LLM-ACL: Large Language Model-Guided Adversarial Curriculum Learning for Robust Reinforcement Learning

Anonymous Authors[1]

Abstract

Deep reinforcement learning (DRL) has achieved significant success in complex control tasks; however, ensuring robustness against environmental perturbations remains a critical challenge. While Automatic Domain Randomization (ADR) has been introduced to address this issue, existing methods typically rely on fixed hyperparameters or predefined heuristics. These rigid approaches often fail to adapt to the agent's changing learning state. To overcome these limitations, we propose LLM-ACL (Large Language Model-Guided Adversarial Curriculum Learning), a novel framework that leverages the reasoning capabilities of Large Language Models to generate context-aware training strategies. Instead of following a static schedule, LLM-ACL functions as an intelligent instructor that analyzes the agent's training trajectories and statistical indicators in real-time. Based on this analysis, the system dynamically adjusts the environment and reward structure through three adaptive phases: Boost (for acceleration), Maintain (for stabilization), and Perturb (for adversarial robustness). We evaluate our method on continuous control benchmarks, including MuJoCo environments with four distinct difficulty layouts and LunarLander. Experimental results demonstrate that LLM-ACL outperforms both standard RL algorithms and baseline ADR methods, achieving superior performance in nominal environments while significantly enhancing robustness against unexpected perturbations. We provide an open-source implementation to facilitate reproducibility and future research (`https://github.com/lococaeco/DL_Project`).

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

Deep Reinforcement Learning (DRL) has demonstrated remarkable proficiency in solving complex sequential decision-making problems, ranging from high-dimensional continuous control to strategic game playing. However, a critical bottleneck limiting the widespread deployment of DRL agents in real-world scenarios is their lack of *robustness*. Policies trained in stationary simulation environments often suffer from severe performance degradation when exposed to slight environmental variations, sensor noise, or dynamics mismatches, commonly referred to as the *sim-to-real gap*. Consequently, developing robust RL agents capable of maintaining stability across diverse environmental conditions remains an essential challenge for practical application.

To mitigate this issue, techniques such as Domain Randomization (DR) and Automatic Domain Randomization (ADR) have been widely adopted. ADR attempts to gradually adjust the difficulty of the environment—such as the range of physical parameters—based on the agent's performance. However, existing ADR methods suffer from fundamental limitations rooted in their lack of semantic understanding. First, they rely on *fixed heuristics*, depending on predefined thresholds or rigid mathematical formulas to adjust difficulty. Second, they lack *context-awareness*; simple metrics cannot distinguish whether a failure is due to an overly difficult task or the agent's immature policy. Without understanding this learning context, mechanical difficulty adjustments can lead to policy collapse (when tasks become too hard too fast) or learning plateaus (when tasks remain too simple).

We argue that an effective robustness training curriculum must be context-aware, capable of diagnosing the agent's specific state. Inspired by recent research demonstrating that Large Language Models (LLMs) can reason about complex text to assist RL exploration, we extend this paradigm to the domain of *robustness training*. We hypothesize that an LLM can act as an *Intelligent Coach* by analyzing training logs—such as reward trends, failure causes, and action patterns—to determine whether the agent needs a confidence-building "Boost" or a resilience-testing "Perturb."

In this paper, we propose **LLM-ACL (Large Language**

Model-Guided Adversarial Curriculum Learning), a novel framework where an LLM periodically diagnoses the agent and dynamically generates a curriculum. The system adaptively selects one of three strategies: (1) *Boost* to accelerate learning during early stages or performance dips; (2) *Maintain* to preserve stability; and (3) *Perturb* to inject adversarial noise (e.g., action noise, parameter shifts) once the agent reaches proficiency. Notably, we demonstrate that this sophisticated reasoning does not require massive computational resources. Our experiments were conducted using the open-weights `Meta-Llama-3-8B-Instruct` model on a single NVIDIA RTX A5000 GPU, with each training session completed within 2 to 5 hours. This highlights the feasibility and efficiency of our framework for standard research environments.

Our key contributions are summarized as follows:

- We introduce LLM-ACL, a context-aware framework that leverages LLM reasoning to generate dynamic robustness curricula without modifying the underlying RL algorithm.

- We empirically demonstrate that LLM-ACL achieves superior nominal performance and robustness compared to standard PPO and heuristic ADR baselines across variable-difficulty MuJoCo benchmarks and LunarLander environments.

