

# DORAEMON+: Enhancing Adaptive Domain Randomization via Performance-Gated Warmup

1<sup>st</sup> Alessandro Benvenuti

*M.Sc. Degree in Computer Engineering  
Politecnico di Torino  
Turin, Italy  
s343748@studenti.polito.it*

2<sup>nd</sup> Irene Bartolini

*M.Sc. Degree in Computer Engineering  
Politecnico di Torino  
Bologna, Italy  
s345905@studenti.polito.it*

**Abstract**—Bridging the reality gap remains a critical challenge in deploying Deep Reinforcement Learning (DRL) policies onto physical systems. While domain adaptation methods have shown promise, they often suffer from initial instability and slow convergence during the transfer phase. This paper introduces an enhanced sim-to-real transfer framework based on Doraemon, augmented with a novel initial warmup strategy. This warmup phase is designed to stabilize the policy initialization, effectively mitigating the distribution shift between simulated and real-world dynamics before full adaptation begins. We validate our approach on high-dimensional continuous control tasks, specifically using the MuJoCo Hopper and Half-Cheetah environments. Experimental results demonstrate that the proposed Doraemon-with-warmup architecture significantly reduces transfer variance and achieves higher asymptotic performance compared to standard baselines. These findings suggest that structured initialization is a key component for robust sim-to-real deployment in complex locomotive agents.

## I. INTRODUCTION

Deep Reinforcement Learning (DRL) has emerged as a powerful paradigm for continuous control in high-dimensional robotic systems, enabling agents to learn complex motor skills through interaction with simulated environments. However, policies trained in simulations often fail to generalize to physical hardware, a phenomenon known as the **reality gap**. This failure is primarily driven by discrepancies in physical properties (e.g., mass, friction, damping) and unmodeled dynamics that are not captured during the simulation phase.

To mitigate this gap, **Domain Randomization (DR)** [1] techniques are widely employed. By varying dynamics parameters during training, DR aims to create a policy that is invariant to environmental fluctuations. However, the efficacy of DR is heavily dependent on the choice of sampling distribution. In this paper, we investigate more sophisticated strategies to automate the discovery of robust parameters. We specifically compare traditional Uniform Domain Randomization (UDR), that relies on predefined, static ranges, against **DORAEMON** (**Domain Randomization via Entropy Maximization**), an active domain randomization framework. Unlike static methods, DORAEMON treats the sampling distribution as a dynamic entity, utilizing a constrained optimization approach to maximize the entropy of environmental parameters while maintaining a baseline level of agent performance.

Our experiments focus on two MuJoCo environments, **Hopper** and **HalfCheetah**. These environments provide rigorous benchmarks for evaluating how different DR methods handle significant deviations in physical parameters.

## II. RELATED WORK

Sim-to-real transfer aims to bridge the discrepancy between simulated physics and real-world dynamics, often referred to as the reality gap. A popular approach is Domain Randomization (DR), which varies simulator parameters (e.g., friction, mass) to train robust policies. However, standard DR requires tedious manual tuning of the parameter distributions. If the distribution is too narrow, the policy fails to generalize; if it is too wide, the policy becomes over-conservative and performance collapses.

To address the limitations of manual tuning, recent works have proposed automating the DR process. Methods like Automatic Domain Randomization (ADR) [2] seek to automate the curriculum by iteratively expanding the randomization boundaries based on policy performance. However, ADR is inherently restricted to uniform distributions and suffers from data inefficiency.

DORAEMON (Domain Randomization via Entropy Maximization) [3] addresses this by treating the problem as a constrained optimization task. Instead of matching a target, it explicitly maximizes the entropy of the dynamics distribution (i.e., the diversity of the environment). Crucially, it constrains this growth using a ‘success rate’ threshold, ensuring the task difficulty never exceeds the agent’s current capabilities. This allows the agent to automatically discover the widest possible range of solvable dynamics without real-world data.

The gradual expansion of dynamics in DORAEMON implicitly resembles Curriculum Learning, where task difficulty is adjusted over time. Previous works in Self-Paced Learning [4], [5] have shown that guiding the agent from easy to hard tasks stabilizes training.

However, even with automatic entropy maximization, the initial exposure to randomized dynamics can destabilize the policy before it establishes a robust manifold. While DORAEMON manages the growth of the distribution, our work explicitly introduces an initial warmup phase to calibrate the agent’s internal dynamics before the entropy maximization process

begins. This aligns with findings that structured initialization prevents early divergence in high-dimensional spaces.

