Control Regularization for Reduced Variance Reinforcement Learning

Richard Cheng [1] **Abhinav Verma** [2] **Gábor Orosz** [3] **Swarat Chaudhuri** [2] **Yisong Yue** [1] **Joel W. Burdick** [1]

Abstract

Dealing with high variance is a significant challenge in model-free reinforcement learning (RL). Existing methods are unreliable, exhibiting high variance in performance from run to run using different initializations/seeds. Focusing on problems arising in continuous control, we propose a functional regularization approach to augmenting model-free RL. In particular, we regularize the behavior of the deep policy to be similar to a policy prior, i.e., we regularize in function space. We show that functional regularization yields a bias-variance trade-off, and propose an adaptive tuning strategy to optimize this trade-off. When the policy prior has control-theoretic stability guarantees, we further show that this regularization approximately preserves those stability guarantees throughout learning. We validate our approach empirically on a range of settings, and demonstrate significantly reduced variance, guaranteed dynamic stability, and more efficient learning than deep RL alone.

1. Introduction

Reinforcement learning (RL) focuses on finding an agent's policy (i.e. controller) that maximizes long-term accumulated reward. This is done by the agent repeatedly observing its state, taking an action (according to a policy), and receiving a reward. Over time the agent modifies its policy to maximize its long-term reward. Amongst other applications, this method has been successfully applied to control tasks (Lillicrap et al., 2016; Schulman et al., 2015; Ghosh et al., 2018), learning to stabilize complex robots.

In this paper, we focus particularly on policy gradient (PG) RL algorithms, which have become popular in solving continuous control tasks (Duan et al., 2016). Since PG algorithms focus on maximizing the long-term reward through trial and error, they can learn to control complex tasks without a prior model of the system. This comes at the cost of slow, high variance, learning – complex tasks can take millions of iterations to learn. More importantly, variation between learning runs can be very high, meaning some runs of an RL algorithm succeed while others fail depending on randomness in initialization and sampling. Several studies have noted this high variability in learning as a significant hurdle for the application of RL, since learning becomes unreliable (Henderson et al., 2018; Arulkumaran et al., 2017; Recht, 2019). All policy gradient algorithms face the same issue.

We can alleviate the aforementioned issues by introducing a control-theoretic prior into the learning process using functional regularization. Theories and procedures exist to design stable controllers for the vast majority of real-world physical systems (from humanoid robots to robotic grasping to smart power grids). However, conventional controllers for complex systems can be highly suboptimal and/or require great effort in system modeling and controller design. It would be ideal then to leverage *simple*, suboptimal controllers in RL to reliably learn high-performance policies.

In this work, we propose a policy gradient algorithm, CORE-RL (COntrol REgularized Reinforcement Learning), that utilizes a functional regularizer around a, typically suboptimal, control prior (i.e. a controller designed from any prior knowledge of the system). We show that this approach significantly lowers variance in the policy updates, and leads to higher performance policies when compared to both the baseline RL algorithm and the control prior. In addition, we prove that our policy can maintain control-theoretic stability guarantees throughout the learning process. Finally, we empirically validate our approach using three benchmarks: a car-following task with real driving data, the TORCS racecar simulator, and a simulated cartpole problem. In summary, the main contributions of this paper are as follows:

- We introduce functional regularization using a control prior, and prove that this significantly reduces variance during learning at the cost of potentially increasing bias.
- We provide control-theoretic stability guarantees throughout learning when utilizing a robust control prior.
- We validate experimentally that our algorithm, CORE-

[1]California Institute of Technology, Pasadena, CA [2]Rice University, Houston, TX [3]University of Michigan, Ann Arbor, MI. Correspondence to: Richard Cheng <rcheng@caltech.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

RL, exhibits reliably higher performance than the base RL algorithm (and control prior), achieves significant variance reduction in the learning process, and maintains stability throughout learning for stabilization tasks.

2. Related Work

Significant previous research has examined variance reduction and bias in policy gradient RL. It has been shown that an unbiased estimate of the policy gradient can be obtained from sample trajectories (Williams, 1992; Sutton et al., 1999; Baxter & Bartlett, 2000), though these estimates exhibit extremely high variance. This variance can be reduced without introducing bias by subtracting a baseline from the reward function in the policy gradient (Weaver & Tao, 2001; Greensmith et al., 2004). Several works have studied the optimal baseline for variance reduction, often using a critic structure to estimate a value function or advantage function for the baseline (Zhao et al., 2012; Silver et al., 2014; Schulman et al., 2016; Wu et al., 2018). Other works have examined variance reduction in the value function using temporal regularization or regularization directly on the sampled gradient variance (Zhao et al., 2015; Thodoroff et al., 2018). However, even with these tools, variance still remains problematically high in reinforcement learning (Islam et al., 2017; Henderson et al., 2018). Our work aims to achieve significant further variance reduction *directly on the policy* using control-based *functional* regularization.

Recently, there has been increased interest in functional regularization of deep neural networks, both in reinforcement learning and other domains. Work by Le et al. (2016) has utilized functional regularization to guarantee smoothness of learned functions, and Benjamin et al. (2018) studied properties of functional regularization to limit function distances, though they relied on pointwise sampling from the functions which can lead to high regularizer variance. In terms of utilizing control priors, work by Johannink et al. (2018) adds a control prior during learning, and empirically demonstrates improved performance. Researchers in Farshidian et al. (2014); Nagabandi et al. (2017) used model-based priors to produce a good *initialization* for their RL algorithm, but did not use regularization during learning.

Another thread of related work is that of safe RL. Several works on model-based RL have looked at constrained learning such that stability is always guaranteed using Lyapunov-based methods (Perkins & Barto, 2003; Chow et al., 2018; Berkenkamp et al., 2017). However, these approaches do not address reward maximization or they overly constrain exploration. On the other hand, work by Achiam et al. (2017) has incorporated constraints (such as stability) into the learning objective, though model-free methods only guarantee approximate constraint satisfaction after a learning period, not during learning (García & Fernández, 2015). Our work proves stability properties throughout learning by taking advantage of the robustness of control-theoretic priors.

3. Problem Formulation

Consider an infinite-horizon discounted Markov decision process (MDP) with deterministic dynamics defined by the tuple (S, A, f, r, γ), where S is a set of states, A is a continuous and convex action space, and $f : S \times A \to S$ describes the system dynamics, which is unknown to the learning agent. The evolution of the system is given by the following dynamical system and its continuous-time analogue,

$$\begin{aligned} s_{t+1} &= f(s_t, a_t) = f^{known}(s_t, a_t) + f^{unknown}(s_t, a_t), \\ \dot{s} &= f_c(s, a) = f_c^{known}(s, a) + f_c^{unknown}(s, a), \end{aligned} \tag{1}$$

where f^{known} captures the known dynamics, $f^{unknown}$ represents the unknowns, $\dot{s}$ denotes the continuous time-derivative of the state s, and $f_c(s, a)$ denotes the continuous-time analogue of the discrete time dynamics $f(s_t, a_t)$. A control prior can typically be designed from the known part of the system model, f^{known}.

Consider a stochastic policy $\pi_\theta(a|s) : S \times A \to [0, 1]$ parameterized by θ. RL aims to find the policy (i.e. parameters, θ) that maximizes the expected accumulated reward $J(\theta)$:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]. \tag{2}$$

Here $\tau \sim \pi_\theta$ is a trajectory $\tau = \{s_t, a_t, ..., s_{t+n}, a_{t+n}\}$ whose actions and states are sampled from the policy distribution $\pi_\theta(a|s)$ and the environmental dynamics (1), respectively. The function $r(s, a) : S \times A \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor.

This work focuses on policy gradient RL methods, which estimate the gradient of the expected return $J(\theta)$ with respect to the policy based on sampled trajectories. We can estimate the gradient, $\nabla_\theta J$, as follows (Sutton et al., 1999),

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta}\Big[\nabla_\theta \log \pi_\theta(\tau) Q^{\pi_\theta}(\tau)\Big] \\ &\approx \sum_{i=1}^{N}\sum_{t=1}^{T}[\nabla_\theta \log \pi_\theta(s_{i,t}, a_{i,t}) Q^{\pi_\theta}(s_{i,t}, a_{i,t})], \end{aligned} \tag{3}$$

where $Q^{\pi_\theta}(s, a) = \mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}) | s_t = s, a_t = a\Big]$. With a good Q-function estimate, the term (3) is a low-bias estimator of the policy gradient, and utilizes the variance-reduction technique of subtracting a baseline from the reward. However, the resulting policy gradient *still* has very high variance with respect to θ, because the

expectation in term (3) must be estimated using a finite set of sampled trajectories. This high variance in the policy gradient, $\text{var}_\theta[\nabla_\theta J(\theta_k)]$, translates to high variance in the updated policy, $\text{var}_\theta[\pi_{\theta_{k+1}}]$, as seen below,

$$\begin{aligned}
&\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\theta_k), \\
&\pi_{\theta_{k+1}} = \pi_{\theta_k} + \alpha \frac{d\pi_{\theta_k}}{d\theta} \nabla_\theta J(\theta_k) + \mathcal{O}(\Delta\theta^2), \\
&\text{var}_\theta[\pi_{\theta_{k+1}}] \approx \alpha^2 \frac{d\pi_{\theta_k}}{d\theta} \text{var}_\theta[\nabla_\theta J(\theta_k)] \frac{d\pi_{\theta_k}}{d\theta}^T \quad \text{for } \alpha \ll 1,
\end{aligned} \tag{4}$$

where α is the user-defined learning rate. It is important to note that the variance we are concerned about is *with respect to the parameters* θ, not the noise in the exploration process.

To illustrate the variance issue, Fig. 1 shows the results of 100 separate learning runs using direct policy search on the OpenAI gym task *Humanoid-v1* (Recht, 2019). Though high rewards are often achieved, huge variance arises from random initializations and seeds. In this paper, we show that introducing a control prior reduces learning variability, improves learning efficiency, and can provide control-theoretic stability guarantees during learning.

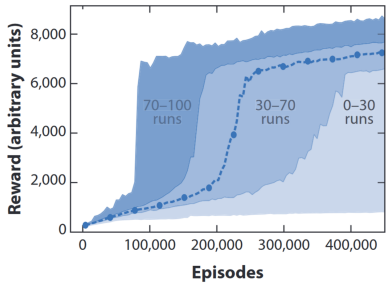

Figure 1. Performance on humanoid walking task from 100 training runs with different initializations. Results from (Recht, 2018).