2. Related Work

2.1. Deep Reinforcement Learning

Deep Reinforcement Learning has been widely studied as a framework for sequential decision-making, where an agent learns a policy by interacting with an environment to maximize cumulative reward.

Reinforcement Learning formulate sequential decision-making as an MDP $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $P(s'|s,a)$ the transition dynamics, $r(s,a)$ the reward function, and $\gamma \in (0,1]$ the discount factor.

At each timestep, the agent observes a state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$ according to a policy $\pi(a_t|s_t)$, and receives a scalar reward r_t.

The objective of reinforcement learning is to learn an optimal policy that maximizes the expected cumulative discounted return. Formally, the return is defined as

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k},$$

and the optimal policy π^* is given by

$$\pi^* = \arg\max_{\pi} \ \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right],$$

where $\tau = (s_0, a_0, s_1, a_1, \dots)$ denotes a trajectory induced by policy π. Learning in RL therefore amounts to approximating this optimal policy through interaction with the environment.

Early successes in RL primarily relied on value-based methods such as Q-learning, which are effective in discrete and low-dimensional state and action spaces. However, extending these approaches to continuous control domains poses significant challenges due to the difficulty of action maximization and instability caused by function approximation.

To address continuous action spaces, policy gradient methods and actor-critic architectures have been introduced. On-policy policy gradient methods, such as Proximal Policy Optimization (PPO), optimize policies directly by maximizing a surrogate objective with constrained updates. PPO improves training stability compared to earlier methods like TRPO by using clipped probability ratios, making it simple and robust in practice. However, as an on-policy method, PPO suffers from low sample efficiency since collected data cannot be reused across policy updates, which limits scalability in data-expensive environments.

In contrast, off-policy actor-critic methods aim to improve sample efficiency by reusing past experience. Deterministic Policy Gradient (DPG) methods and their deep variant, Deep Deterministic Policy Gradient (DDPG), combine an actor network for policy representation with a critic network for value estimation, enabling off-policy learning in continuous domains. While DDPG significantly improves sample efficiency over on-policy approaches such as PPO, it is known to suffer from overestimation bias and training instability, making performance highly sensitive to hyperparameter choices.

Twin Delayed Deep Deterministic Policy Gradient (TD3) was proposed to mitigate these issues by introducing three key modifications: clipped double Q-learning to reduce overestimation bias, delayed policy updates to stabilize learning, and target policy smoothing to prevent exploitation of value estimation errors. These improvements significantly enhance robustness and performance over DDPG, establishing TD3 as a strong baseline for off-policy continuous control.

Parallel to deterministic approaches, maximum entropy reinforcement learning has emerged as an alternative paradigm that augments the standard RL objective with an entropy term, encouraging stochasticity in the learned policy. Soft Actor-Critic (SAC) builds on this framework by optimizing a stochastic policy in an off-policy actor-critic setting. By jointly maximizing expected return and policy entropy, SAC promotes efficient exploration and improved robustness. Unlike PPO, which relies on on-policy data and constrained updates for stability, SAC achieves both stability and sample efficiency through off-policy learning and entropy regularization. Compared to DDPG and TD3, SAC explicitly maintains stochastic policies, resulting in superior robustness and reduced sensitivity to hyperparameters across a wide range of continuous control benchmarks.

2.2. Sim-to-Real Transfer

Sim-to-real transfer remains a central challenge in robotics and reinforcement learning, as policies trained in simulation often fail to generalize to real-world environments due to modeling inaccuracies and unmodeled dynamics.

Domain Randomization has emerged as a widely adopted approach to address this issue by training policies under a distribution of simulated environments with randomized physical and visual parameters. Prior work demonstrated that sufficiently diverse randomization of dynamics, textures, lighting, and sensor noise can enable policies to transfer successfully to real robots without explicit system identification.

While domain randomization has shown promising empirical results, its effectiveness critically depends on the choice of randomization ranges, which are typically specified manually. Overly narrow distributions lead to overfitting to simulation, whereas excessively broad distributions can hinder learning and result in unstable or overly conservative policies. This reliance on heuristic tuning limits the scalability and robustness of conventional domain randomization methods.

Automatic Domain Randomization (ADR) was proposed to overcome these limitations by adaptively adjusting the domain randomization distribution during training. Instead of fixing parameter ranges a priori, ADR expands the randomization bounds based on the agent's performance, gradually exposing the policy to increasingly challenging environments. By focusing training on the boundary of the agent's capabilities, ADR effectively constructs a curriculum over environment parameters and can be interpreted as a form of adversarial or worst-case training. This approach has been shown to substantially improve robustness and sim-to-real performance in complex manipulation tasks, such as dexterous in-hand object manipulation.

