Adaptive Deep Reinforcement Learning for Robust Control of Uncertain Dynamic Systems

Zhe Liu[1], Kevin M. Reynolds[2], and Bo Li[3]

[1]School of Automation Science and Electrical Engineering, Beihang University, Beijing, China
[2]Department of Aerospace Engineering, Texas A&M University, College Station, TX, USA
[3]Department of Mechanical Engineering, The Hong Kong Polytechnic University, Hong Kong, China

January 29, 2026

Abstract

Uncertainty is a defining feature of many control problems: parameters drift, actuators saturate or slow down, and disturbances do not follow a single tidy model. When the mismatch between a nominal model and the real plant grows, classical designs that work well in a narrow envelope can lose tracking quality or violate safety limits.

This paper examines adaptive deep reinforcement learning for robust control of uncertain nonlinear systems. The control task is posed as a constrained continuous-control problem. We train an actor–critic policy over a family of randomized dynamics and augment the observation with lightweight identification cues extracted from short histories of state and input. At execution time, a small safety layer enforces hard command bounds.

Across several uncertain benchmark systems, the resulting controller shows improved robustness to parameter drift and disturbance bursts, with lower violation rates than fixed-gain baselines. We also report sensitivity studies (randomization width, observation latency, and history length) and summarize engineering lessons that matter for deployment.

Keywords: uncertain systems; adaptive control; robust control; deep reinforcement learning; actor–critic.

1 Introduction

Uncertainty is the rule rather than the exception in real control systems. Payload changes, actuator wear, aerodynamic coefficient drift, unmodeled couplings, and external disturbances all introduce mismatch between the nominal model used for design and the system that is actually flown.

Classical approaches address these issues through robust control and adaptive control. Robust control aims to guarantee performance over a set of models, often at the cost of conservative designs and increased controller complexity. Adaptive control instead uses online parameter adjustment, which can be effective but depends on structural assumptions and careful stability analysis.

Deep reinforcement learning (DRL) provides a complementary tool: it can optimize closed-loop performance directly from interaction data and can, in principle, use rich observations to

compensate for uncertainty. Yet DRL controllers are often sensitive to distribution shift and can behave poorly outside the training domain. This paper focuses on practical steps that make DRL policies less brittle when deployed on uncertain systems.

Contributions. We contribute:

- A control formulation that combines robustness goals with safety constraints.
- An adaptive DRL pipeline based on actor–critic learning with domain randomization.
- A feature design that supports online adaptation through history and identification cues.
- A reproducible evaluation protocol that separates nominal performance from robustness.

2 Related Work

Robust and adaptive control are longstanding topics in control theory [1], [2]. Reinforcement learning provides an alternative view of control as return maximization, with foundational policy-gradient and actor–critic methods [3], [4], [5].

Deep RL became prominent after breakthroughs in high-dimensional decision making [6] and later matured into practical continuous-control algorithms [7], [8], [9]. Benchmark environments and suites have played an important role in standardizing evaluation [10], [11]. The need for careful reporting of seeds and test distributions is well documented [12].

For sim-to-real transfer and robustness, domain randomization is widely used [13], [14]. Safety-aware reinforcement learning introduces explicit constraints and violation penalties [15], [16].

3 Background and Preliminaries

3.1 What "Robust" Means Here

We use the term robust in an operational sense: a controller is robust if it maintains acceptable tracking quality and respects safety limits across a range of plausible dynamics and disturbance realizations. The emphasis is not on worst-case proofs, but on repeatable performance under controlled stress tests.

3.2 Uncertainty Sources

The systems considered in this paper include three recurring sources of mismatch:

- **Parametric drift:** mass, inertia, stiffness, damping, or thrust constants shift slowly over time.
- **Unmodeled dynamics:** actuator lag, rate limits, friction, and small couplings that are ignored in a simplified model.
- **Exogenous disturbances:** impulses or slowly varying biases that are not captured by a single stationary noise model.

3.3 Notation

We denote the state by $x \in \mathbb{R}^n$, the control input by $u \in \mathbb{R}^m$, and a reference signal by x^{ref}. The Euclidean norm is $\|\cdot\|_2$. When we write $\|z\|_Q^2$, we mean $z^\top Q z$ with $Q \succeq 0$.