4. Control Regularization

The policy gradient allows us to optimize the objective from sampled trajectories, but it does not utilize any prior model. However, in many cases we have enough system information to propose at least a crude nominal controller. Therefore, suppose we have a (suboptimal) control prior, $u_{prior} : S \to A$, and we want to combine our RL policy, π_{θ_k}, with this control prior at each learning stage, k. Before we proceed, let us define $u_{\theta_k} : S \to A$ to represent the realized controller sampled from the stochastic RL policy $\pi_{\theta_k}(a|s)$ (we will use u to represent deterministic policies and π to represent the analogous stochastic ones). We propose to combine the RL policy with the control prior as follows,

$$u_k(s) = \frac{1}{1+\lambda} u_{\theta_k}(s) + \frac{\lambda}{1+\lambda} u_{prior}(s), \tag{5}$$

where we assume a continuous, convex action space. Note that $u_k(s)$ is the realized controller sampled from stochastic policy π_k, whose distribution over actions has been shifted by u_{prior} such that $\pi_k\left(\frac{1}{1+\lambda}a + \frac{\lambda}{1+\lambda}u_{prior} \,\middle|\, s\right) = \pi_{\theta_k}(a|s)$. We refer to u_k as the *mixed policy*, and u_{θ_k} as the RL policy.

Utilizing the mixed policy (5) is equivalent to placing a functional regularizer u_{prior} on the RL policy, u_{θ_k}, with regularizer weight λ. Let $\pi_{\theta_k}(a|s)$ be Gaussian distributed: $\pi_{\theta_k} = \mathcal{N}(\overline{u}_{\theta_k}, \Sigma)$, so that Σ describes the exploration noise. Then we obtain the following,

$$\overline{u}_k(s) = \frac{1}{1+\lambda} \overline{u}_{\theta_k}(s) + \frac{\lambda}{1+\lambda} u_{prior}(s), \tag{6}$$

where the control prior, u_{prior} can be interpreted as a Gaussian prior on the mixed control policy (see Appendix A). Let us define the norm $\|u_1 - u_2\|_\Sigma = (u_1 - u_2)^T \Sigma^{-1} (u_1 - u_2)$.

Lemma 1. *The policy $\overline{u}_k(s)$ in Equation (6) is the solution to the following regularized optimization problem,*

$$\begin{aligned}
\overline{u}_k(s) = \arg\min_u \; &\left\| u(s) - \overline{u}_{\theta_k} \right\|_\Sigma \\
&+ \lambda ||u(s) - u_{prior}(s)||_\Sigma, \quad \forall s \in S,
\end{aligned} \tag{7}$$

which can be equivalently expressed as the constrained optimization problem,

$$\begin{aligned}
\overline{u}_k(s) = \arg\min_u \; &\left\| u(s) - \overline{u}_{\theta_k} \right\|_\Sigma \\
\text{s.t.} \quad &||u(s) - u_{prior}(s)||_\Sigma \le \tilde{\mu}(\lambda) \quad \forall s \in S,
\end{aligned} \tag{8}$$

where $\tilde{\mu}$ constrains the policy search. Assuming convergence of the RL algorithm, $\overline{u}_k(s)$ converges to the solution,

$$\begin{aligned}
\overline{u}_k(s) = \arg\min_u \; &\left\| u(s) - \arg\max_{\overline{u}_\theta} \mathbb{E}_{\tau \sim \overline{u}} \left[r(s,a) \right] \right\|_\Sigma \\
&+ \lambda ||u(s) - u_{prior}(s)||_\Sigma, \quad \forall s \in S \;\text{ as }\; k \to \infty
\end{aligned} \tag{9}$$

This lemma is proved in Appendix A. The equivalence between (6) and (7) illustrates that the control prior acts as a functional regularization (recall that $\overline{u}_{\theta_k}$ solves the reward maximization problem appearing in (9)). The policy mixing (6) can *also* be interpreted as constraining policy search near the control prior, as shown by (8). More weight on the control prior (higher λ) constrains the policy search more heavily. In certain settings, the problem can be solved in the constrained optimization formulation (Le et al., 2019).

4.1. CORE-RL Algorithm

Our learning algorithm is described in Algorithm 1. At the high level, the process can be described as:

- First compute the control prior based on prior knowledge (Line 1). See Section 5 for details on controller synthesis.

Algorithm 1 Control Regularized RL (CORE-RL)

1: Compute the control prior, u_{prior} using the known model $f^{known}(s,a)$ (or other prior knowledge)
2: Initialize RL policy π_{θ_0}
3: Initialize array $\mathcal{D}$ for storing rollout data
4: Set $k=1$ (representing k^{th} policy iteration)
5: **while** $k <$ Episodes **do**
6: Evaluate policy $\pi_{\theta_{k-1}}$ at each timestep
7: **if** Using Adaptive Mixing Strategy **then**
8: At each timestep, compute regularization weight λ
9: for the control prior using the TD-error from (12).
10: **else**
11: Set constant regularization weight λ
12: **end if**
13: Deploy mixed policy π_{k-1} from (5) to obtain
14: rollout of state-action-reward for T timesteps.
15: Store resulting data (s_t, a_t, r_t, s_{t+1}) in array $\mathcal{D}$.
16: Using data in $\mathcal{D}$, update policy using any policy
17: gradient-based RL algorithm (e.g. DDPG, PPO)
18: to obtain θ_k.
19: $k = k+1$
20: **end while**
21: **return** Policy π_{θ_k}, u_{prior} ▷ Overall controller

- For a given policy iteration, compute the regularization weight, λ, using the strategy described in Section 4.3 (Lines 7-9). The algorithm can also use a fixed regularization weight, λ (Lines 10-11).
- Deploy the mixed policy (5) on the system, and record the resulting states/action/rewards (Lines 13-15).
- At the end of each policy iteration, update the policy based on the recorded state/action/rewards (Lines 16-18).

4.2. Bias-Variance Tradeoff

Theorem 1 formally states that mixing the policy gradient-based controller, π_{θ_k}, with the control prior, u_{prior}, decreases learning variability. However, the mixing may introduce bias into the learned policy that depends on the (a) regularization λ, and (b) sub-optimality of the control prior. Bias is defined in (10) and refers to the difference between the mixed policy and the (potentially locally) optimal RL policy *at convergence*.

Theorem 1. *Consider the mixed policy (5) where π_{θ_k} is a policy gradient-based RL policy, and denote the (potentially local) optimal policy to be π_{opt}. The variance (4) of the mixed policy arising from the policy gradient is reduced by a factor $(\frac{1}{1+\lambda})^2$ when compared to the RL policy with no control prior.*

However, the mixed policy may introduce bias proportional to the sub-optimality of the control prior. If we let $D_{sub} = D_{TV}(\pi_{opt}, \pi_{prior})$, then the policy bias (i.e. $D_{TV}(\pi_k, \pi_{opt})$) is bounded as follows,

$$D_{TV}(\pi_k, \pi_{opt}) \geq D_{sub} - \frac{1}{1+\lambda} D_{TV}(\pi_{\theta_k}, \pi_{prior})$$
$$D_{TV}(\pi_k, \pi_{opt}) \leq \frac{\lambda}{1+\lambda} D_{sub} \quad \text{as } k \to \infty \tag{10}$$

where $D_{TV}(\cdot,\cdot)$ represents the total variation distance between two probability measures (i.e. policies). Thus, if D_{sub} and λ are large, this will introduce policy bias.

The proof can be found in Appendix B. Recall that π_{prior} is the stochastic analogue to the deterministic control prior u_{prior}, such that $\pi_{prior}(a|s) = \mathbb{1}(a = u_{prior}(s))$ where $\mathbb{1}$ is the indicator function. Note that the bias/variance results apply to the *policy* – not the accumulated reward.

Intuition: Using Figure 2, we provide some intuition for the control regularization discussed above. Note the following:

1) The explorable region of the state space is denoted by the set $\mathcal{S}_{st}$, which grows as λ decreases and vice versa. This illustrates the constrained policy search interpretation of regularization in the state space.
2) The difference between the control prior trajectory and optimal trajectory (i.e. D_{sub}) *may* bias the final policy depending on the explorable region $\mathcal{S}_{st}$ (i.e. λ). Fig 2. shows this difference, and its implications, in state space.
3) If the optimal trajectory is within the explorable region, then we can learn the corresponding optimal policy – otherwise the policy will remain suboptimal.

Points 1 and 3 will be formally addressed in Section 5.

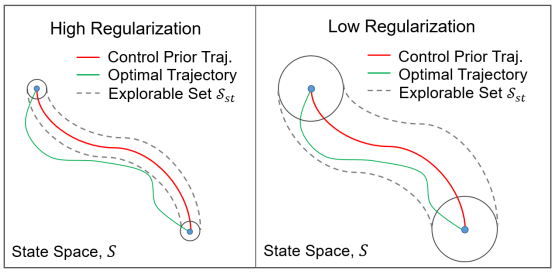

Figure 2. Illustration of optimal trajectory vs. control-theoretic trajectory with the explorable set $\mathcal{S}_{st}$. (a) With high regularization, set $\mathcal{S}_{st}$ is small so we cannot learn the optimal trajectory. (b) With lower regularization, set $\mathcal{S}_{st}$ is larger so we *can* learn the optimal trajectory. However, this also enlarges the policy search space.

4.3. Computing the mixing parameter λ

A remaining challenge is automatically tuning λ, especially as we acquire more training data. While setting a fixed λ can perform well, intuitively, λ should be large when the RL controller is highly uncertain, and it should decrease as we become more confident in our learned controller.

Consider the multiple model adaptive control (MMAC) framework, where a set of controllers (each based on a different underlying model) are generated. A meta-controller computes the overall controller by selecting the weighting for different candidate controllers, based on how close the underlying system model for each candidate controller is to the "true" model (Kuipers & Ioannou, 2010). Inspired by this approach, we should weight the RL controller proportional to our confidence in its model. Our confidence should be state-dependent (i.e. low confidence in areas of the state space where little data has been collected). However, since the RL controller does not utilize a dynamical system model, we propose measuring confidence in the RL controller via the magnitude of the *temporal difference* (TD) error,

$$|\delta^{\pi}(s_t)| = |r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) - Q^{\pi}(s_t, a_t)|, \quad (11)$$

where $a_t \sim \pi(a|s_t)$, $a_{t+1} \sim \pi(a|s_{t+1})$. This TD error measures how poorly the RL algorithm predicts the value of subsequent actions from a given state. A high TD-error implies that the estimate of the action-value function at a given state is poor, so we should rely more heavily on the control prior (a high λ value). In order to scale the TD-error to a value in the interval $\lambda \in [0, \lambda_{max}]$, we take the negative exponential of the TD-error, computed at run-time,

$$\lambda(s_t) = \lambda_{max}\left(1 - e^{-C|\delta(s_{t-1})|}\right). \quad (12)$$

The parameters C and λ_{max} are tuning parameters of the adaptive weighting strategy. Note that Equation (12) uses $\delta(s_{t-1})$ rather than $\delta(s_t)$, because computing $\delta(s_t)$ requires measurement of state s_{t+1}. Thus we rely on the reasonable assumption that $\delta(s_t) \approx \delta(s_{t-1})$, since s_t should be very close to s_{t-1} in practice.

