_k + Q u v_k \delta v_k], \quad g_{v_k} = -Q_{vv}^{-1} [Q v_k + Q v u_k \delta u_k],
\]

respectively. The tuple \( \{g_{u_k}, g_{u_k}, g_{v_k}, G_{u_k}\} \) can be computed efficiently as shown in (18). We can construct linear Gaussian controllers with mean given by the deterministic optimal solutions and the covariance proportional to the curvatures of the respective \( Q \) functions:

\[
p(u_k | x_k) = N(\bar{u} + g_{u_k} + G_{u_k} \delta x_k, Q_{uu}^{-1}),
\]
\[
p(v_k | x_k) = N(\bar{v} + g_{v_k} + G_{v_k} \delta x_k, Q_{vv}^{-1}).
\]

[22] has shown that these type of distributions optimize an objective function with maximum entropy given by

\[
\arg \min_{p(\bar{u}) \in N(\bar{u})} \mathbb{E}[\ell(\bar{u}) - \mathcal{H}(p(\bar{u}))]
\]

subject to \( p(x_{t+1} | x_t, u_t) = N(x_{t+1}; f x_t x_t + f u u_t u_t, F_t) \)

(9)

while \( p(v_k | x_k) \) optimizes

\[
\arg \max_{p(\bar{v}) \in N(\bar{v})} \mathbb{E}[\ell(\bar{v}) - \mathcal{H}(p(\bar{v}))]
\]

subject to \( p(x_{t+1} | x_t, v_t) = N(x_{t+1}; f x_t x_t + f v v_t v_t, F_t) \)

(10)

where \( x^i_t, u^i_t, v^i_t \) is the system’s trajectory evolution over all states, \( i \), visited by both local controllers and \( \mathcal{H} \) is the differential entropy. Equation (9) produces a trajectory that follows the widest, highest-entropy distribution while minimizing the expected cost under linearized dynamics and quadratic cost; (10) produces an opposing trajectory to what \( p(u_k | x_k) \) does by maximizing the expected cost under locally linear quadratic assumptions about the dynamics.

Note that the open-loop control strategies in (8) depend on the action of the other player. Therefore, equations (8) ensure we have a cooperative game in which the protagonist and the adversary alternate between taking best possible and worst possible local actions during the trajectory optimization phase. This helps maintain equilibrium around the system’s desired trajectory, while ensuring robustness in local policies. Substituting (8) into (7) and equating coefficients of \( \delta x_k, \delta u_k, \delta v_k \) to those of \( V(x_k + \delta x_k, k) = V(x_k, k) + V_{x_k} \delta x_k + \frac{1}{2} V_{xx_k} \delta x_k^2 \), we obtain a quadratic value function at time \( k \), through the backward pass,

\[
\Delta V_k = -\frac{1}{2} [Q u u_k Q_{uu}^{-1} Q u_k + Q v v_k Q_{vv}^{-1} Q v_k]
\]

\[
V_{x_k} = Q x x_k, \quad V_{x u_k} = Q x u_k, \quad V_{u u_k} = Q u u_k, \quad V_{x v_k} = Q x v_k, \quad V_{v v_k} \quad (11)
\]

\[
V_{x x_k} = Q x x_k, \quad V_{x u_k} = Q x u_k, \quad V_{u u_k} = Q u u_k, \quad V_{x v_k} = Q x v_k, \quad V_{v v_k} \quad (11)
\]

Say, the protagonist first implements its strategy, then transmits its information to the adversary, who subsequently chooses its strategy; it follows that the adversary can choose a more favorable outcome since it knows what the protagonist’s choice of strategy is. It becomes obvious that the best action for the protagonist is to choose a control strategy that is an optimal response to the choice of the adversary determined from

\[
\delta u_k = \min \ell(x_k, u_k, v_k) \quad \text{max} \ell(x_k, u_k, v_k).
\]

Similarly, if the roles of the players are changed, the protagonist response to the adversary’s worst choice will be

\[
\delta u_k = \max \ell(x_k, u_k, v_k) \quad \text{min} \ell(x_k, u_k, v_k).
\]

Therefore, it does not matter that the order of play is predetermined. We end up with an iterative dynamic game, where each agent’s strategy depends on its previous actions. The update rules for the \( Q \) coefficients are determined using a Gauss-Newton approximation and is given in (16) in appendix VIII.