### III. METHOD

In this work, we address the sim-to-real transfer problem by modeling the simulator as a distribution of Markov Decision Processes (MDPs) parameterized by dynamics parameters  $\xi$  (e.g., leg mass, friction, damping). Our goal is to find a policy  $\pi_\theta$  that is robust across a maximal range of these parameters.

#### A. Domain Randomization via Entropy Maximization

We adopt the DORAEMON framework, which automates the selection of the training distribution  $\nu_\phi(\xi)$ . Rather than manually tuning fixed boundaries, DORAEMON treats the problem as a constrained optimization task by maximizing the entropy of the distribution  $\mathcal{H}(\nu_\phi)$ , which represents the environment diversity, while ensuring the policy maintains a minimum probability of success.

Let  $\sigma(\tau) \in \{0, 1\}$  be a binary indicator function defining whether a trajectory  $\tau$  is successful. The expected probability of success under the distribution  $\nu_\phi$  is defined as:

$$\mathcal{G}(\theta, \phi) = \mathbb{E}_{\xi \sim \nu_\phi(\xi)} [\mathbb{E}_{\tau \sim p_\theta(\tau|\xi)} [\sigma(\tau)]] \quad (1)$$

The optimization objective is to maximize entropy subject to a success threshold  $\alpha \in [0, 1]$ :

$$\max_{\phi} \mathcal{H}(\nu_\phi) \quad \text{s.t.} \quad \mathcal{G}(\theta, \phi) \geq \alpha \quad (2)$$

To solve this efficiently during training, we employ a Lagrangian relaxation method. The objective function minimized at each update step is:

$$\min_{\phi, \lambda} \mathcal{L}(\phi, \lambda) = \min_{\phi, \lambda} -(\mathcal{H}(\nu_\phi) + \lambda \cdot (\mathcal{G}(\theta, \phi) - \alpha)) \quad (3)$$

where  $\lambda \geq 0$  is a Lagrangian multiplier that automatically adjusts the penalty based on the agent's performance.

#### B. Performance-Gated Warmup Initialization

During the initial phase of training, a critical vulnerability of the standard DORAEMON objective (2) arises. At initialization, a randomly initialized policy  $\pi_\theta$  typically exhibits a success rate  $\mathcal{G}(\theta, \phi_{\text{init}}) \approx 0$  across the parameter space. If we attempted to update the distribution parameters  $\phi$  immediately using (3), the gradient estimation would fail. Specifically, with zero successful trajectories ( $\forall \tau, \sigma(\tau) = 0$ ), the REINFORCE signal becomes uninformative:

$$\nabla_\phi J \propto \sum_{\xi} (\underbrace{\sigma(\tau) - b}_{\approx 0}) \nabla_\phi \log \nu_\phi(\xi) \approx 0 \quad (4)$$

Under these conditions, the gradients cannot guide the distribution toward "feasible" regions, leading to random updates or stagnation where the Lagrangian multiplier  $\lambda$  increases indefinitely without effect.

To resolve this, we implement a *Performance-Gated Warmup* that enforces a static distribution until a stability condition is met. We define a boolean state  $W_{\text{complete}}$  (initially false) and a warmup threshold  $\alpha_{\text{warmup}}$  (where typically  $\alpha_{\text{warmup}} \approx \alpha$ ). The update rule for  $\phi$  is conditional:

- 1) **Static Phase** ( $W_{\text{complete}} = \text{False}$ ): While the empirical success rate is  $\hat{\mathcal{G}} < \alpha_{\text{warmup}}$ , we strictly freeze the distribution parameters  $\phi_{t+1} \leftarrow \phi_t$ . This forces the policy to train exclusively on the nominal dynamics until it establishes a robust manifold capable of solving the base task.
- 2) **Activation** ( $W_{\text{complete}} \leftarrow \text{True}$ ): At the moment  $\hat{\mathcal{G}} \geq \alpha_{\text{warmup}}$  for the first time, the warmup phase is permanently concluded.
- 3) **Adaptive Phase** ( $W_{\text{complete}} = \text{True}$ ): For all subsequent steps, we apply the standard DORAEMON update (3) regularly, allowing the distribution to expand or contract based on the dual optimization of entropy and success.

This "latching" mechanism ensures that entropy maximization is only triggered once non-zero gradients are available to meaningfully guide the expansion of the reality gap.