3. Approach

3.1. Problem Formulation

We consider a standard Markov Decision Process (MDP) defined as

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma),$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ denotes transition probabilities, $\mathcal{R}$ is the reward function, and γ is the discount factor.

To realize the *Intelligent Coach* paradigm, we extend the environment formulation to support both adversarial perturbations and supportive assistance:

$$\mathcal{M}_{\text{curriculum}}(\xi, \beta) = (\mathcal{S}, \mathcal{A}, \mathcal{P}_{\xi}, \mathcal{R}_{\beta}, \gamma),$$

where ξ represents the perturbation parameters (controlling difficulty) and β represents assistance parameters (controlling guidance).

3.2. Framework Architecture

The pipeline operates in a closed loop. The RL agent interacts with the environment, generating interaction logs. The **LLM Curriculum Decider** analyzes these logs to diagnose the agent's learning state and selects a strategic mode: *Boost*, *Maintain*, or *Perturb*. The **Curriculum Environment Wrapper** translates this strategy into specific parameter adjustments for observations, actions, and rewards.

3.3. Multi-Dimensional Curriculum Layer

We define a unified interaction layer that can apply both adversarial noise (Perturb) and supportive shaping (Boost).

3.3.1. Observation Perturbations (Adversarial)

To simulate sensor noise and degrade agent confidence:

- **Additive Gaussian Noise:**

$$\tilde{o}_t = o_t + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_{\text{obs}}^2)$$

- **Observation Dropout:**

$$\tilde{o}_t^{(i)} = \mathbb{I}(u > p_{\text{drop}}) \cdot o_t^{(i)}, \quad u \sim \mathcal{U}(0, 1)$$

3.3.2. Action Perturbations (Adversarial)

To simulate actuator failures and delays:

- **Action Noise & Scaling:**

$$\tilde{a}_t = \alpha \cdot a_t + \delta, \quad \delta \sim \mathcal{N}(0, \sigma_{\text{act}}^2)$$

- **Action Delay:**

$$\tilde{a}_t = a_{t-d}, \quad d \in \{0, \ldots, d_{\max}\}$$

3.3.3. Dynamics Shifts (Adversarial)

We alter physical properties to test generalization:

$$\mu' = \mu \times k_{\text{fric}}, \qquad m' = m \times k_{\text{mass}}.$$

where $k \in [0.5, 1.5]$ represents the scaling factor relative to the nominal physics engine.

3.3.4. ASSISTANCE MECHANISMS (BOOST)

To implement the "Boost" strategy mentioned in the introduction, we introduce supportive interventions when the agent struggles:

- **Physics Simplification:** Reducing complex dynamics (e.g., gravity scaling) to stabilize learning.

$$g' = g \times k_{\text{gravity}}, \quad k_{\text{gravity}} \in (0, 1]$$

- **Reward Shaping:** injecting auxiliary rewards to guide the agent out of local optima.

$$\tilde{r}_t = r_t + \beta \cdot \mathbb{I}(\text{progress condition met})$$

3.4. LLM-Guided Strategic Planning

The LLM acts as a high-level reasoning unit that determines the curriculum strategy σ_{mode} based on the agent's recent history window W.

3.4.1. STRATEGIC MODES

The LLM classifies the agent's status into one of three modes:

1. **Boost (Confidence Building):** Triggered when failure rates satisfy $f_{\text{rate}} > \tau_{\text{high}}$ or reward trends are negative.
   - Action: Set $\xi \to 0$ (remove noise), Set $k_{\text{gravity}} < 1$, Enable $\beta > 0$.
2. **Maintain (Stability Check):** Triggered when performance is stable but not yet robust.
   - Action: Fix ξ at current levels, verify variance $\text{Var}(R) < \epsilon$.
3. **Perturb (Resilience Testing):** Triggered when success rate $S_{\text{rate}} \approx 1.0$.
   - Action: Increase noise $\sigma_{\text{obs}}, \sigma_{\text{act}}$ and apply dynamics shifts $k_{\text{fric}}, k_{\text{mass}}$.