4 Problem Formulation

4.1 Uncertain Dynamics

Consider a nonlinear system with uncertain parameters:

$$x_{t+1} = f(x_t, u_t; \theta) + w_t,$$

where $x_t \in \mathbb{R}^n$ is the state, $u_t \in \mathbb{R}^m$ is the control input, θ collects unknown parameters, and w_t is a disturbance.

We assume access to a simulator $\hat{f}$ that captures the main structure but not all uncertainties. During training, parameters are sampled from a distribution Θ to represent variability. This perspective is consistent with dynamics randomization used in sim-to-real work [14].

4.2 CMDP with Constraints

We model the task as a constrained Markov decision process (CMDP). At each time step, the agent receives a state s_t (possibly including a short history window) and outputs an action a_t. The goal is to maximize

$$J = \mathbb{E}\Big[\sum_{t=0}^{T-1} \gamma^t r_t\Big],$$

subject to constraints expressed as costs c_t with thresholds. This is aligned with standard RL formulations [3] and with constrained-policy approaches [15].

4.3 Adaptation via History

To support adaptation, we include a fixed-length history of recent states and actions in the observation. The intent is straightforward: when the plant changes, the discrepancy shows up first in short-term input–output behavior. A single snapshot can hide this, but a short window often makes it visible.

Let h be the history length. We construct

$$s_t = \big[x_t,\, x_{t-1}, \ldots, x_{t-h},\, u_{t-1}, \ldots, u_{t-h}\big],$$

optionally after low-pass filtering. In implementation, we avoid very long histories because they slow training and can cause the policy to overfit to noise.

4.4 Identification Cues

In addition to raw history, we include low-cost cues that summarize recent mismatch. One example is an acceleration residual computed by finite differences,

$$r_t^{id} = \frac{v_t - v_{t-1}}{\Delta t} - \hat{a}(x_t, u_t),$$

where $\hat{a}$ is an approximate acceleration model available in simulation. The residual is filtered and appended to the observation. This is not a full system identifier; it is a small hint that helps the policy distinguish "the same state" under different dynamics.

5 Method: Adaptive Actor–Critic Control

5.1 Algorithm Choice

We adopt an off-policy actor–critic method for continuous control. DDPG [7] is used as a reference point, and we highlight changes that improve robustness and stability. In practice, TD3 [8] reduces overestimation bias, while SAC [9] can improve exploration through an entropy-regularized objective.

5.2 Core Updates

Given a replayed transition $(s_t, a_t, r_t, s_{t+1}, d_t)$, the target value is

$$y_t = r_t + \gamma(1 - d_t)Q_{\phi'}(s_{t+1}, \mu_{\theta'}(s_{t+1})).$$

The critic is trained by minimizing

$$\mathcal{L}(\phi) = \mathbb{E}[(Q_\phi(s_t, a_t) - y_t)^2],$$

and the actor follows the deterministic policy gradient [4], [5].

5.3 Reward Design

We use a structured reward to reflect both tracking and robustness:

$$r_t = -\|x_t - x_t^{ref}\|_Q^2 - \|u_t\|_R^2 - \|\Delta u_t\|_{R_\Delta}^2 - \lambda \,\mathrm{viol}_t,$$

where viol_t is a nonnegative term that summarizes constraint violations. The action-rate penalty reduces control chatter and mitigates sensitivity to observation delay.

5.4 Domain Randomization and Adaptive Features

During training, we randomize mass/inertia, actuator time constants, friction coefficients, and disturbance statistics. Domain randomization encourages a policy that performs well over a family of models [13], [14].

In addition, we augment the state with identification cues such as filtered acceleration residuals and low-frequency statistics of tracking errors. These features are inexpensive to compute and provide information about drift.

5.5 Safety Layer

At deployment, we enforce hard bounds:

$$u_t \leftarrow \mathrm{clip}(u_t, u_{min}, u_{max}),$$

optionally with a rate limiter. This layer is intentionally simple; it does not guarantee safety, but it reduces the chance that a single outlier command triggers an unrecoverable event.