#### C. Algorithmic Implementation

To solve the optimization problem in (3), we parameterize the distribution  $\nu_\phi$  using differentiable probability distributions. We implemented two distinct variants to handle different types of dynamics parameters:

- 1) **Beta Distribution** ( $\nu_\phi = \text{Beta}(\alpha_\phi, \beta_\phi)$ ): Used for bounded parameters. The parameters  $\phi = \{\alpha_\phi, \beta_\phi\}$  are optimized in log-space to strictly enforce positivity. The Beta distribution is particularly effective for Sim-to-Real as it allows the distribution to shift from a peaked Gaussian-like shape to a uniform distribution over the bounded support.
- 2) **Gaussian Distribution** ( $\nu_\phi = \mathcal{N}(\mu, \sigma)$ ): Used for unbounded parameters. We optimize  $\phi = \{\mu, \log \sigma\}$ , maximizing the entropy  $\mathcal{H} \propto \sum \log \sigma$ .

**Gradient Estimation:** Since the success metric  $\sigma(\tau)$  is non-differentiable with respect to the environment parameters, we cannot use the reparameterization trick directly. Instead, we employ the REINFORCE estimator to compute the gradients for the distribution parameters  $\phi$ :

$$\nabla_\phi \mathcal{L} \approx -\nabla_\phi \mathcal{H} - \lambda \frac{1}{N} \sum_{i=1}^N (\hat{\sigma}_i - b) \nabla_\phi \log \nu_\phi(\xi_i) \quad (5)$$

where  $b$  is a baseline (the mean batch success rate) used to reduce variance.

**Dual Ascent:** The Lagrangian multiplier  $\lambda$  acts as the "price" of safety. We update it via dual ascent using the error between the target  $\alpha$  and the current success rate  $\hat{\mathcal{G}}$ :

$$\lambda_{t+1} \leftarrow \max(0, \lambda_t + \eta_\lambda (\alpha - \hat{\mathcal{G}})) \quad (6)$$

This auto-tunes the penalty: if the agent performs well ( $\hat{\mathcal{G}} > \alpha$ ),  $\lambda$  decreases, allowing the entropy term to dominate and expanding the distribution. If performance decreases,  $\lambda$  increases, forcing the distribution to contract.

**Algorithm 1** DORAEMON with Success-Based Warmup

**Require:** Initial  $\phi_0$ ,  $\theta_0$ , target  $\alpha$ ,  $\alpha_{\text{warmup}}$ , buffer size  $K$

- 1:  $W \leftarrow \text{True}$
- 2: **for**  $i = 0$  **to**  $M$  **do**
- 3:   Sample  $K$  dynamics  $\xi \sim \nu_{\phi_i}$
- 4:   Collect  $K$  trajectories  $\tau$  using  $\pi_{\theta_i}$
- 5:   Update policy parameters  $\theta_{i+1}$  via RL
- 6:    $\hat{\mathcal{G}} \leftarrow \text{CalculateSuccessRate}(\tau)$
- 7:   **if**  $W$  is True **then**
- 8:     **if**  $\hat{\mathcal{G}} < \alpha_{\text{warmup}}$  **then**
- 9:        $\phi_{i+1} \leftarrow \phi_i$  {Stay static}
- 10:      **continue**
- 11:     **else**
- 12:        $W \leftarrow \text{False}$  {Activate DORAEMON}
- 13:      **end if**
- 14:     **end if**
- 15:     Update  $\phi_{i+1}$  to maximize  $\mathcal{H}(\nu_{\phi}) + \lambda(\hat{\mathcal{G}} - \alpha)$
- 16:     Update Lagrange multiplier  $\lambda$
- 17: **end for**

## IV. EXPERIMENTS

### A. Environments

We evaluate the proposed methods on two continuous control tasks from the MuJoCo suite, integrated via the Gymnasium library:

- **Hopper:** a two-dimensional one-legged figure consisting of **four main body parts**; the torso at the top, the thigh in the middle, the leg at the bottom, and a single foot on which the entire body rests. The goal is to maintain balance while hopping forward, with the episode terminating if it falls.
- **HalfCheetah:** a two-dimensional robot consisting of **nine body parts** and 8 joints connecting them. The goal is to apply torque to the joints to make the cheetah run forward as fast as possible. Unlike the Hopper, the HalfCheetah is constrained to a vertical plane and does not terminate upon falling, making it a benchmark for high-speed coordination and gait robustness.