In the forward pass, we integrate the state equation, \( x \), compute the protagonist’s deterministic optimal policy and update the trajectory as follows:

\[
g(x_k) = \bar{u} + g_{u_k} + G_{u_k} (x_k - \bar{x}_k)
\]

\[
x_{k+1} = f_k(x_k, u_k, v_k)
\]

(12)

Compared to previous GPS algorithms, the local controllers not only produce locally linear Gaussian controllers that favors the widest and highest entropy, they also have robustness to disturbance and modeling errors built into them in the \( H_{\infty} \) sense.

We arrive at a saddle point in the energy space of the cost function and we posit that the local controllers generated during the trajectory optimization phase become robust to external perturbations, modeling errors e.t.c. We arrive at a saddle point in the energy space of the cost function and we posit that the local controllers generated during the trajectory optimization phase become robust to external perturbations, modeling errors e.t.c. The next section shows how we generate the function \( V(x_k) \) that guarantees saddle-point equilibria for our examples.

B. Supervised learning of global neural network policies

The policies \( p_{\tau_k}(u_k | x_k) \) and \( p_{\tau_k}(v_k | x_k) \) will ideally be generated for all possible initial states \( x_k \) \~ p(x_k). Since the ILQG-based linearized dynamics will only be valid within a finite region of the state space; we used the KL-divergence
constraint proposed in [5] to ensure the current protagonist policy does not diverge too much from the previous policy.

The learning problem involves imposing KL constraints on the cost function such that the protagonist controller distribution agrees with the global policy \( \pi_d (u_k | x_k) \) by performing the following alternating optimization between two steps at each iteration \( i \):

\[
\begin{align*}
\text{arg min}_{\pi_{i+1}} \text{max}_{p_{i+1}} & \left[ \mathbb{E}_{p_{i+1}} \left( \tau \right) \sum_{k=1}^{K} l(x_k, u_k, v_k) \right] \\
\text{s.t.} \quad D_{KL}(p_i (u_k | x_k), \pi_i) \leq \epsilon,
\end{align*}
\]

Essentially, \( p(u_k | x_k) \) above generates robust local policies; the first step in (13) solves for a robust local policy \( p(u_k | x_k) \) via the min-max operation, by constraining \( p(u_k | x_k) \) against its global policy \( \pi_i \) using the given KL divergence constraint; the second step projects the local linear Gaussian controller distribution onto the constraint set \( \Pi_\Theta \), with respect to the divergence \( D(p_i, \pi) \). The local policy that governs the agent’s dynamics is given by

\[
p(u_k | x_k) = \mathcal{N}(\bar{u} + g_{u_k} + G_{u_k} \delta x_k, Q_{uu_k}^{-1}).
\]

Notice that the state is linearly dependent on the mean of the distribution \( p(u_k | x_k) \) and the covariance is independent of \( v_k \); we therefore end up with a linear Gaussian controller for the robust guided policy search algorithm. For linear Gaussian dynamics and policies, the iterative KL constraint during the S-step translates to minimizing the KL-divergence between policies i.e.,

\[
D_{KL}(p_i(\tau) \| \pi_\theta(\tau)) = \sum_{k=1}^{K} \mathbb{E}_{p(u_k | x_k)} D_{KL}(p(u_k | x_k) \| \pi_\theta(u_k | x_k)).
\]

For the nonlinear cases that we treat in this work, the KL-divergence term in the S-step above is flipped as proposed in [5] so that \( D_{KL}(\pi_\theta | p(u_k | x_k)) \) minimizes the augmented stage cost under \( p_i(u_k | x_k) \) w.r.t \( \pi_\theta(u_k | x_k) \). Therefore, the S-step minimizes,

\[
\frac{1}{|D_i|} \sum_{i,k} D_{KL}(\pi_\theta(u_{i,k} | x_k) \| p_i(u_k | x_k)) \approx \sum_{i,k,j} E_{p_i(u_k | x_k)} D_{KL}(\pi_\theta(u_{i,k,i,j} | x_k) \| p_i(u_k | x_k)),
\]