Equation (12) yields a low value of λ if the RL action-value function predictions are accurate. This measure is chosen because the (explicit) underlying model of the RL controller is the value function (rather than a dynamical system model). Our experiments show that this adaptive scheme based on the TD error allows better tuning of the variance and performance of the policy.

5. Control Theoretic Stability Guarantees

In many controls applications, it is crucial to ensure dynamic stability, not just high rewards, during learning. When a (crude) dynamical system model is available, we can utilize classic controller synthesis tools (i.e. LQR, PID, etc.) to obtain a stable control prior in a region of the state space. In this section, we utilize a well-established tool from robust control theory ($\mathcal{H}^{\infty}$ control), to analyze system stability under the mixed policy (5), and prove stability guarantees throughout learning when using a robust control prior.

Our work is built on the idea that the control prior should *maximize robustness to disturbances and model uncertainty*, so that we can treat the RL control, u_{θ_k}, as a performance-maximizing "disturbance" to the control prior, u_{prior}. The mixed policy then takes advantage of the stability properties of the robust control prior, *and* the performance optimization properties of the RL algorithm. To obtain a robust control prior, we utilize concepts from $\mathcal{H}^{\infty}$ control (Doyle, 1996).

Consider the nonlinear dynamical system (1), and let us linearize the *known* part of the model $f_c^{known}(s, a)$ around a desired equilibrium point to obtain the following,

$$\begin{aligned} \dot{s} &= As + B_1 w + B_2 a \\ z &= C_1 s + D_{11} w + D_{12} a \end{aligned} \quad (13)$$

where $w \in \mathbb{R}^{m_1}$ is the disturbance vector, and $z \in \mathbb{R}^{p_1}$ is the controlled output. Note that we analyze the continuous-time dynamics rather than discrete-time, since all mechanical systems have continuous time dynamics that can be discovered through analysis of the system Lagrangian. However, similar analysis can be done for discrete-time dynamics. We make the following standard assumption – conditions for its satisfaction can be found in (Doyle et al., 1989),

Assumption 1. *A $\mathcal{H}^{\infty}$ controller exists for linear system (13) that stabilizes the system in a region of the state space.*

Stability here means that system trajectories are bounded around the origin/setpoint. We can then synthesize an H^{∞} controller, $u^{\mathcal{H}^{\infty}}(s) = -Ks$, using established techniques described in (Doyle et al., 1989). The resulting controller is robust with *worst-case disturbances* attenuated by a factor ζ_k before entering the output, where $\zeta_k < 1$ is a parameter returned by the synthesis algorithm. See Appendix F for further details on $\mathcal{H}^{\infty}$ control and its robustness properties.

Having synthesized a robust $\mathcal{H}^{\infty}$ controller for the *linear system model* (13), we are interested in how those robustness properties (e.g. disturbance attenuation by ζ_k) influence the nonlinear system (1) controlled by the mixed policy (5). We rewrite the system dynamics (1) in terms of the linearization (13) plus a disturbance term as follows,

$$\dot{s} = f_c(s, a) = As + B_2 a + d(s, a), \quad (14)$$

where $d(s, a)$ gathers together all dynamic uncertainties and nonlinearities. To keep this small, we could use feedback linearization based on the nominal nonlinear model (1).

We now analyze stability of the nonlinear system (14) under the mixed policy (5) using Lyapunov analysis (Khalil, 2000). Consider the Lyapunov function $V(s) = s^T P s$, where P is obtained when synthesizing the $\mathcal{H}^{\infty}$ controller (see Appendix F). If we can define a closed region, $\mathcal{S}_{st}$, around the origin such that $\dot{V}(s) < 0$ outside that region, then by standard Lyapunov analysis, $\mathcal{S}_{st}$ is forward invariant and asymptotically stable (note $\dot{V}(s)$ is the time-derivative of the Lyapunov function). Since the $\mathcal{H}^{\infty}$ control law satisfies an Algebraic Riccati Equation, we obtain the following relation,

Lemma 2. *For any state s, satisfaction of the condition,*

$$2s^T P\Big(d(s,a) + \frac{1}{1+\lambda} B_2 u_e\Big) < s^T (C_1^T C_1 + \frac{1}{\zeta_k^2} P B_1 B_1^T P) s,$$

implies that $\dot{V}(s) < 0$.

This lemma is proved in Appendix C. Note that $u_e = u_\theta - u^{\mathcal{H}^\infty}$ denotes the difference between the RL controller and control prior, and (A, B_1, B_2, C_1) come from (13). Let us bound the RL control output such that $\|u_e\|_2 \leq C_\pi$, and define the set $\mathcal{C} = \{(s,u) \in (S,A) \;\big|\; \|u_e\|_2 \leq C_\pi,\ H^\infty \text{ control is stabilizing}\}$. We also bound the "disturbance" $\|d(s,a)\|_2 \leq C_D$, for all $s \in \mathcal{C}$, and define the minimum singular value $\sigma_m(\zeta_k) = \sigma_{min}(C_1^T C_1 + \frac{1}{\zeta_k^2} P B_1 B_1^T P)$, which reflects the robustness of the control prior (i.e. larger σ_m imply greater robustness). Then using Lemma 2 and Lyapunov analysis tools, we can derive a conservative set that is guaranteed asymptotically stable and forward invariant under the mixed policy, as described in the following theorem (proof in Appendix D).

Theorem 2. *Assume a stabilizing H^∞ control prior within the set $\mathcal{C}$ for the dynamical system (14). Then asymptotic stability and forward invariance of the set $\mathcal{S}_{st} \subseteq \mathcal{C}$*

$$\mathcal{S}_{st} : \{s \in \mathbb{R}^n : \|s\|_2 \leq \frac{1}{\sigma_m(\zeta_k)} \Big(2\|P\|_2 C_D + \frac{2}{1+\lambda} \|P B_2\|_2 C_\pi\Big)\ ,\ s \in \mathcal{C}\}. \tag{15}$$

is guaranteed under the mixed policy (5) for all $s \in \mathcal{C}$. The set $\mathcal{S}_{st}$ contracts as we (a) increase robustness of the control prior (increase $\sigma_m(\zeta_k)$), (b) decrease our dynamic uncertainty/nonlinearity C_D, or (c) increase weighting λ on the control prior.

Put simply, Theorem 2 says that all states in $\mathcal{C}$ will converge to (and remain within) set $\mathcal{S}_{st}$ under the mixed policy (5). Therefore, the stability guarantee is stronger if $\mathcal{S}_{st}$ has smaller cardinality. The set $\mathcal{S}_{st}$ is drawn pictorally in Fig. 2, and essentially dictates the explorable region. Note that $\mathcal{S}_{st}$ is *not* the region of attraction.

Theorem 2 highlights the tradeoff between robustness parameter, ζ_k, of the control prior, the nonlinear uncertainty in the dynamics C_D, and the utilization of the learned controller, λ. If we have a more robust control prior (higher $\sigma_m(\zeta_k)$) or better knowledge of the dynamics (smaller C_D), we can heavily weight the learned controller (lower λ) during the learning process while still guaranteeing stability.

While shrinking the set $\mathcal{S}_{st}$ and achieving asymptotic stability along a trajectory or equilibrium point may seem desirable, Fig. 2 illustrates why this is not necessarily the case in an RL context. The optimal trajectory for a task typically deviates from the nominal trajectory (i.e. the control theoretic-trajectory), as shown in Fig. 2 – the set $\mathcal{S}_{st}$ illustrates the explorable region under regularization. Fig. 2(a) shows that we do not want strict stability of the nominal trajectory, and instead would like *limited* flexibility (a sufficiently large $\mathcal{S}_{st}$) to explore. By increasing the weighting on the learned policy π_{θ_k} (decreasing λ), we expand the set $\mathcal{S}_{st}$ and allow for greater exploration around the nominal trajectory (at the cost of stability) as seen in Fig. 2(b).

6. Empirical Results

We apply the CORE-RL Algorithm to three problems: (1) cartpole stabilization, (2) car-following control with experimental data, and (3) racecar driving with the TORCS simulator. We show results using DDPG or PPO or TRPO (Lillicrap et al., 2016; Schulman et al., 2017; Schulman et al., 2015) as the policy gradient RL algorithm (PPO + TRPO results moved to Appendix G), though any similar RL algorithm could be used. All code can be found at `https://github.com/rcheng805/CORE-RL`.

Note that our results focus on reward rather than bias. Bias (as defined in Section 4.2) assumes convergence to a (locally) optimal policy, and does not include many factors influencing performance (e.g. slow learning, failure to converge, etc.). In practice, Deep-RL algorithms often do not converge (or take very long to do so). Therefore, reward better demonstrates the influence of control regularization on performance, which is of greater practical interest.

6.1. CartPole Problem

We apply the CORE-RL algorithm to control of the cartpole from the OpenAI gym environment (*CartPole-v1*). We modified the CartPole environment so that it takes a *continuous* input, rather than discrete input, and we utilize a reward function that encourages the cartpole to maintain its $x-$position while keeping the pole upright. Further details on the environment and reward function are in Appendix E. To obtain a control prior, we assume a crude model (i.e. linearization of the nonlinear dynamics with $\approx 60\%$ error in the mass and length values), and from this we synthesize an $\mathcal{H}^\infty$ controller. Using this control prior, we run Algorithm 1 with several different regularization weights, λ. For each λ, we run CORE-RL 6 times with different random seeds.

Figure 4a plots reward improvement over the control prior, which shows that the regularized controllers perform much better than the baseline DDPG algorithm (in terms of variance, reward, and learning speed). We also see that intermediate values of λ (i.e. $\lambda \approx 4$) result in the best learning, demonstrating the importance of policy regularization.