### B. Hopper Experiment 1: Emergent Robustness to Unmodeled Dynamics

In our initial analysis, we evaluated DORAEMON’s ability to induce robust behaviors against unmodeled dynamics. We trained a Soft Actor-Critic (SAC) agent on the *Hopper* environment for 5 million timesteps, randomizing the masses of the thigh, leg, and foot via a Gaussian distribution. The specific hyperparameters used for this training run are listed in Table I.

Crucially, the *Torso Mass* was held fixed at its nominal value during training to serve as a proxy for unseen physical variations. The resulting policy was then evaluated zero-shot across a range of torso masses. As illustrated in Fig. 1, the agent maintained near-optimal performance (Reward > 1800) within a shift range of  $[-1.5, +1.0]$  kg relative to the nominal

TABLE I  
HYPERPARAMETERS FOR HOPPER DORAEMON TRAINING.

Parameter	Value
Algorithm	SAC
Total Timesteps	5,000,000
Success Threshold ( $\mathcal{R}_{thr}$ )	1,200
Target Success Rate ( $\alpha$ )	0.8
Learning Rate (SAC)	$1 \times 10^{-3}$
Param. Learning Rate ( $\alpha_{\mu, \sigma}$ )	0.01
Multiplier Learning Rate ( $\alpha_\lambda$ )	0.5
Initial $\lambda$	1.0
Minimum Std ( $\sigma_{min}$ )	0.01

mass, confirming that the robust manifold learned from the lower-body dynamics successfully generalized to the unmodeled upper-body mass variations without explicit training.

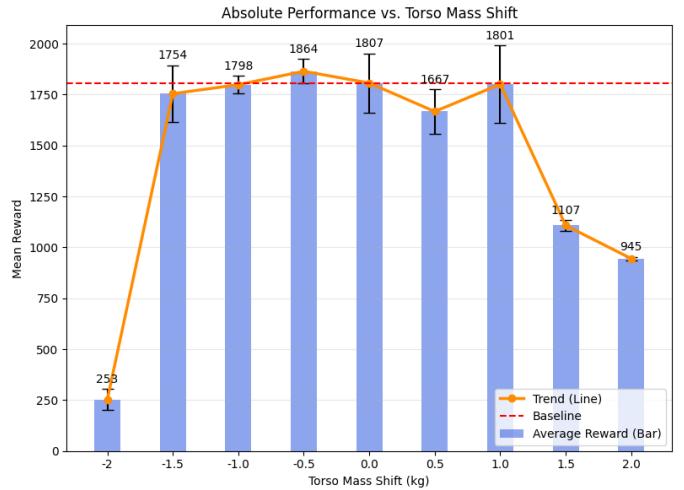


Fig. 1. Zero-Shot Robustness Analysis on Torso Mass. The agent maintains  $> 90\%$  performance within the  $[-1.5, +1.0]$  kg range, validating the robust manifold learned by DORAEMON.

These results validate that DORAEMON prevents overfitting to the nominal physics configuration without requiring exhaustive randomization of every possible simulation parameter.

### C. Hopper Experiment 2: Automatic Curriculum vs. Fixed Baselines

In the second experiment, we increased the task complexity by randomizing seven dynamics parameters simultaneously: the masses of the thigh, leg, and foot, along with surface friction and joint damping of the thigh, leg and foot. The specific hyperparameters used for DORAEMON are listed in Table II. This time we opted for a Beta distribution, to allow it to shift from a peaked Gaussian-like shape to a uniform distribution over the bounded support.

1) *Training Dynamics and Stability Analysis:* The training progression, illustrated in Fig. 2, provides a clear visualization of the entropy-maximization process and its stability limits.

- **Entropy Maximization (Bottom Plots):** The algorithm successfully drove the distribution parameters (Alpha/Beta) toward 1.0, effectively transforming the initial

TABLE II  
HYPERPARAMETERS FOR HOPPER DORAEMON TRAINING.

Parameter	Value
Algorithm	SAC
Total Timesteps	20,000,000
Success Threshold ( $\mathcal{R}_{thr}$ )	1,200
Target Success Rate ( $\alpha$ )	0.65
Learning Rate (SAC)	$1 \times 10^{-3}$
Param. Learning Rate ( $\alpha_{\mu, \sigma}$ )	0.01
Multiplier Learning Rate ( $\alpha_\lambda$ )	0.01
Initial $\lambda$	1.0
Initial $\alpha$	7.0
Min $\alpha$	1.0
Initial $\beta$	7.0
Min $\beta$	1.0

peaked distribution into a uniform distribution over the search space. This confirms DORAEMON’s ability to maximize environment diversity autonomously.