where \( x_{k,i,j} \) is the \( j \)th sample from \( p_i(x_k) \) obtained by running \( p_i(u_k | x_k) \) on the real system, and \( D_i \) are the trajectory samples rolled out on the system. Our robust GPS algorithm is thus given in algorithm 2. \( \epsilon \) is adjusted based on the formulation in [5]. The dynamics \( p_i(x_{k+1} | x_k, u_k, v_k) \) are fitted to samples \( \{ x^i_{k+1}, u^i_k, v^i_k \} \) using a mixture of Gaussian models to the generated samples \( \{ x^i_{k+1}, u^i_k, v^i_k \} \) at iteration \( i \) for all time \( k \). Specifically, we incorporate a normal-inverse-Wishart prior on the Gaussian model as described in [4, §A.3].
Algorithm 2 Robust guided policy search: unknown nonlinear dynamics

1: for iteration $k \in \{1, \ldots, K\}$ do
2: Generate samples $D_{ui} = \{\pi_{i,j}\}$ and $D_{vi} = \{\pi_{vi,j}\}$ by running $p_i(u_k|x_k)$ and $p_i(v_k|x_k)$ or $\pi_{th}(u|x_k)$ and $\pi_{th}(v|x_k)$
3: Fit linear-Gaussian dynamics $p_i(x_{k+1}|x_k, u_k, v_k)$ using samples in $D_{ui}$ and $D_{vi}$
4: Fit linearized global policies $\pi_{th}(u_k|x_k)$ based on the local linear Gaussian controllers in steps 2
5: C-step: $P_{adv} \leftarrow \arg\min_{\pi_{th}} \max_{w} \left[ \mathbb{E}_{p(\bar{\tau})} \left( \sum_{k=1}^{K} l(x_k, u_k, v_k) \right) \right]$ such that $D_{KL}(p_{adv}(\tau)||\pi_{th}(\tau)) \leq \epsilon$
6: S-step: $\pi_{th} \leftarrow \arg\min_{\theta} \max_{\pi_{th}} \sum_{k,i,j} D_{KL}(\pi_{th}(u_k|x_k, i, j)||p_i(u_k|x_k, i, j))$ (via supervised learning)
7: Adjust $\epsilon$
8: end for

Fig. 3: Sensitivity Analysis for Peg Insertion Task

Fig. 4: Sensitivity Analysis for Arm Swing-up Task

dexterous manipulation. The robot has 12 states consisting of joint angles and angular velocities with two controller states. We train the protagonist’s policy using the GPS algorithm. We then pit an adversarial disturbance against the trained policy so that the adversary stays in closed-loop with the trained protagonist; The closed-loop cost function is given by

$$\ell(x_k, u_k, v_k) = \frac{1}{2} w_u u_k^T u_k + w_p \ell_{12}(d_{x_k} - d^*) - \gamma v_k^T v_k$$

(15)

where $\gamma$ represents the disturbance term, $d_{x_k}$ denotes the end effector’s (EE) position at state $x_k$ and $d^*$ denotes the EE’s position at the slot’s base. $\ell_{12}(\zeta)$ is a term that makes the peg reach the target at the hole’s base, precisely given by $\frac{1}{2} \alpha^2 \zeta + (\alpha + \zeta^2)^2$. We set $w_u$ and $w_p$ to $10^{-6}$ and 1 respectively. For various values of $\gamma$, we check the sensitivity of the trained policy and its effect on the task performance by maximizing the cost function above w.r.t $v_k$. We run each sensitivity experiment for a total of 10 iterations. Fig. 3 shows that the adversary causes a sharp degradation in the protagonist’s performance for values of $\gamma < 1.5$. This corresponds to when the GPS-trained policy gets destabilized and the arm struggles to reach the desired target. As values of $\gamma \geq 1.5$, however, we find that the adversary has a reduced effect on task performance: the adversary’s effect decreases as $\gamma$ gets larger. Video of this result is available at https://goo.gl/52rKnt. This experiment further confirms that the state-of-the-art reinforcement learning algorithms fail in the presence of additive disturbances to their parameter space making them brittle when used in situations that call for robustness. To mitigate these sensitivity errors, we implement the robust two-player, zero-sum game framework provided in V in order to develop more robust deep RL controllers and mitigate modeling errors and uncertainty.
B. Robust RL with GPS