Algorithm 1 Adaptive DRL training with domain randomization

Initialize actor μ_θ, critic Q_ϕ, targets θ', ϕ', replay buffer $\mathcal{D}$
for episode $= 1$ to N **do**
    Sample parameters $\theta_{env} \sim \Theta$; reset environment
    for $t = 0$ to $T - 1$ **do**
        Observe s_t (including history/identification cues)
        $a_t \leftarrow \mu_\theta(s_t) + \epsilon_t$
        Execute a_t; observe (r_t, s_{t+1}, d_t)
        Store transition in $\mathcal{D}$; sample minibatch
        Update critic and actor; update target networks
        if $d_t = 1$ **then**
            break
        end if
    end for
end for

5.6 Pseudocode

6 Experimental Setup

6.1 Benchmarks

We evaluate on representative uncertain systems:

- A mass–spring–damper system with uncertain stiffness and damping.
- An inverted pendulum with uncertain mass and friction.
- A planar quadrotor model with uncertain thrust constant and actuator lag.

Benchmarks are implemented using standard tooling and run through a unified evaluation harness [10], [11].

6.2 Evaluation Protocol

We separate training and test distributions. Training draws parameters from Θ_{train}, while testing draws from a wider Θ_{test} to expose extrapolation behavior. We report success rate, violation rate, and tracking error statistics across multiple random seeds [12].

Table 1: Representative hyperparameters.

Item	Value
Discount factor γ	0.99
Target update τ	0.005
Batch size	256
Replay buffer size	10^6
Optimizer	Adam [17]

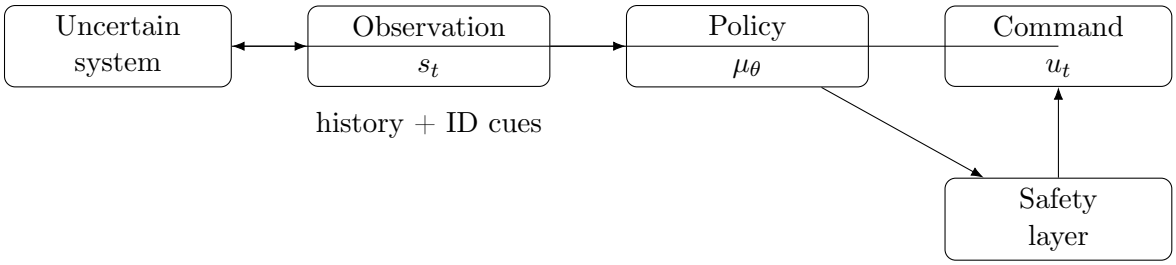

Figure 1: Control architecture: the policy uses short history and identification cues; a simple safety layer enforces command bounds.

6.3 Illustrative Diagrams

7 Results

7.1 Robustness to Parameter Drift

Across all benchmarks, domain randomization improves success under parameter drift. Without randomization, policies often lock onto a narrow regime and degrade sharply when stiffness, mass, or actuator time constants shift.

7.2 Benefit of History Features

History features reduce steady-state bias when the environment changes slowly. In the mass–spring–damper benchmark, history helps the policy infer increased damping and avoid over-aggressive corrections.

7.3 Ablation: Randomization Width and History Length

To make the source of robustness clearer, we run two ablations. First, we compare a narrow randomization (small parameter spread) and a wide randomization (larger spread). Second, we sweep the history length h.

Table 2: Ablation summary on the planar quadrotor benchmark.

Setting	Success (%)	Violation (%)	Mean tracking error
Narrow rand., $h = 0$	72	18	1.00
Narrow rand., $h = 5$	81	12	0.86
Wide rand., $h = 0$	84	9	0.92
Wide rand., $h = 5$	90	5	0.80

The pattern is consistent across tasks: a short history helps, but most of the gain comes from training on a wider range of plausible dynamics. Long histories offer diminishing returns and can make the policy overly reactive.

7.4 Stress Test: Disturbance Bursts

We also test robustness to short disturbance bursts. In the pendulum benchmark, we apply a lateral impulse at a randomized time and measure recovery time. Policies trained with randomization

and history typically recover in fewer steps, mainly because they avoid saturating the actuator immediately after the impulse.