3.4.2. ADAPTIVE CURRICULUM OUTPUT

The LLM outputs a JSON plan $\mathcal{P}$ containing:

$$\mathcal{P} = \{\sigma_{\text{mode}}, \xi_{\text{target}}, \beta_{\text{target}}, \text{constraints}\}$$

This plan is parsed by the wrapper to dynamically adjust the MDP parameters for the next K episodes.

Algorithm: LLM-Guided Adversarial Curriculum Learning

Algorithm 1 LLM-Guided Adversarial Curriculum Learning (LLM-ACL)

```
1: Initialize: Policy $\pi_\theta$, Critic $Q_\phi$, Replay Buffer $\mathcal{D}$
2: Initialize: LLM Decider $\mathcal{L}$, History Window $W \leftarrow \emptyset$
3: for episode $k = 1$ to $M$ do
4:   if $k \mod K == 0$ then
5:     Diagnosis: Analyze $W$ (Reward trend, Failure causes)
6:     Plan Generation: $\mathcal{P} \leftarrow \mathcal{L}(W)$ {LLM selects Boost / Maintain / Perturb}
7:   end if
8:   Parse $\mathcal{P}$ to set environment parameters $\xi$ (noise) and $\beta$ (boost)
9:   Reset environment $s_0$ with physics parameters $\Phi(\xi)$
10:  for step $t = 0$ to $T$ do
11:    Select action $a_t \sim \pi_\theta(o_t)$
12:    // Apply Action Perturbations
13:    $\tilde{a}_t \leftarrow \text{PerturbAction}(a_t, \xi)$
14:    Execute $\tilde{a}_t$, receive nominal reward $r_t$ and state $s_{t+1}$
15:    // Apply Observation Perturbations
16:    $\tilde{o}_{t+1} \leftarrow \text{PerturbObservation}(s_{t+1}, \xi)$
17:    // Apply Boost (Reward Shaping) if active
18:    $\tilde{r}_t \leftarrow r_t + \text{BoostReward}(\beta, s_{t+1})$
19:    Store transition $(o_t, \tilde{a}_t, \tilde{r}_t, \tilde{o}_{t+1})$ in $\mathcal{D}$
20:    Train: Update $\theta, \phi$ (e.g., PPO/SAC)
21:  end for
22:  Update window $W$ with episode statistics
23: end for
```

4. Experiments

In this section, we empirically validate the effectiveness of **LLM-ACL** across diverse continuous control tasks. Our experiments are designed to answer two primary questions: (1) Does LLM-ACL successfully guide agents to robust policies without hindering learning progress? (2) How does LLM-ACL compare against static heuristic-based curriculum methods (ADR) in terms of sample efficiency and final performance?

4.1. Experimental Setup

Environments. We evaluate our framework on four standard continuous control benchmarks to ensure generalizability across varying degrees of dimensionality and dynamic instability.

- **Box2D (LunarLanderContinuous-v3):** Selected as a sanity check for stability in low-dimensional state spaces.

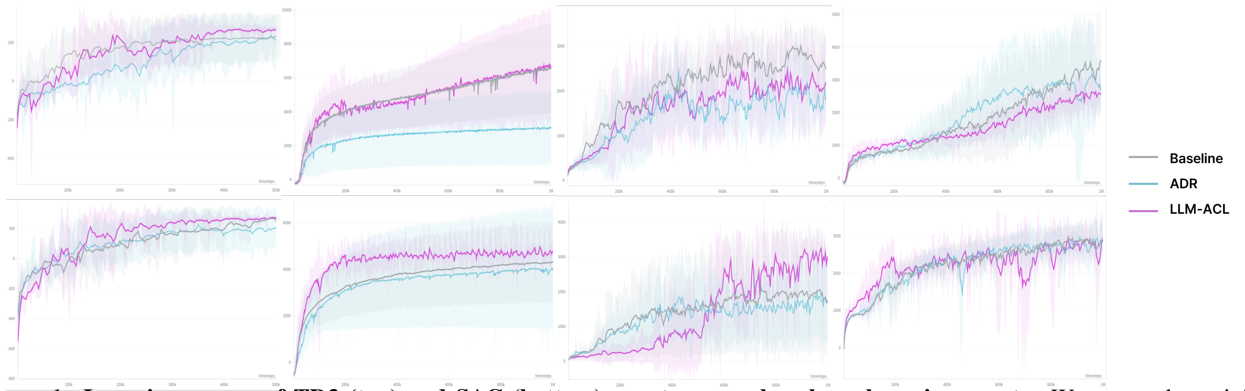

Figure 1. **Learning curves of TD3 (top) and SAC (bottom) agents across benchmark environments.** We report the training performance of Vanilla (Gray), ADR (Blue), and LLM-ACL (Purple) on LunarLander, HalfCheetah, Hopper, and Ant. The upper row corresponds to TD3-based agents, while the lower row corresponds to SAC-based agents. Shaded regions indicate the standard error across four random seeds. Although LLM-ACL exhibits higher variance during early training due to active perturbation injection, it consistently recovers and matches or exceeds the asymptotic performance of both Vanilla and ADR baselines for both TD3 and SAC, demonstrating improved robustness without sacrificing long-term performance.