Figure 4b better illustrates the performance-variance trade-

off. For small λ, we see high variance *and* poor performance. With intermediate λ, we see higher performance and lower variance. As we further increase λ, variance continues to decrease, but the performance also decreases since policy exploration is heavily constrained. The adaptive mixing strategy performs very well, exhibiting low variance through learning, and converging on a high-performance policy.

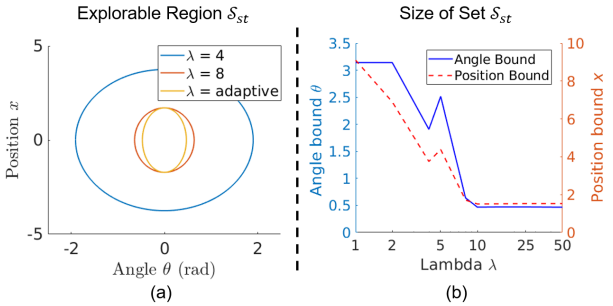

Figure 3. Stability region for CartPole under mixed policy. (a) Illustration of the stability region for different regularization, λ. For each λ shown, the trajectory goes to and remains within the corresponding stability set throughout training. (b) Size of the stability region in terms of the angle θ, and position x. As λ increases, we are guaranteed to remain closer to the equilibrium point during learning.

While Lemma 1 proved that the mixed controller (6) has the same *optimal* solution as optimization problem (7), when we ran experiments directly using the loss in (7), we found that performance (i.e. reward) was worse than CORE-RL and still suffered high variance. In addition, learning with pre-training on the control prior likewise exhibited high variance and had worse performance than CORE-RL.

Importantly, according to Theorem 2, the system should maintain stability (i.e. remain within an invariant set around our desired equilibrium point) *throughout the learning process*, and the stable region shrinks as we increase λ. Our simulations exhibit exactly this property as seen in Figure 3, which shows the *maximum* deviation from the equilibrium point across all episodes. The system converges to a stability region throughout learning, and this region contracts as we increase λ. Therefore, regularization not only improves learning performance and decreases variance, but can capture stability guarantees from a robust control prior.

6.2. Experimental Car-Following

We next examine experimental data from a chain of 5 cars following each other on an 8-mile segment of a single-lane public road. We obtain position (via GPS), velocity, and acceleration data from each of the cars, and we control the acceleration/deceleration of the 4^{th} car in the chain. The goal is to learn an optimal controller for this car that maximizes fuel efficiency while avoiding collisions. The experimental setup and data collection process are described in (Ge et al., 2018). For the control prior, we utilize a bang-bang controller that (inefficiently) tries to maintain a large distance from the car in front and behind the controlled car. The reward function penalizes fuel consumption and collisions (or near-collisions). Specifics of the control prior, reward function, and experiments are in Appendix E.

For our experiments, we split the data into 10 second "episodes", shuffle the episodes, and run CORE-RL six times with different random seeds (for several different λ).

Figure 4a shows again that the regularized controllers perform much better than the baseline DDPG algorithm for the car-following problem, and demonstrates that regularization leads to performance improvements over the control prior and gains in learning efficiency. Figure 4b reinforces that intermediate values of λ (i.e. $\lambda \approx 5$) exhibit optimal performance. Low values of λ exhibit significant deterioration of performance, because the car must learn (with few samples) in a much larger policy search space; the RL algorithm does not have enough data to converge on an optimal policy. High values of λ also exhibit lower performance because they heavily constrain learning. Intermediate λ allow for the best learning using the limited number of experiments.

Using an adaptive strategy for setting λ (or alternatively tuning to an optimal λ), we obtain high-performance policies that improve upon both the control prior and RL baseline controller. The variance is also low, so that the learning process *reliably* learns a good controller.

6.3. Driving in TORCS

Finally we run CORE-RL to generate controllers for cars in *The Open Racing Car Simulator* (TORCS) (Wymann et al., 2014). The simulator provides readings from 29 sensors, which describe the environment state. The sensors provide information like car speed, distance from track center, wheel spin, etc. The controller decides values for the acceleration, steering and braking actions taken by the car.

To obtain a control prior for this environment, we use a simple PID-like linearized controller for each action, similar to the one described in (Verma et al., 2018). These types of controllers are known to have sub-optimal performance, while still being able to drive the car around a lap. We perform all our experiments on the CG-Speedway track in TORCS. For each λ, we run the algorithm 5 times with different initializations and random seeds.

For TORCS, we plot *laptime improvement over the control prior* so that values above zero denote improved performance over the prior. The laps are timed out at 150s, and the objective is to minimize lap-time by completing a lap as fast as possible. Due to the sparsity of the lap-time signal, we use a pseudo-reward function during training that pro-

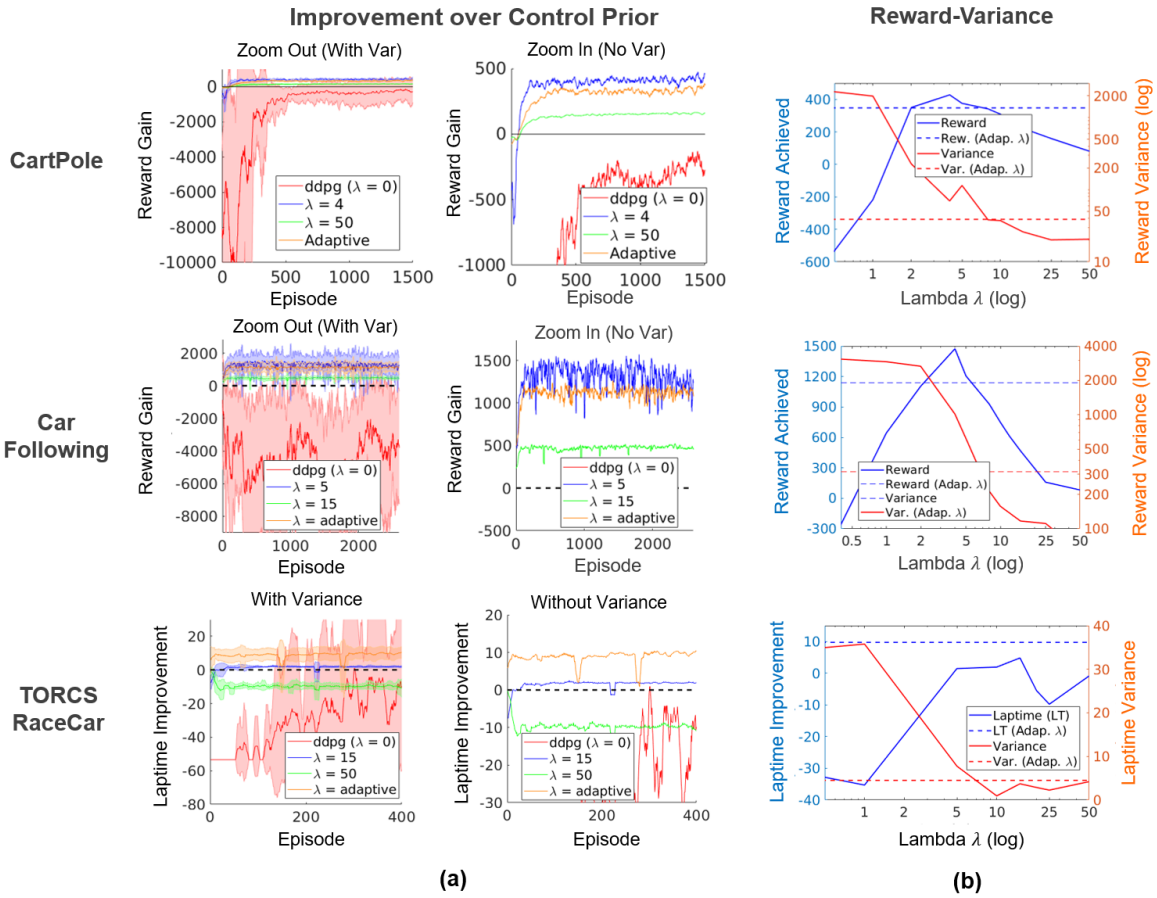

Figure 4. Learning results for CartPole, Car-Following, and TORCS RaceCar Problems using DDPG. (a) Reward improvement over control prior with different set values for λ or an adaptive λ. The right plot is a zoomed-in version of the left plot without variance bars for clarity. Values above the dashed black line signify improvements over the control prior. (b) Performance and variance in the reward as a function of the regularization λ, across different runs of the algorithm using random initializations/seeds. Dashed lines show the performance (i.e. reward) and variance using the adaptive weighting strategy. Variance is measured for all episodes across all runs. Adaptive λ and intermediate values of λ exhibit best learning. Again, performance is baselined to the control prior, so any performance value above 0 denotes improvement over the control prior.

vides a heuristic estimate of the agent's performance at each time step during the simulation (details in Appendix E).

Once more, Figure 4a shows that regularized controllers perform better on average than the baseline DDPG algorithm, and that we improve upon the control prior with proper regularization. Figure 4b shows that intermediate values of λ exhibit good performance, but using the adaptive strategy for setting λ in the TORCS setting gives us the highest-performance policy that significantly beats both the control prior and DDPG baseline. Also, the variance with the adaptive strategy is significantly lower than for the DDPG baseline, which again shows that the learning process *reliably* learns a good controller.

Note that we have only shown results for DDPG. Results for PPO and TRPO are similar for CartPole and Car-following (different for TORCS), and can be found in Appendix G.

7. Conclusion

A significant criticism of RL is that random seeds can produce vastly different learning behaviors, limiting application of RL to real systems. This paper shows, through theoretical results and experimental validation, that our method of control regularization substantially alleviates this problem, enabling significant variance reduction and performance improvements in RL. This regularization can be interpreted as constraining the explored action space during learning.

Our method also allows us to capture dynamic stability properties of a robust control prior to guarantee stability during learning, and has the added benefit that it can easily incorporate different RL algorithms (e.g. PPO, DDPG, etc.). The main limitation of our approach is that it relies on a reasonable control prior, and it remains to be analyzed how bad of a control prior can be used while still aiding learning.

Acknowledgements

This work was funded in part by Raytheon under the Learning to Fly program, and by DARPA under the Physics-Infused AI Program.

References

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained Policy Optimization. In *International Conference on Machine Learning (ICML)*, 2017.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 2017.

Baxter, J. and Bartlett, P. Reinforcement learning in POMDP's via direct gradient ascent. *International Conference on Machine Learning*, 2000.