As proposed in section V, our goal is to improve the robustness of the controller’s policy in the presence of modeling errors and uncertainties and transfer errors. We follow the formulation in section V and generate $v_k$ from zero-mean, unit variance noise samples in every iteration. We employ various values of $\gamma$ as a robustness parameter and we run the iterative dynamic game during the trajectory optimization phase of the GPS algorithm. Specifically, for the values of $\gamma$ that the erstwhile policies in the previous subsection fail, we show that our iterative dynamic game provides robustness in performance at test time compared against using just the GPS algorithm. We run experiments on the peg insertion task to verify the algorithm:

VII. CONCLUSION

We have evaluated the sensitivity of select deep reinforcement learning algorithms and shown that despite the most carefully designed policies, such policies implemented on real-world agents exhibit a potential for disastrous performances when unexpected such as when there exist modeling errors and discrepancy between training environment and real-world roll-outs (as evidenced by the results from the two dynamics the agent faces in our sensitivity experiment).

VIII.

The $Q$-coefficients are estimated using LQR as follows:

$$Q_{xk} = \ell_{xk} + f_{xk}^T V_{xk+1}$$
$$Q_{uk} = \ell_{uk} + f_{uk}^T V_{xk+1}$$
$$Q_{vk} = \ell_{vk} + f_{vk}^T V_{xk+1}$$

$$Q_{xxk} = \ell_{xxk} + f_{xxk}^T V_{xxk+1} + f_{xxk}^T f_{xk}$$
$$Q_{uxk} = \ell_{uxk} + f_{uxk}^T V_{xxk+1} + f_{uxk}^T f_{xk}$$
$$Q_{vxk} = \ell_{vxk} + f_{vxk}^T V_{xxk+1} + f_{vxk}^T f_{xk}$$

$$Q_{uuk} = \ell_{uuk} + f_{uuk}^T V_{xxk+1} + f_{uuk}^T f_{xk}$$
$$Q_{vvk} = \ell_{vvk} + f_{vvk}^T V_{xxk+1} + f_{vvk}^T f_{xk}$$

IX.

The closed-form equations for the closed loop policy obtained after substituting (8) are given by

$$\delta u_k = -K^*_k (Q_{uk} Q_{vvk} - Q_{uvk} Q_{vk})$$
$$+ (Q_{uxk} Q_{vvk} - Q_{uvk} Q_{vxk}) \delta x_k$$

$$\delta v_k = K^*_k (Q_{uvk} Q_{uxk} - Q_{uuu} Q_{vxk})$$
$$+ (Q_{uvk} Q_{uxk} - Q_{uuu} Q_{vxk}) \delta x_k$$

where $K^*_k = (Q_{uuu} Q_{xxk} - Q_{uvk} Q_{vxk})^{-1}$. The above equations can be written compactly as,

$$\delta u_k = g_{uk} + G_{uk} \delta x_k, \quad \delta v_k = g_{vk} + G_{vk} \delta x_k$$

Similarly, we find that the value coefficients are

$$V_{xk} = -K_v (Q_{uv} Q_{uu} Q_{ux} - Q_{uv} Q_{vv} Q_{vx})$$
$$+ Q_{uv} Q_{vx} Q_{uv} - 2Q_{uv} Q_{ux} Q_{vk}$$
$$+ 2Q_{uv} Q_{vx} Q_{uv} - 2Q_{uv} Q_{vv} Q_{vx}$$

$$V_{xxk} = -K_v (Q_{uv} Q_{xx} Q_{uv} - 2Q_{uv} Q_{vx} Q_{vv})$$
$$+ Q_{vx} Q_{uv} Q_{vx} + Q_{uv} Q_{vx} Q_{vx} - Q_{uu} Q_{vv} Q_{vx}$$

where $K_v = \frac{1}{2} (Q_{uv} Q_{xx} Q_{uv})^{-1}$, and $\Delta V_k = V(x_k, \delta x_k, k) - V(x_k, k)$.