- **MuJoCo (HalfCheetah-v4, Hopper-v4, Ant-v4):** Selected to represent high-dimensional locomotion tasks with complex contact dynamics. In particular, *Ant-v4* serves as a critical testbed for coordinating multi-joint movements under perturbations.

All agents are trained for **1 million timesteps** to evaluate sample efficiency under a constrained training budget, except for the agent trained on LunarLander-v3, which is trained for **0.5 million timesteps**.

Baselines. We compare LLM-ACL against two distinct baseline categories using four standard RL algorithms (A2C, PPO, TD3, SAC):

1. **Vanilla RL:** Standard algorithms trained in a fixed, nominal environment. This serves as an upper bound for nominal performance but typically lacks robustness.

2. **Automatic Domain Randomization (ADR):** A heuristic-based curriculum baseline that linearly expands the range of perturbation noise (e.g., force, friction) as training progresses. This represents the current standard for robust RL without semantic reasoning.

4.2. Training Dynamics and Nominal Performance

We first analyze the learning trends to verify whether the adversarial interventions by LLM-ACL negatively impact the agent's ability to learn the primary task. Figure 1 illustrates the average return over training steps for TD3 and SAC algorithms across the selected environments.

Adaptation to Perturbations. A key observation from the learning curves is the distinct trajectory of LLM-ACL (Purple). In the early to mid-stages of training, LLM-ACL occasionally exhibits higher variance or slight performance dips compared to the Vanilla baseline (Gray). This phenomenon is attributed to the *Perturb* phase, where the LLM actively injects noise or alters physical parameters to test the agent's resilience. However, unlike unguided randomization which can cause policy collapse, LLM-ACL's context-aware scheduling ensures these perturbations remain within a recoverable range.

Convergence and Stability. Despite the active interference, LLM-ACL demonstrates robust convergence. As shown in the *Ant-v4* and *Hopper-v4* curves, our method successfully catches up to, and often matches, the asymptotic performance of the ADR baseline (Blue). This confirms that the *Boost* and *Maintain* phases effectively counterbalance the adversarial challenges, allowing the agent to consolidate its learning. Consequently, LLM-ACL achieves robustness without sacrificing the fundamental capability to solve the task, effectively bridging the gap between stability and adaptability.

4.3. Analysis

Table 2 highlights that robustness under perturbations cannot be accurately assessed by peak performance alone. While vanilla and ADR-based methods occasionally achieve higher average returns, they frequently suffer from severe performance degradation under unseen dynamics. In contrast, LLM-ACL consistently mitigates such failures, resulting in more stable returns across environments and base algorithms.

In particular, LLM-ACL demonstrates reduced variance and improved worst-case performance in high-dimensional continuous control tasks such as HalfCheetah-v4 and Ant-v4, suggesting that context-aware interventions enable the agent to better adapt to structured disturbances than naive randomization strategies.

Table 1. **Nominal Performance Comparison.** Evaluation of mean and standard deviation over 4 random seeds in standard environments without perturbations. We compare vanilla algorithms, ADR-augmented versions, and our LLM-ACL method.