Benjamin, A. S., Rolnick, D., and Kording, K. Measuring and Regularizing Networks in Function Space. *arXiv:1805.08289*, 2018.

Berkenkamp, F., Turchetta, M., Schoellig, A. P., and Krause, A. Safe Model-based Reinforcement Learning with Stability Guarantees. In *Neural Information Processing Systems (NeurIPS)*, 2017.

Chow, Y., Nachum, O., Duenez-Guzman, E., and Ghavamzadeh, M. A Lyapunov-based Approach to Safe Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Doyle, J. Robust and Optimal Control. In *Conference on Decision and Control*, 1996.

Doyle, J., Glover, K., Khargonekar, P., and Francis, B. State-space solutions to standard H/sub 2/ and H/sub infinity / control problems. *IEEE Transactions on Automatic ControlTransactions on Automatic Control*, 1989. ISSN 00189286. doi: 10.1109/9.29425.

Duan, Y., Chen, X., Schulman, J., and Abbeel, P. Benchmarking Deep Reinforcement Learning for Continuous Control. In *International Conference on Machine Learning (ICML)*, 2016.

Farshidian, F., Neunert, M., and Buchli, J. Learning of closed-loop motion control. In *IEEE International Conference on Intelligent Robots and Systems*, 2014.

García, J. and Fernández, F. A Comprehensive Survey on Safe Reinforcement Learning. *JMLR*, 2015.

Ge, J. I., Avedisov, S. S., He, C. R., Qin, W. B., Sadeghpour, M., and Orosz, G. Experimental validation of connected automated vehicle design among human-driven vehicles. *Transportation Research Part C: Emerging Technologies*, 2018.

Ghosh, D., Singh, A., Rajeswaran, A., Kumar, V., and Levine, S. Divide-and-conquer reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, volume abs/1711.09874, 2018.

Greensmith, E., Bartlett, P., and Baxter, J. Variance reduction techniques for gradient estimates in reinforcement learning. *JMLR*, 2004.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep Reinforcement Learning that Matters. In *AAAI Conference on Artificial Intelligence*, 2018.

Islam, R., Henderson, P., Gomrokchi, M., and Precup, D. Reproducibility of Benchmarked Deep Reinforcement Learning of Tasks for Continuous Control. In *Reproducibility in Machine Learning Workshop*, 2017.

Johannink, T., Bahl, S., Nair, A., Luo, J., Kumar, A., Loskyll, M., Aparicio Ojea, J., Solowjow, E., and Levine, S. Residual Reinforcement Learning for Robot Control. *arXiv e-prints*, art. arXiv:1812.03201, Dec 2018.

Khalil, H. K. *Nonlinear Systems (Third Edition)*. Prentice Hall, 2000.

Kuipers, M. and Ioannou, P. Multiple model adaptive control with mixing. *IEEE Transactions on Automatic Control*, 2010.

Le, H., Kang, A., Yue, Y., and Carr, P. Smooth Imitation Learning for Online Sequence Prediction. In *International Conference on Machine Learning (ICML)*, 2016.

Le, H. M., Voloshin, C., and Yue, Y. Batch policy learning under constraints. In *International Conference on Machine Learning*, 2019.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. *arXiv e-prints*, art. arXiv:1708.02596, Aug 2017.

Perkins, T. J. and Barto, A. G. Lyapunov design for safe reinforcement learning. *Journal of Machine Learning Research*, 2003.

Recht, B. A Tour of Reinforcement Learning: The View from Continuous Control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):253–279, 2019.

Schulman, J., Levine, S., Moritz, P., Jordan, M., and Abbeel, P. Trust Region Policy Optimization. In *International Conference on Machine Learning (ICML)*, 2015.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *International Conference on Learning Representations*, 2016.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal Policy Optimization Algorithms. *arXiv e-prints*, art. arXiv:1707.06347, Jul 2017.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic Policy Gradient Algorithms. *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014.

Sutton, R., McAllester, D., Singh, S. P., and Mansour, Y. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in Neural Information Processing Systems*, 1999.

Thodoroff, P., Durand, A., Pineau, J., and Precup, D. Temporal Regularization for Markov Decision Process. In *Advances in Neural Information Processing Systems*, 2018.

Verma, A., Murali, V., Singh, R., Kohli, P., and Chaudhuri, S. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

Weaver, L. and Tao, N. The Optimal Reward Baseline for Gradient-Based Reinforcement Learning. In *Uncertainty in Artificial Intelligence (UAI)*, 2001.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.

Wu, C., Rajeswaran, A., Duan, Y., Kumar, V., Bayen, A. M., Kakade, S., Mordatch, I., and Abbeel, P. Variance Reduction for Policy Gradient with Action-Dependent Factorized Baselines. In *International Conference on Learning Representations*, 2018.

Wymann, B., Espié, E., Guionneau, C., Dimitrakakis, C., Coulom, R., and Sumner, A. TORCS, The Open Racing Car Simulator. http://www.torcs.org, 2014.

Zhao, T., Hachiya, H., Niu, G., and Sugiyama, M. Analysis and improvement of policy gradient estimation. *Neural Networks*, 2012.

Zhao, T., Niu, G., Xie, N., Yang, J., and Sugiyama, M. Regularized Policy Gradients : Direct Variance Reduction in Policy Gradient Estimation. *Proceedings of the Asian Conference on Machine Learning*, 2015.

Appendix: Control Regularization for Reduced Variance Reinforcement Learning

A. Proof of Lemma 1

Lemma 1. *The policy $\overline{u}_k(s)$ in Equation (6) is the solution to the following regularized optimization problem,*

$$\begin{aligned}\overline{u}_k(s) = \arg\min_{u} \quad & \Big\|u(s) - \overline{u}_{\theta_k}\Big\|^2 \\ & + \lambda||u(s) - u_{prior}(s)||^2, \quad \forall s \in S,\end{aligned} \tag{16}$$

which can be equivalently expressed as the constrained optimization problem:

$$\begin{aligned}\overline{u}_k(s) = \arg\min_{u} \quad & \Big\|u(s) - \overline{u}_{\theta_k}\Big\|^2 \\ \text{s.t.} \quad & ||u(s) - u_{prior}(s)||^2 \leq \tilde{\mu}(\lambda) \quad \forall s \in S,\end{aligned} \tag{17}$$

where $\tilde{\mu}$ constrains the policy search. Assuming convergence of the RL algorithm, $\overline{u}_k(s)$ converges to the solution,

$$\begin{aligned}\overline{u}_k(s) = \arg\min_{u} \quad & \Big\|u(s) - \arg\max_{u_\theta} \mathbb{E}_{\tau\sim u}\Big[r(s,a)\Big]\Big\|^2 \\ & + \lambda||u(s) - u_{prior}(s)||^2, \quad \forall s \in S \quad \text{as} \quad k \to \infty\end{aligned} \tag{18}$$

Proof.

Equivalence between (6) and (16) : Let $\pi_{\theta_k}(a|s)$ be a Gaussian distributed policy with mean $\overline{u}_{\theta_k}(s)$: $\pi_{\theta_k}(a|s) \sim \mathcal{N}(\overline{u}_{\theta_k}(s), \Sigma)$. Thus, Σ describes exploration noise. From the mixed policy definition (6), we can obtain the following Gaussian distribution describing the mixed policy:

$$\begin{aligned}\pi_k(a|s) &= \mathcal{N}(\frac{1}{1+\lambda}\overline{u}_{\theta_k} + \frac{1}{1+\lambda}u_{prior}, \Sigma) \\ &= \frac{1}{c_N}\mathcal{N}\Big(\overline{u}_{\theta_k}(s), (1+\lambda)\Sigma\Big) \cdot \mathcal{N}\Big(u_{prior}(s), \frac{1+\lambda}{\lambda}\Sigma\Big),\end{aligned} \tag{19}$$

where the second equality follows based on the properties of products of Gaussians. Let us define $\|u_1 - u_2\|_\Sigma = (u_1 - u_2)^T\Sigma^{-1}(u_1 - u_2)$, and let $|\Sigma|$ be the determinant of $|\Sigma|$. Then, distribution (19) can be rewritten as the product,

$$\begin{aligned}\mathbb{P}(X(s)) = &-c_1 \exp(-\frac{1}{2(1+\lambda)}\|X(s) - \overline{u}_{\theta_k}(s)\|_\Sigma) \times \\ &- c_1\lambda^{\frac{k}{2}} \exp(-\frac{\lambda}{2(1+\lambda)}\|X(s) - u_{prior}(s)\|_\Sigma) \\ c_1 = &\frac{1}{c_N\sqrt{(2\pi)^k(1+\lambda)^k|\Sigma|}}\end{aligned} \tag{20}$$

where $X(s)$ is a random variable with $\mathbb{P}(X(s))$ representing the probability of taking action X from state s under policy (6). Further simplifying this PDF, we obtain:

$$\begin{aligned}\mathbb{P}(X(s)) = c_2 \exp\Big(&- \|X(s) - \overline{u}_{\theta_k}(s)\|_\Sigma \\ &- \lambda\|X(s) - u_{prior}(s)\|_\Sigma\Big) \\ c_2 = &\frac{\lambda^{\frac{k}{2}}}{c_N(2\pi)^k(1+\lambda)^k|\Sigma|}\end{aligned} \tag{21}$$

Since the probability $\mathbb{P}(X(s))$ is maximized when the argument of the exponential in Equation (21) is minimized, then the maximum probability policy can be expressed as the solution to the following regularized optimization problem,

$$\begin{aligned}\overline{u}_k(s) = \arg\min_{u}(s) \quad & \|u(s) - \overline{u}_{\theta_k}(s)\|_\Sigma + \\ & \lambda\|u(s) - u_{prior}(s)\|_\Sigma, \quad \forall s \in S.\end{aligned} \tag{22}$$

Therefore the mixed policy $\overline{u}_k(s)$ from Equation (6) is the solution to Problem (16) .

Convergence of (16) to (18): Note that $\overline{u}_{\theta_k}$ and π_{θ_k} are parameterized by the same θ_k and represent the iterative solution to the optimization problem $\arg\max_\theta \mathbb{E}_{\tau\sim u_k}\Big[r(\tau)\Big]$ at the latest policy iteration. Thus, assuming convergence of the RL algorithm, we can rewrite problem (22) as follows,

$$\begin{aligned}\overline{u}_k = \arg\min_{u} \quad & \Big\|u(s) - \arg\max_{u_{\theta_k}} \mathbb{E}_{\tau\sim u_k}\Big[r(s,a)\Big]\Big\|^2 \\ & + \lambda||u(s) - u_{prior}(s)||^2, \quad \forall s \in S.\end{aligned} \tag{23}$$

Equivalence between (16) and (17) : Finally, we want to show that the solutions for regularized problem (16) and the constrained optimization problem (17) are equivalent.