- **The Stability-Plasticity Dilemma:** As hypothesized in the original DORAEMON authors’ conclusion, we observed an instance of catastrophic forgetting. Around update step 400, the cumulative variance of the seven parameters exceeded the policy’s stability margin. The success rate (Top Plot) dropped sharply below the target  $\alpha = 0.65$ , falling to  $\approx 0.4$ .
- **Lagrangian Response (Second Plot):** The dual variable  $\lambda$  correctly identified this violation, increasing exponentially to penalize the entropy term. However, the policy had already destabilized under the high-noise regime.

This behavior highlights the critical importance of the monitoring callbacks. Rather than using the final converged model (which was struggling under excessive entropy), we utilized the history tracking to restore the policy from the optimal checkpoint (approx. Update 350) where the agent satisfied the constraints with maximal entropy, effectively filtering out the catastrophic collapse.

2) *Performance Comparison:* To evaluate the efficacy of the automatic curriculum, we compared DORAEMON against three static Uniform Domain Randomization (UDR) baselines:

- 1) *Narrow UDR*: ( $\xi \sim \mathcal{U}(0.9\xi_{\text{nom}}, 1.1\xi_{\text{nom}})$ ): Randomization range is too small to induce robustness.
- 2) *Wide UDR*: ( $\xi \sim \mathcal{U}(0.1\xi_{\text{nom}}, 10\xi_{\text{nom}})$ ): Randomization range is excessively large, including physically unsolvable configurations.
- 3) *Tuned UDR*: ( $\xi \sim \mathcal{U}(0.5\xi_{\text{nom}}, 2\xi_{\text{nom}})$ ): A manually optimized range that serves as an “oracle” baseline.

To quantify the generalization capabilities of the learned policies, we evaluated the agents across a 2D grid of environment variations, specifically shifting the *Torso Mass* ( $-2.0\text{kg}$  to  $+2.0\text{kg}$ ) and *Surface Friction* ( $-0.9$  to  $+2.1$ ). Fig. 3 illustrates the zero-shot transfer performance.

**The Failure of Naive Randomization:** The *Wide UDR* agent (bottom-right) exhibits catastrophic failure across the entire test range (Mean Reward  $< 500$ ). By sampling uniformly from an excessively wide distribution, the agent was forced to train on physically infeasible configurations, preventing

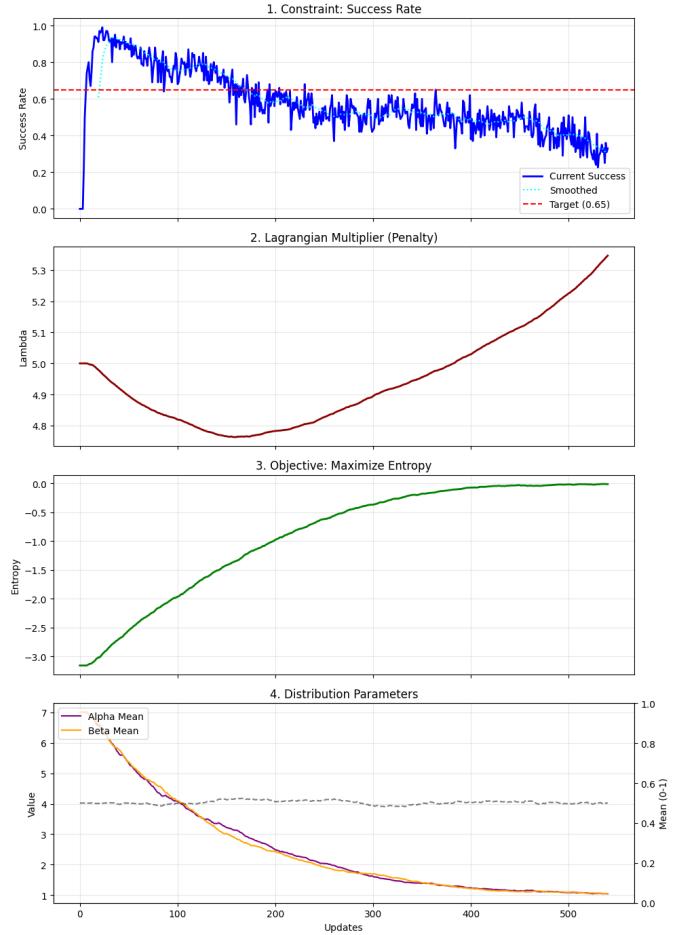


Fig. 2. Training Dynamics for Experiment 2 (7-Parameter Hopper). 1) Success rate holds until around update 400, then collapses under stress. 2) Lambda (penalty) rises to counteract the drop. 3) Entropy rises steadily. 4) Distribution parameters ( $\alpha, \beta$ ) converge toward 1 (Uniform), maximizing diversity.

convergence. This confirms that simply maximizing variance without constraints is detrimental.