In practice, the inverse terms in the recursive equations above will result in numerical errors when the matrices are not well-posed. To guarantee simulation stability, we replace $K_v$ and $K^*_v$ with identity when their leading minors are not positive definite.

REFERENCES

[1] I. Mordatch, K. Lowrey, G. Andrew, Z. Popovic, and E. V. Todorov, “Interactive control of diverse complex characters with neural networks,” in Advances in Neural Information Processing Systems, 2015, pp. 3132–3140.
[2] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, “Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search,” in Robotics and Automation (ICRA), 2016 IEEE International Conference on. IEEE, 2016, pp. 528–535.
[3] “Learning Complex Neural Network Policies with Trajectory Optimization,” Proceedings of the 31st International Conference on Machine Learning (ICML-14), vol. 32, pp. 829–837, 2014.
[4] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-End Training of Deep Visuomotor Policies,” Journal of Machine Learning Research, vol. 17, pp. 1–40, 2016.
[5] W. Montgomery and S. Levine, “Guided Policy Search as Approximate Mirror Descent,” arXiv preprint arXiv:1607.04614, 2016.
[6] T. Summers, O. Ogunmolu, and N. Gans, “Robustness Margins and Robust Guided Policy Search for Deep Reinforcement Learning,” IEEE/RSJ International Conference on Robots and Intelligent Systems, (Abstract Only Track), 2017.
[7] T. Başar and P. Bernhard, H-infinity Optimal Control And Related Minimax Design Problems: A Dynamic Game Approach. Springer Science & Business Media, 2008.
[8] M. L. Littman, “Markov Games as a Framework for Multi-agent Reinforcement Learning,” in Proceedings of the Eleventh International Conference on Machine Learning, vol. 157, 1994, pp. 157–163.
[9] J. Morimoto and K. Doya, “Robust Reinforcement Learning,” Neural computation, vol. 17, no. 2, pp. 335–359, 2005.
[10] R. J. Williams, “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning,” Machine Learning, vol. 8, pp. 229–256, 1992.
[11] N. Heess, D. Th, S. Sriram, J. Lemmon, J. Merlon, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. Riedmiller, and D. Silver, “Emergence of Locomotion Behaviours in Rich Environments,” 2017.
[12] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, “Robust Adversarial Reinforcement Learning,” arXiv preprint arXiv:1703.02702, 2017.
[13] Basar, Tamer and Olsder, Geert Jan, Dynamic Noncooperative Game Theory. Academic Press, New York, 1999.
[14] M. P. Deisenroth, G. Neumann, and J. Peters, “A Survey on Policy Search for Robotics,” Foundations and Trends in Robotics, vol. 2, no. 1, pp. 1–142, 2011.
[15] D. Bertsekas, Dynamic Programming and Optimal Control. Athena Scientific, Nashua, NH, USA, 2005.
[16] E. Todorov and W. Li, “A Generalized Iterative Lag Method For Locally-optimal Feedback Control Of Constrained Nonlinear Stochastic Systems,” 43rd IEEE Conference on Decision and Control.
[17] K. Zhou, J. C. Doyle, K. Glover, et al., Robust and optimal control. Prentice Hall New Jersey, 1996, vol. 40.
[18] Y. Tassa, T. Erez, and E. Todorov, “Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization,” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2012.

[19] J. Morimoto, G. Zeglin, and C. Atkeson, “Minimax Differential Dynamic Programming: Application to A Biped Walking Robot,” *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, no. October, pp. 1927–1932, 2003.

[20] H. Kelley, *AIAA Astrodynamics Specialists Conference*, Yale University, August 1963.

[21] *IEEE Transactions on Automatic Control*, pp. AC–12, 666, 1967.

[22] S. Levine and V. Koltun, “Guided Policy Search,” *Proceedings of the 30th International Conference on Machine Learning*, vol. 28, pp. 1–9, 2013.

[23] 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.

[24] E. Catto, “Pybox2d.” [Online]. Available: https://github.com/pybox2d/pybox2d

[25] C. Finn, M. Zhang, J. Fu, X. Tan, Z. McCarthy, E. Scharff, and S. Levine, “Guided Policy Search Code Implementation,” 2016, software available from rll.berkeley.edu/gps. [Online]. Available: http://rll.berkeley.edu/gps