	Environments			
Method	LunarLander-v3	HalfCheetah-v4	Ant-v4	Hopper-v4
A2C				
Vanilla	-65.12 ± 135.29	85.66 ± 229.26	-32.22 ± 6.17	179.75 ± 35.94
+ ADR	-130.12 ± 12.42	393.07 ± 14.38	-18.76 ± 0.47	68.46 ± 0.83
+ LLM-ACL (Ours)	**-89.53 ± 74.67**	**240.16 ± 372.19**	**-3.97 ± 8.30**	**210.10 ± 60.77**
PPO				
Vanilla	162.00 ± 64.83	1512.78 ± 151.41	1622.60 ± 536.01	3182.61 ± 708.47
+ ADR	254.44 ± 3.64	1477.40 ± 64.66	1959.27 ± 61.20	1074.53 ± 78.51
+ LLM-ACL (Ours)	**201.10 ± 43.48**	**2008.69 ± 1002.08**	**1589.38 ± 649.09**	**2343.56 ± 546.96**
TD3				
Vanilla	283.07 ± 4.29	4324.29 ± 1478.69	2747.63 ± 456.31	1566.38 ± 1553.56
+ ADR	292.13 ± 8.21	4941.03 ± 67.29	2722.34 ± 33.46	1901.97 ± 430.35
+ LLM-ACL (Ours)	**278.09 ± 2.20**	**4320.68 ± 603.18**	**3008.72 ± 338.49**	**3545.54 ± 47.23**
SAC				
Vanilla	203.37 ± 163.15	6927.36 ± 2459.05	3581.61 ± 1426.43	2454.26 ± 964.47
+ ADR	270.55 ± 24.90	1944.48 ± 5.90	2893.11 ± 42.48	1893.24 ± 109.69
+ LLM-ACL (Ours)	**274.60 ± 6.23**	**6920.57 ± 3144.92**	**2759.95 ± 405.86**	**2136.94 ± 907.87**

5. Conclusion

Conventional domain randomization techniques for sim-to-real transfer primarily rely on naive search strategies, such as uniform or random sampling over environment parameters, without leveraging contextual information about the task or the agent. While these approaches can improve robustness to certain types of environmental variability, they remain limited when perturbations directly affect the observation space in structured or semantically meaningful ways.

In this work, we proposed a context-aware domain randomization framework **LLM-ACL** that explicitly incorporates task and agent context into the randomization process. By leveraging the world knowledge of large language models (LLMs), our approach reasons over structural information such as the dimensionality and semantics of observation and action spaces, allowing the agent to identify and prioritize dimensions that are critical for task performance. This enables targeted and meaningful interventions, rather than indiscriminate perturbations, during training.

The **LLM-ACL** is designed to adapt to intervention-driven scenarios, in which perturbations are actively introduced during learning. Through experiments on 4 continuous control tasks, we demonstrated that our method not only performs competitively in nominal environments but also maintains robust performance under various perturbation settings. These results indicate that context-aware domain randomization provides a viable and scalable alternative to conventional randomization strategies, enabling the real-world control.

6. Discussion

Despite these advantages, several limitations remain. First, the current framework relies on predefined safety bounds and curriculum stages, which may require task-specific adjustment in highly complex environments. Second, while LLM-guided reasoning provides flexibility and interpretability, it introduces additional computational cost that may be non-trivial in large-scale or real-time settings. Future work could explore lightweight surrogate models or learned curriculum policies to approximate similar decision-making with reduced cost.

More broadly, extending this framework to multi-task, multi-agent, or real-world robotic settings remains an open challenge. Incorporating richer semantic information—such as task structure or causal relationships—into the curriculum controller may further enhance generalization beyond the perturbations considered in this work.

References

Duda, R. O., Hart, P. E., and Stork, D. G. *Pattern Classification*. John Wiley and Sons, 2nd edition, 2000. https://onlinelibrary.wiley.com/doi/book/10.1002/9780470546411. (Duda et al., 2000).

Fujimoto, S., Van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, 2018. https://arxiv.org/abs/1802.09477. (Fujimoto et al., 2018).

Table 2. **Robustness Performance Comparison.** Evaluation under perturbed environmental dynamics (e.g., mass, friction, action noise). This table demonstrates the stability of each method against unseen variations.