First, note that Problem (16) is the dual to Problem (17), where λ is the dual variable. Clearly problem (16) is convex in u. Furthermore, Slater's condition holds, since there is always a feasible point (e.g. trivially $u(s) = u_{prior}(s)$). Therefore strong duality holds. This means that $\exists\lambda \geq 0$ such that the solution to Problem (17) must also be optimal for Problem (16).

To show the other direction, fix $\lambda > 0$ and define $R(u) = \|u(s) - \overline{u}_{\theta_k}(s)\|^2$ and $C(u) = ||u(s) - u_{prior}(s)||^2$ for all $s \in S$. Let us denote u^* as the optimal solution for Problem (16) with $C(u^*) = \tau > \tilde{\mu}$ (note we can choose $\tilde{\mu}$). However supposed u^* is *not* optimal for Problem (17). Then there exists $\tilde{u}$ such that $R(u^*) < R(\tilde{u})$ and $C(\tilde{u}) \leq \tilde{\mu}$. Denote the difference in the two rewards by $R(\tilde{u}) - R(u^*) = R_{diff}$. Thus the following relations hold,

$$R(\tilde{u}) + \lambda C(\tilde{u}) < R(u^*) + \lambda C(u^*) + R_{diff} + \lambda\big[\tilde{\mu} - \tau\big]. \tag{24}$$

This leads to the conditional statement,

$$\begin{aligned} &R_{diff} + \lambda\big[\tilde{\mu} - \tau\big] \geq 0 \\ &\quad \Rightarrow \quad R(\tilde{u}) + \lambda C(\tilde{u}) < R(u^*) + \lambda C(u^*). \end{aligned} \tag{25}$$

For fixed λ, there always exists $\tilde{\mu} > 0$ such that the condition $R_{diff} + \lambda\big[\tilde{\mu} - \tau\big] \geq 0$ holds. However, this leads to a contradiction, since we assumed that u^* is optimal for Problem (16). We can conclude then that $\exists \tilde{\mu}$ such that the solution to Problem (16) must be optimal for Problem (17). Therefore, Problems (16) and (17) have equivalent solutions.

□

B. Proof of Theorem 1

Theorem 1. *Consider the mixed policy (5) where π_{θ_k} is an RL controller learned through policy gradients, and denote the (potentially local) optimal policy to be π_{opt}. The variance (4) of the mixed policy arising from the policy gradient is reduced by a factor $(\frac{1}{1+\lambda})^2$ when compared to the RL policy with no control prior.*

However, the mixed policy may introduce bias proportional to the sub-optimality of the control prior. More formally, if we let $D_{sub} = D_{TV}(\pi_{opt}, \pi_{prior})$, then the policy bias (i.e. $D_{TV}(\pi_k, \pi_{opt})$) is bounded as follows:

$$\begin{aligned} &D_{TV}(\pi_k, \pi_{opt}) \geq D_{sub} - \frac{1}{1+\lambda} D_{TV}(\pi_{\theta_k}, \pi_{prior}) \\ &D_{TV}(\pi_k, \pi_{opt}) \leq \frac{\lambda}{1+\lambda} D_{sub} \qquad \text{as } k \to \infty \end{aligned} \tag{26}$$

where $D_{TV}(\cdot, \cdot)$ represents the total variation distance between two probability measures (i.e. policies). Thus, if D_{sub} and λ are large, this will introduce policy bias.

Proof. Let us define the stochastic action (i.e. random variable) $\mathcal{A}^{act}_{k+1} \sim \pi_{\theta_{k+1}}(a|s)$. Then recall from Equation (4) that assuming a fixed, Gaussian distributed policy, $\pi_{\theta_k}(a|s)$,

$$\text{var}_\theta[\mathcal{A}^{act}_{k+1}|s] \approx \alpha^2 \frac{d\pi_{\theta_k}}{d\theta} \text{var}_\theta[\nabla_\theta J(\theta_k)] \frac{d\pi_{\theta_k}}{d\theta}^T. \tag{27}$$

Based on the mixed policy definition (5), we obtain the following relation between the variance of π_k and π_{θ_k} (the mixed policy and RL policy, respectively),

$$\begin{aligned} \text{var}_\theta[\pi_{k+1}] &= \text{var}_\theta\Big[\frac{1}{1+\lambda}\mathcal{A}^{act}_{k+1} + \frac{\lambda}{1+\lambda}u_{prior}|s\Big] \\ &= \frac{1}{(1+\lambda)^2}\text{var}_\theta[\mathcal{A}^{act}_{k+1}|s] \\ &= \frac{\alpha^2}{(1+\lambda)^2}\frac{d\pi_{\theta_k}}{d\theta}\text{var}_\theta[\nabla_\theta J(\theta_k)]\frac{d\pi_{\theta_k}}{d\theta}^T. \end{aligned} \tag{28}$$

Compared to the variance (4), we achieve a variance reduction when utilizing the same learning rate α. Taking the same policy gradient from (4), var$[\nabla_\theta J(\theta_k)]$, then the variance is reduced by a factor of $(\frac{1}{1+\lambda})^2$ by introducing policy mixing.

Lower variance comes at a price – potential introduction of bias into policy. Let us define the policy bias as $D_{TV}(\pi_k, \pi_{opt})$, and let us denote $D_{sub} = D_{TV}(\pi_{opt}, \pi_{prior})$. Since total variational distance, D_{TV} is a metric, we can use the triangle inequality to obtain:

$$D_{TV}(\pi_k, \pi_{opt}) \geq D_{TV}(\pi_{prior}, \pi_{opt}) - D_{TV}(\pi_{prior}, \pi_k). \tag{29}$$

We can further break down the term $D_{TV}(\pi_{prior}, \pi_k)$:

$$\begin{aligned} &D_{TV}(\pi_{prior}, \pi_k) \\ &\quad = \sup_{(s,a)\in S\text{x}A} \Big|\pi_{prior} - \frac{1}{1+\lambda}\pi_{\theta_k} - \frac{\lambda}{1+\lambda}\pi_{prior}\Big| \\ &\quad = \frac{1}{1+\lambda}\sup_{(s,a)\in S\text{x}A}|\pi_{\theta_k} - \pi_{prior}| \\ &\quad = \frac{1}{1+\lambda}D_{TV}(\pi_{\theta_k}, \pi_{prior}). \end{aligned} \tag{30}$$

This holds for all $k \in \mathbb{N}$. From (29) and (30), we can obtain the lower bound in (26),

$$D_{TV}(\pi_k, \pi_{opt}) \geq D_{sub} - \frac{1}{1+\lambda} D_{TV}(\pi_{\theta_k}, \pi_{prior})$$

To obtain the upper bound, let the policy gradient algorithm with *no* control prior achieve asymptotic convergence to the (locally) optimal policy π_{opt} (as proven for certain classes of function approximators in (Sutton et al., 1999)). Denote this policy as $\pi^{(p)}_{\theta_k}$, such that $\pi^{(p)}_{\theta_k} \to \pi_{opt}$ as $k \to \infty$. In this case, we can derive the total variation distance between

the mixed policy (5) and the optimal policy as follows,

$$\begin{aligned} &D_{TV}(\pi_{opt}, \pi_k^{(p)}) \\ &\quad= \sup_{(s,a)\in SxA} |\pi_{opt} - \frac{1}{1+\lambda}\pi_{\theta_k}^{(p)} - \frac{\lambda}{1+\lambda}\pi_{prior}| \\ &\quad= \frac{\lambda}{1+\lambda} \sup_{(s,a)\in SxA} |\pi_{opt} - \pi_{prior}| \quad \text{as } k \to \infty \\ &\quad= \frac{\lambda}{1+\lambda} D_{TV}(\pi_{opt}, \pi_{prior}) \quad \text{as } k \to \infty \\ &\quad= \frac{\lambda}{1+\lambda} D_{sub} \quad \text{as } k \to \infty. \end{aligned} \tag{31}$$

Note that this represents an *upper bound* on the bias, since it assumes that $\pi_{\theta_k}^{(p)}$ is uninfluenced by π_{prior} during learning. It shows that $\pi_{\theta_k}^{(p)}$ is a feasible policy, but not necessarily optimal when accounting for regularization with π_{prior}. Therefore, we can obtain the upper bound:

$$\begin{aligned} D_{TV}(\pi_{opt}, \pi_k) &\leq D_{TV}(\pi_{opt}, \pi_k^{(p)}) \\ &= \frac{\lambda}{1+\lambda} D_{sub} \quad \text{as } k \to \infty. \end{aligned} \tag{32}$$

□

C. Proof of Lemma 2

Lemma 2. *For any state s, satisfaction of the condition,*

$$2s^T P\Big(d(s,a) + \frac{1}{1+\lambda} B_2 u_e\Big) < s^T (C_1^T C_1 + \frac{1}{\gamma_k^2} P B_1 B_1^T P) s,$$

implies that $\dot{V}(s) < 0$.

Proof. Recall that we are analyzing the Lyapunov function $V(s) = s^T P s$, where P is taken from the Algebraic Riccati Equation (50). Let us take the time derivative of the Lyapunov function as follows:

$$\begin{aligned} \dot{V}(s) = \frac{dV}{ds}\dot{s} &= 2s^T P\Big(As + B_2 a + d(s,a)\Big) \\ &= s^T(-C_1^T C_1 - \frac{1}{\gamma_k^2} P B_1 B_1^T P)s + 2s^T P d(s,a) + \\ &\quad \frac{2}{1+\lambda} s^T P B_2 (u_{\theta_k} - u_{prior}) \\ &= s^T(-C_1^T C_1 - \frac{1}{\gamma_k^2} P B_1 B_1^T P)s + 2s^T P\Big(d(s,a) + \\ &\quad \frac{1}{1+\lambda} B_2 u_e\Big). \end{aligned} \tag{33}$$

The second equality comes from the Algebraic Riccati Equation (50), which the dynamics satisfy by design of the $\mathcal{H}^\infty$ controller. From here, it follows directly that if,

$$2s^T P\Big(d(s,a) + \frac{1}{1+\lambda} B_2 u_e\Big) < s^T (C_1^T C_1 + \frac{1}{\gamma_k^2} P B_1 B_1^T P) s,$$

then $\dot{V}(s) < 0$. □

D. Proof of Theorem 2

Theorem 2. *Assume a stabilizing H^∞ control prior within the set $\mathcal{C}$ for our dynamical system (14). Then asymptotic stability and forward invariance of the set $\mathcal{S}_{st} \subseteq \mathcal{C}$*

$$\begin{aligned} \mathcal{S}_{st} : \{s \in \mathbb{R}^n : \|s\|_2 \leq \frac{1}{\sigma_m(\gamma_k)} \Big(&2\|P\|_2 C_D \\ &+ \frac{2}{1+\lambda}\|PB_2\|_2 C_\pi\Big)\ ,\ s \in \mathcal{C}\}. \end{aligned} \tag{34}$$

is guaranteed under the mixed policy (5) for all $s \in \mathcal{C}$. The set $\mathcal{S}_{st}$ contracts as we (a) increase robustness of the control prior (increase $\sigma_m(\gamma_k)$), (b) decrease our dynamic uncertainty/nonlinearity C_D, or (c) increase weighting λ on the control prior.