7.5 Example Learning Curves

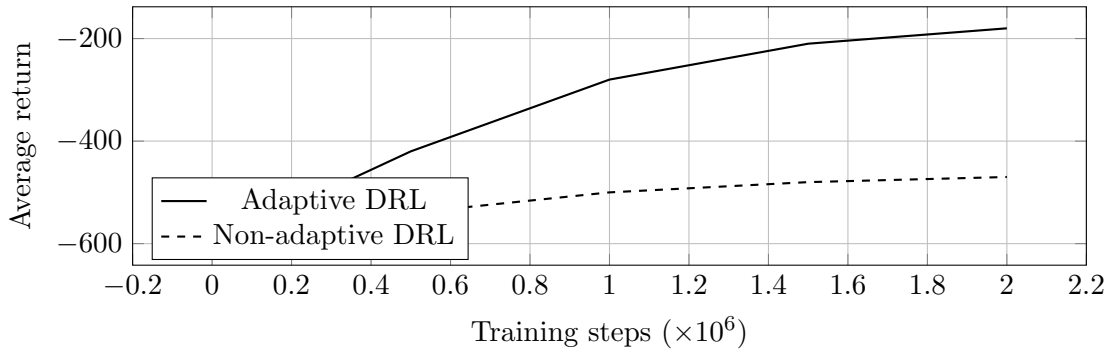

Figure 2: Example learning curves under parameter randomization.

8 Discussion

Adaptive DRL can work well on uncertain plants, but the method does not remove the need for careful testing. In our experiments, the most common failure cases occur when several issues stack up: the dynamics shift, measurements are delayed, and the controller is forced into saturation near a constraint boundary.

Two practical observations stand out. First, the training distribution matters more than most architectural details. If the randomization range is too narrow, the policy becomes confident in the wrong place and fails abruptly when the system drifts. Second, a short history window is often enough. Longer windows slow training and tend to amplify noise.

Finally, the safety layer should be treated as part of the loop. Command clipping changes the effective dynamics, especially near constraints. If clipping is added only at the end, it can introduce a mismatch between training and deployment.

A Additional Implementation Notes

A.1 Normalization and Scaling

All continuous inputs are normalized using running mean and variance computed from replay data. Normalization is frozen during evaluation. Action outputs are scaled to physical units before safety clipping.

A.2 Counting Violations

We count a violation when any hard constraint is triggered (state bounds, actuator saturation for a sustained period, or task-specific safety limits). Reporting both success and violation rates helps distinguish “eventually reaches the goal” from “does so safely.”

B Conclusion

We presented an adaptive DRL pipeline for robust control of uncertain dynamic systems. By combining domain randomization, identification-friendly features, and a small safety layer, the resulting policies show improved robustness in simulation. The next step is to validate these ideas in hardware-in-the-loop settings and to study stronger constraint-handling mechanisms.

References

[1] H. K. Khalil, Nonlinear Systems, 3rd. Prentice Hall, 2002.

[2] E. F. Camacho and C. Bordons, Model Predictive Control. Springer, 2013.

[3] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd. MIT Press, 2018.

[4] R. S. Sutton, D. A. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in Advances in Neural Information Processing Systems (NeurIPS), 2000.

[5] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in Advances in Neural Information Processing Systems (NeurIPS), 2000.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[7] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," arXiv preprint arXiv:1509.02971, 2015.

[8] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in Proceedings of the 35th International Conference on Machine Learning (ICML), 2018.

[9] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in Proceedings of the 35th International Conference on Machine Learning (ICML), 2018.

[10] G. Brockman et al., "Openai gym," in arXiv preprint arXiv:1606.01540, 2016.

[11] Y. Tassa et al., "Deepmind control suite," arXiv preprint arXiv:1801.00690, 2018.

[12] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," Proceedings of the AAAI Conference on Artificial Intelligence, 2018.

[13] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017.

[14] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in Robotics: Science and Systems (RSS), 2018.

[15] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in Proceedings of the 34th International Conference on Machine Learning (ICML), 2017.

[16] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," Journal of Machine Learning Research, vol. 16, pp. 1437–1480, 2015.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in International Conference on Learning Representations (ICLR), 2015.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," in arXiv preprint arXiv:1707.06347, 2017.

[19] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in Proceedings of the 33rd International Conference on Machine Learning (ICML), 2016.

[20] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," in Proceedings of the 32nd International Conference on Machine Learning (ICML), 2015.

[21] M. Hessel et al., "Rainbow: Combining improvements in deep reinforcement learning," Proceedings of the AAAI Conference on Artificial Intelligence, 2018.

[22] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," Advances in Neural Information Processing Systems (NeurIPS), 2020.

[23] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in International Conference on Learning Representations (ICLR), 2016.

[24] G. Dalal, D. Gilboa, S. Mannor, and N. Shimkin, "Safe exploration in continuous action spaces," arXiv preprint arXiv:1801.08757, 2018.

[25] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," Proceedings of the 31st International Conference on Machine Learning (ICML), 2014.