	Environments			
Method	LunarLander-v3	HalfCheetah-v4	Ant-v4	Hopper-v4
A2C				
Vanilla	-119.79 ± 126.92	43.43 ± 213.04	-39.73 ± 7.07	179.66 ± 11.91
+ ADR	-266.74 ± 76.45	271.03 ± 85.71	-22.13 ± 2.06	68.80 ± 3.12
+ LLM-ACL (Ours)	**-103.71 ± 54.99**	**191.38 ± 332.45**	**-3.77 ± 4.97**	**157.79 ± 18.87**
PPO				
Vanilla	138.83 ± 28.89	1392.41 ± 34.17	1498.72 ± 316.73	1418.08 ± 343.08
+ ADR	170.62 ± 60.87	1280.95 ± 151.41	1550.60 ± 375.52	828.78 ± 242.44
+ LLM-ACL (Ours)	**165.17 ± 13.39**	**1861.27 ± 901.32**	**1222.47 ± 509.87**	**1026.05 ± 239.68**
TD3				
Vanilla	156.77 ± 6.52	2705.05 ± 848.83	992.44 ± 187.90	810.37 ± 632.59
+ ADR	185.00 ± 80.28	2589.43 ± 907.54	1299.47 ± 956.94	1078.07 ± 520.99
+ LLM-ACL (Ours)	**145.64 ± 10.79**	**3100.38 ± 495.47**	**1037.13 ± 230.86**	**1287.65 ± 65.12**
SAC				
Vanilla	124.83 ± 59.68	3727.03 ± 1222.21	1596.36 ± 506.25	1074.72 ± 331.34
+ ADR	171.81 ± 81.40	1268.92 ± 392.20	1398.03 ± 370.04	807.10 ± 262.90
+ LLM-ACL (Ours)	**142.77 ± 5.45**	**3733.55 ± 1395.92**	**1517.53 ± 222.76**	**910.02 ± 107.95**

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018. https://arxiv.org/abs/1801.01290. (Haarnoja et al., 2018).

OpenAI, Andrychowicz, M., Baker, B., et al. Learning dexterous in-hand manipulation. *International Journal of Robotics Research (IJRR)*, 2020. https://arxiv.org/abs/1808.00177. (OpenAI et al., 2020).

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018. https://arxiv.org/abs/1710.06537. (Peng et al., 2018).

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. https://arxiv.org/abs/1707.06347. (Schulman et al., 2017).

Wang, X., Xu, Y., Ma, Y., Zhang, H., and Levine, S. Llm-explorer: Self-directed exploration with large language models. *arXiv preprint arXiv:2404.03626*, 2024. https://arxiv.org/abs/2404.03626. (Wang et al., 2024).

A. Appendix

A.1. LLM Prompt for Adaptive Curriculum Control

```
=== CURRENT STATUS ===
Total Episodes: 1240
Learning Phase: mid
Target Reward: 500

Performance:
  Mean Reward: 362.4 (72.5%)
  Std Reward:   85.1
  Min/Max:     120.0 / 498.3

Health Indicators:
  Fail Rate:    18.7%
  Reward Trend: plateau
  Robustness:   0.54 (1.0 = untested, <0.6 = weak)

Recent Episodes:
  1236: r=351.2, fail=False
  1237: r=365.9, fail=False
  1238: r=340.7, fail=True
  1239: r=372.4, fail=False
  1240: r=382.0, fail=False

=== DECISION RULES ===
BOOST:    mean_reward < 150 OR fail_rate > 50% OR early phase.
MAINTAIN: reward_trend == increasing.
PERTURB:  mean_reward > 350 with low robustness OR plateau.

=== YOUR TASK ===
Suggested mode: PERTURB (plateau + low robustness)

Output a JSON object with your decision.
BOOST:   set reward_scale (1.0--1.5), exploration_bonus (0.0--0.2)
PERTURB: set episode_plan: obs_noise, action_noise, action_delay

Example:
{"mode":"boost","reasoning":"early failure","stage":0,
 "boost_config":{"reward_scale":1.2,"exploration_bonus":0.1},
 "perturb_config":{"episode_plan":[]}}

Your JSON response:
```

Figure 2. **Prompt used for the LLM-based Adaptive Curriculum Controller (LLM-ACL).** The prompt summarizes the agent's training status, performance statistics, robustness indicators, and predefined decision rules, and requests a structured JSON response specifying the next intervention strategy.

A.2. Training and Curriculum Hyperparameters

Table 3 summarizes the global training configuration, agent settings, and curriculum-related hyperparameters used in all experiments. Unless otherwise stated, these values are kept fixed across environments and algorithms to ensure fair and consistent comparisons. In particular, the same training budget, evaluation protocol, and logging configuration are applied to both baseline methods and the proposed LLM-ACL framework.

Table 3. Global training and ADR hyperparameters.

Category	**Parameter**	**Value / Description**
General	Seed	42
	Total timesteps	1,000,000
	Evaluation frequency	10,000 steps
	Checkpoint frequency	500,000 steps
	Logging interval	1,000 steps
	Device	Auto
ADR	Enabled	False
	Update frequency	2,048 steps
	Performance threshold	0.8
	Expansion rate	0.05
	Initial randomization range	0.0

A.3. Environment Configuration

Table 4. Environment specifications for continuous control benchmarks. Observation and action dimensions follow the standard Gymnasium implementations. The reward scale indicates a typical performance range considered satisfactory for a well-trained agent under nominal conditions.