Proof.

Step (1): Find a set in which Lemma 2 is satisfied.

Consider the condition in Lemma 2. Since the right hand side is positive (quadratic), we can consider a bound on the stability condition as follows,

$$\begin{aligned} |2s^T P d(s,a) + \frac{2}{1+\lambda} s^T P B_2 u_e| < \\ s^T (C_1^T C_1 + \frac{1}{\gamma_k^2} P B_1 B_1^T P) s. \end{aligned} \tag{35}$$

Clearly any set of s that satisfy condition (35) also satisfy the condition in Lemma 2. To find such a set, we bound the terms in Condition (35) as follows,

$$\begin{aligned} &|2s^T P d(s,a) + \frac{2}{1+\lambda} s^T P B_2 u_e| \\ &\quad\leq\ 2|s^T P d(s,a)| + \frac{2}{1+\lambda}|s^T P B_2 u_e| \\ &\quad\leq\ 2\|s\|_2\|P\|_2 C_D + \frac{2}{1+\lambda}\|s\|_2\|PB_2\|_2 C_\pi, \end{aligned} \tag{36}$$

where the first inequality follows from the triangle inequality; the second inequality uses our bounds on the disturbance, C_D and control input difference C_π, as well as the Cauchy-Schwarz inequality. Now consider the right

hand side of Condition (35). Recall that $\sigma_m(\gamma_k) = \sigma_{min}(C_1^T C_1 + \frac{1}{\gamma_k^2} P B_1 B_1^T P)$, the minimum singular value. Then the following holds,

$$\sigma_m(\gamma_k)\|s\|_2^2 \leq s^T(C_1^T C_1 + \frac{1}{\gamma_k^2} P B_1 B_1^T P)s \tag{37}$$

Using the bounds in (36) and (37), we can say that Condition (35) is guaranteed to be satisfied if the following holds,

$$2\|s\|_2\|P\|_2 C_D + \frac{2}{1+\lambda}\|s\|_2\|PB_2\|_2 C_\pi < \sigma_m(\gamma_k)\|s\|_2^2 \tag{38}$$

The set for which this condition (38) is satisfied can be described by,

$$\mathcal{C} \setminus \mathcal{S}_{st} : \{s \in \mathbb{R}^n : \|s\|_2 > \frac{1}{\sigma_m(\gamma_k)}\Big(2\|P\|_2 C_D + \frac{2}{1+\lambda}\|PB_2\|_2 C_\pi\Big)\ ,\ s \in \mathcal{C}\}. \tag{39}$$

Recall that $\mathcal{C}$ is the set in which the stabilizing $\mathcal{H}^\infty$ controller exists. From Lemma 2, $\dot{V}(s) < 0$ for all $s \in \mathcal{C} \setminus \mathcal{S}_{st}$ described by the set (39).

Step (2): Establish stability and forward invariance of $\mathcal{S}_{st}$.

The Lyapunov function $V(s) = s^T P s$ decreases towards the origin, and we have established that the time derivative of the Lyapunov function is negative for s in set (39). Therefore, any state s described by the set (39) (intersected with $\mathcal{C}$) must move towards the origin (i.e. towards $\mathcal{S}_{st}$). This follows directly from the properties of Lyapunov functions. Therefore, the set $\mathcal{S}_{st}$ described in (34) must be asymptotically stable and forward invariant for all $s \in \mathcal{C}$.

□

E. Description of Experiments

E.1. Experimental Car-Following

In the original car-following experiments, a chain of 8 cars followed each other on an 8-mile segment of a single-lane public road. We obtain position (via GPS), velocity, and acceleration data from each of the cars. We cut this data into 4 sets of chains of 5 cars, in order to maximize the data available to learn from. We then cut this into 10 second "episodes" (100 data points each). We shuffle these training episodes randomly before each run and feed them to the algorithm, which learns the controller for the 4^{th} car in the chain.

The reward function we use in learning is described below:

$$r = -\dot{v}\min(0, a) - 100|G_1(s)| - 50G_2(s),$$

$$G_1(s) = \begin{cases} \frac{1}{s_{front} - s_{curr}} & \text{if } s_{front} - s_{curr} \leq 10 \\ \frac{1}{s_{curr} - s_{back}} & \text{if } s_{curr} - s_{back} \leq 10 \\ 0 & \text{otherwise} \end{cases} \tag{40}$$

$$G_2(s) = \begin{cases} 1 & \text{if } s_{front} - s_{curr} \leq 2 \\ 1 & \text{if } s_{curr} - s_{back} \leq 2 \\ 0 & \text{otherwise} \end{cases}$$

where s_{curr}, s_{front}, and s_{back} denote the position of the controlled car, the car in front of it, and the car behind it. Also, a denotes the control action (i.e. acceleration/deceleration), and $\dot{v}$ denotes the velocity of the controlled car. Therefore, the first term represents the fuel efficiency of the controlled car, and the other terms encourage the car to maintain headway from the other cars and avoid collision.

The control prior we utilize is a simple bang-bang controller that (inefficiently) tries to keep us between the car and front and back. It is described by,

$$a = \begin{cases} 2.5 & \text{if } K_p\Delta s + K_d\Delta v > 0 \\ -5 & \text{if } K_p\Delta s + K_d\Delta v < 0 \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

$$\Delta s = s_{front} - 2s_{curr} - s_{back}$$
$$\Delta v = v_{front} - 2v_{curr} - v_{back}$$

where v_{curr}, v_{front}, and v_{back} denote the velocity of the controlled car, the car in front of it, and the car behind it. We set the constants $K_p = 0.4$ and $K_d = 0.5$. Essentially, the control prior tries to maximize the distance from the car in front and behind, taking into account velocities as well as positions.

E.2. TORCS Racecar Simulator

In its full generality TORCS provides a rich environment with input from up to 89 sensors, and optionally the 3D graphic from a chosen camera angle in the race. The controllers have to decide the values of up to 5 parameters during game play, which correspond to the acceleration, brake, clutch, gear and steering of the car. Apart from the immediate challenge of driving the car on the track, controllers also have to make race-level strategy decisions, like making pit-stops for fuel. A lower level of complexity is provided in the *Practice Mode* setting of TORCS. In this mode all race-level strategies are removed. Currently, so far as we know, state-of-the-art DRL models are capable of racing only in Practice Mode, and this is also the environment that we use. In this mode we consider the input from 29 sensors, and decide values for the acceleration, steering and brake actions.

The control prior we utilize is a linear controller of the form:

$$K_p(\epsilon - o_i) + K_i \sum_{j=i-N}^{i} (\epsilon - o_j) + K_d(o_{i-1} - o_i) \quad (42)$$

Where o_i is the most recent observation provided by the simulator for a chosen sensor, and N is a predetermined constant. We have one controller for each of the actions, acceleation, steering and braking.

The pseudo-reward used during training is given by:

$$r_t = V\cos(\theta) - V\sin(\theta) - V|\texttt{trackPos}| \quad (43)$$

Here V is the velocity of the car, θ is the angle the car makes with the track axis, and `trackPos` provides the position on the track relative to the track's center. This reward captures the aim of maximizing the longitudinal velocity, minimizing the transverse velocity, and penalizing the agent if it deviates significantly from the center of the track.

E.3. CartPole Stabilization

The CartPole simulator is implemented in the OpenAI gym environment ('CartPole-v1'). The dynamics are the same as in the default, as described below,

$$\begin{aligned}
\theta_{t+1} &= x_t + \dot{x}\tau, \\
\dot{\theta}_{t+1} &= \dot{\theta}_t + \Big(\frac{Mg\sin\theta - F\cos\theta - ml\dot{\theta}^2\sin\theta\cos\theta}{\frac{4}{3}Ml - ml\cos^2\theta}\Big)\tau, \\
x_{t+1} &= x_t + \dot{x}\tau, \\
\dot{x}_{t+1} &= \dot{x}_t + \Big(\frac{F + ml\dot{\theta}^2\sin\theta - ml\ddot{\theta}\cos\theta}{M}\Big)\tau,
\end{aligned} \quad (44)$$

where the only modification we make is that the force on the cart can take on a continuous value, $F \in [-10, 10]$, rather than 2 discrete values, making the action space much larger. Since the control prior can already stabilize the CartPole, we also modify the reward to characterize *how well* the control stabilizes the pendulum. The reward function is stated below, and incentivizes the CartPole to keep the pole upright while minimizing movement in the x-direction:

$$r = -100|\theta| - 2x^2. \quad (45)$$

F. Control Theoretic Stability Guarantees

This section in the Appendix goes over the same material in Section 5, but goes into more detail on the $\mathcal{H}^\infty$ problem definition. Consider the linear dynamical system described by:

$$\begin{aligned}
\dot{s} &= As + B_1 w + B_2 a \\
z &= C_1 s + D_{11} w + D_{12} a \\
y &= C_2 s + D_{21} w + D_{22} a
\end{aligned} \quad (46)$$

where $w \in \mathbb{R}^{m_1}$ is the disturbance vector, $u \in \mathbb{R}^{m_1}$ is the control input vector, $z \in \mathbb{R}^{p_1}$ is the error vector (controlled output), $y \in \mathbb{R}^{p_2}$ is the observation vector, and $s \in \mathbb{R}^n$ is the state vector. The system transfer function is denoted,

$$\begin{aligned}
P^s(s) &= \begin{pmatrix} P^s_{11} & P^s_{12} \\ P^s_{21} & P^s_{22} \end{pmatrix} \\
&= \begin{pmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{pmatrix} + \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} (sI - A)^{-1} \begin{bmatrix} B_1 & B_2 \end{bmatrix},
\end{aligned} \quad (47)$$

where A, B_i, C_i, D_{ij} are defined by the system model (46). Let us make the following assumptions,

- The pairs (A, B_2) and (C_2, A) are stabilizable and observable, respectively.
- The algebraic Riccati equation $A^TP + PA + C_1^TC_1 + P(B_2B_2^T - \frac{1}{\gamma_k^2}B_1B_1^T)P = 0$ has positive-semidefinite solution P,
- The algebraic Riccati equation $AP_Y + P_YA^T + B_1^TB_1 = P_Y(C_2C_2^T - \frac{1}{\gamma_k^2}C_1C_1^T)P_Y$ has positive-semidefinite solution P_Y,
- The matrix $\gamma I - P_YP$ is positive definite.

Under these assumptions, we are guaranteed existence of a stabilizing linear $\mathcal{H}^\infty$ controller, $u^{\mathcal{H}^\infty} = -Ks$ (Doyle et al., 1989). The closed-loop transfer function from disturbance, w, to controlled output, z, is:

$$T_{wz} = P^s_{11} + P^s_{12}K(I - P^s_{22}K)^{-1}P^s_{21}. \quad (48)$$

Let $\sigma(\cdot)$ denotes the maximum singular value of the argument, and recall that $\|T_{wz}\|_\infty := \sup_w \sigma(T_{wz}(jw))$. Then the H_∞ controller solves the problem,

$$\min_K \sup_w \sigma(T_{wz}(jw)) = \gamma_k, \quad (49)$$

to give us controller $u^{\mathcal{H}^\infty} = -Ks$. This generates the maximally robust controller so that the *worst-case disturbance* is attenuated by factor γ_k in the system before entering the controlled output. We can synthesize the H_∞ controller using techniques described in (Doyle et al., 1989).

The H_∞ controller is defined as $u^{\mathcal{H}_\infty} = -B_2^TPx$, where P is a positive symmetric matrix satisfying the Algebraic Riccati equation,

$$A^TP + PA + C_1^TC_1 + \frac{1}{\gamma_k^2}PB_1B_1^TP - PB_2B_2^TP = 0, \quad (50)$$

where (A, B_1, B_2, C_1) are defined in (46). The result is that the control law $u^{\mathcal{H}_\infty}$ stabilizes the system with disturbance attenuation $\|T_{wz}\|_\infty \leq \gamma_k$.

Since we are not dealing with a linear system, we need to consider a modification to the dynamics (46) that *linearizes* the dynamics about some equilibrium point and gathers together all non-linearities and disturbances,

$$\dot{s} = f_c(s, a) = As + B_2 a + d(s, a), \tag{51}$$

where $d(s, a)$ captures dynamic uncertainty/nonlinearity as well as disturbances. To keep this small, we could use feedback linearization based on our nominal nonlinear model (1), but this is outside the scope of this work.

Consider the Lyapunov function $V(s) = s^T P s$, where P is taken from Equation (50). We can analyze stability of the uncertain system (14) under the mixed policy (5) using Lyapunov analysis. We can utilize Lemma 2 in this analysis (see Appendix C) in order to compute a set $\mathcal{S}_{st}$ such that $\dot{V}(s) < 0$ in a region outside that set. Satisfaction of this condition guarantees forward invariance of that set (Khalil, 2000), as well as its asymptotic stability (from the region for which $\dot{V}(s) < 0$).

By bounding terms as described in Section 5, we can conservatively compute the set $\mathcal{S}_{st}$ for which $\dot{V}(s) < 0$, which is shown in Theorem 2. See Appendix D for the derivation of the set (i.e. proof of Theorem 2).

G. PPO + TRPO Results

We also ran all experiments using Proximal Policy Optimization (PPO) or Trust Region Policy Optimization (TRPO) in place of DDPG. The results are shown in Figures 5 and 6. The trends mirror those seen in the main paper using DDPG. Low values of λ exhibit significant deterioration of performance, because of the larger policy search space. High values of λ also exhibit lower performance because they heavily constrain learning. Intermediate λ allow for the best learning, with good performance and low variance. Furthermore, adaptive strategies for setting λ allows us to better tune the reward-variance tradeoff.

Note that we do not show results for the TORCS Racecar. This is because we were not able to get the baseline PPO or TRPO agent to complete a lap throughout learning. The code for the PPO, TRPO, and DDPG agent for each environment can be found at `https://github.com/rcheng805/CORE-RL`.

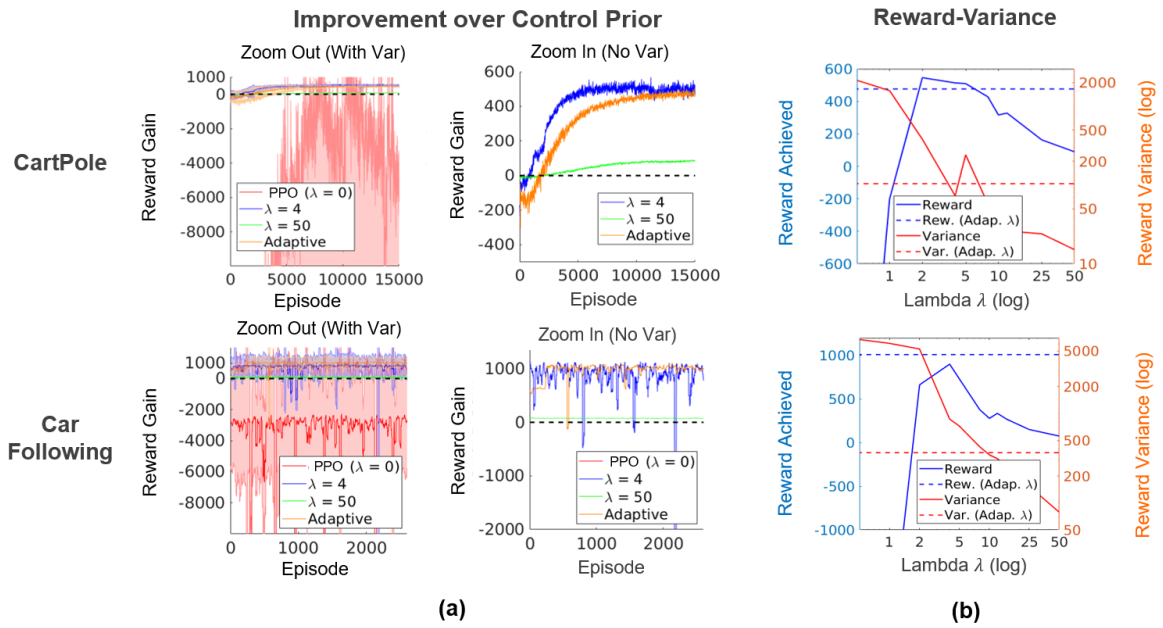

Figure 5. Learning results for CartPole and Car-Following Problems using PPO. (a) Reward improvement over control prior with different set values for λ or an adaptive λ. The right plot is a zoomed-in version of the left plot without variance bars for clarity. Values above the dashed black line signify improvements over the control prior. (b) Performance and variance in the reward as a function of the regularization λ, across different runs of the algorithm using random initializations/seeds. Dashed lines show the performance (i.e. reward) and variance using the adaptive weighting strategy. Variance is measured for all episodes across all runs. Again, performance is baselined to the control prior, so any performance value above 0 denotes improvement over the control prior.

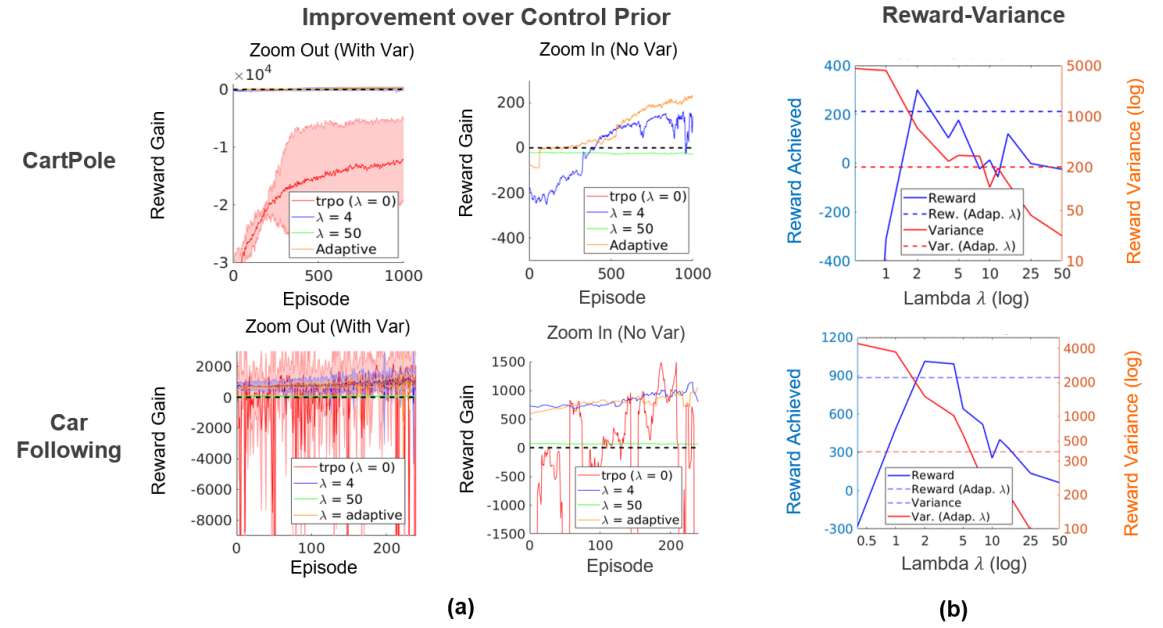

Figure 6. Learning results for CartPole and Car-Following Problems using TRPO. (a) Reward improvement over control prior with different set values for λ or an adaptive λ. The right plot is a zoomed-in version of the left plot without variance bars for clarity. Values above the dashed black line signify improvements over the control prior. (b) Performance and variance in the reward as a function of the regularization λ, across different runs of the algorithm using random initializations/seeds. Dashed lines show the performance (i.e. reward) and variance using the adaptive weighting strategy. Variance is measured for all episodes across all runs. Again, performance is baselined to the control prior, so any performance value above 0 denotes improvement over the control prior.