Environment	Observation Space	Action Space	Reward Type	Satisfactory Reward
LunarLander-v3	$(8,)$	$(4,)$	Dense, shaped	≥ 200
HalfCheetah-v4	$(17,)$	$(6,)$	Dense (velocity-based)	$\geq 4{,}000$
Hopper-v4	$(11,)$	$(3,)$	Dense + termination penalty	$\geq 3{,}000$
Ant-v4	$(105,)$	$(8,)$	Dense (locomotion-based)	$\geq 5{,}000$

A.4. Training Details

To improve training efficiency under perturbed conditions, we introduce the following design components:

(1) Bounded Perturbations via Noise Clipping. We apply stochastic perturbations to both the observation and action spaces while explicitly bounding their magnitudes through clipping. This prevents extreme deviations that could otherwise destabilize early-stage learning or induce uninformative exploration. By restricting perturbations to a controlled range, the agent is able to maintain meaningful state-action transitions while still experiencing sufficient variability for robustness learning.

(2) Curriculum-Based Domain Randomization Scheduling. Rather than applying full-scale domain randomization from the beginning of training, we schedule the perturbation range in a curriculum-based manner. The agent is initially trained in near-nominal environments and is gradually exposed to increasingly challenging perturbations as training progresses. This staged exposure enables the agent to first acquire stable baseline behaviors before adapting to more difficult variations.

(3) Stabilized Exploration under Perturbations. The combination of bounded noise and progressive randomization ensures that exploration remains stable and informative throughout training. In contrast to naive randomization strategies, which can overwhelm the agent with excessive variability, our approach maintains a balance between exploration and policy stability.

Impact on Training Efficiency. Together, these components lead to faster convergence and improved sample efficiency compared to approaches that apply unbounded or fully randomized perturbations from the outset of training.

A.5. Agent Configuration

Table 5. Unified hyperparameter configuration for all baseline algorithms. Unless otherwise specified, hyperparameters are kept fixed across environments to ensure fair comparisons.

Parameter	**A2C**	**PPO**	**TD3**	**SAC**
Algorithm type	On-policy	On-policy	Off-policy	Off-policy
Learning rate	7×10^{-4}	3×10^{-4}	1×10^{-3}	3×10^{-4}
Batch size	–	64	100	256
Rollout steps (n_{steps})	5	2048	–	–
Epochs per update	–	10	–	–
Replay buffer size	–	–	1,000,000	1,000,000
Learning starts	–	–	100	100
Discount factor (γ)	0.99	0.99	0.99	0.99
GAE parameter (λ)	1.0	0.95	–	–
Policy delay	–	–	2	–
Target policy noise	–	–	0.2	–
Target noise clip	–	–	0.5	–
Soft update coefficient (τ)	–	–	0.005	0.005
Entropy coefficient	0.0	0.0	–	Auto
Target entropy	–	–	–	Auto
Value function coefficient	0.5	0.5	–	–
Max gradient norm	0.5	0.5	–	–
Optimizer	RMSProp	Adam	Adam	Adam
Advantage normalization	Disabled	Enabled	–	–

A.6. Curriculum Configuration

Table 6. ALRT (Adaptive LLM-Guided Robustness Training) curriculum and mode configuration.

Category	**Parameter**	**Value / Description**
General	Enabled	True
	LLM update frequency	Every 5 episodes
	History window size	15 episodes
	Max episode horizon	1000 steps
	Max curriculum stage	3
	Initial stage	0
Failure Criterion	Reward failure threshold	< 0.0
Robustness Eval	Evaluation frequency	Every 20 episodes
	Evaluation episodes	3
BOOST Mode	Reward below ratio	< 0.3 of target
	Failure rate above	> 0.5
PERTURB Mode	Reward above ratio	> 0.7 of target
	Robustness below	< 0.6
	Reward plateau window	10 episodes
Target Rewards	HalfCheetah-v4	16,000
	Hopper-v4	3,000
	Walker2d-v4	5,000
	Ant-v4	6,000
	Humanoid-v4	6,000
	LunarLanderContinuous-v3	250
BOOST Config	Reward scaling	1.2
	Exploration bonus	0.1
	Action hint	Disabled
PERTURB Config	Intensity by stage	$\{0 : 0.3,\ 1 : 0.5,\ 2 : 0.7,\ 3 : 1.0\}$