**Autonomous Curriculum vs. Manual Tuning:** DORAEMON (top-left) autonomously discovered a feasibility manifold nearly identical to the manually *Tuned UDR* (bottom-left). Both agents identified the “safe” operational region (Green area) and maintained high performance (Reward  $> 1600$ ) even under significant domain shifts.

**Superior Generalization:** Crucially, DORAEMON outperforms the *Narrow UDR* baseline (top-right) in marginal regions. At high friction shifts ( $+1.5$ ), *Narrow UDR*’s performance degrades to  $\approx 1100$ , whereas DORAEMON retains robust performance ( $> 1200$ ), proving that the entropy-driven curriculum successfully expanded the agent’s robust zone beyond the nominal dynamics.

#### D. Half-Cheetah Experiments

In addition, we evaluate DORAEMON’s effectiveness on the Half-Cheetah environment, implementing a 7-dimensional randomization space. The randomization protocol independently varies the physical properties of the robot’s components

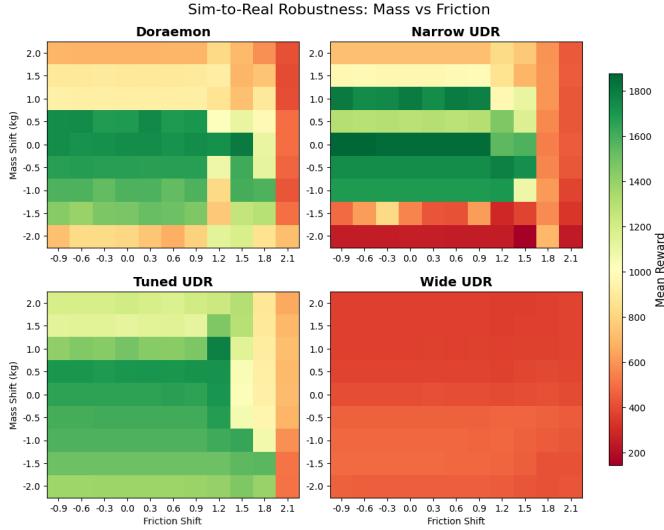


Fig. 3. Zero-Shot Robustness Heatmaps (Mass vs. Friction). **Wide UDR** fails completely due to training on infeasible physics. **DORAEMON** autonomously recovers the robust performance profile of the manually **Tuned UDR**, significantly outperforming the **Narrow** baseline in high-friction regions (right side of the plots).

to break natural symmetries and simulate hardware discrepancies:

- **Link Masses (6D)**: We independently randomize the mass of each limb (thighs, shins, and feet for both front and back legs, no torso). This asynchrony forces the policy to compensate for unbalanced weight distributions and limb-specific dynamics.
- **Floor Friction (1D)**: The sliding friction coefficient between the feet and the ground is varied to simulate different contact surfaces.

1) *Training Configuration and Hyperparameters*: The experiments were conducted using the Soft Actor-Critic (SAC) algorithm and the agent was trained using the DORAEMON framework for a total of 5 million timesteps. The specific hyperparameters used for this training run are listed in Table III. At the end of the training process, DORAEMON successfully expanded the sampling distribution to a standard deviation of  $\sigma = 0.4$  across all seven randomized dimensions. This indicates that the agent reached a level of competence high enough to satisfy the performance constraint ( $\mathcal{R} \geq 6000$ ) even under significant dynamic perturbations.

TABLE III  
HYPERPARAMETERS FOR HALFCHEETAH DORAEMON TRAINING.

Parameter	Value
Algorithm	SAC
Total Timesteps	5,000,000
Success Threshold ( $\mathcal{R}_{thr}$ )	6,000
WarmUp Success Rate	0.7
Learning Rate (SAC)	$3 \times 10^{-3}$
Parameter Learning Rate ( $\alpha_{\mu, \sigma}$ )	0.01
Multiplier Learning Rate ( $\alpha_{\lambda}$ )	0.05
Initial $\lambda$	1.0
Minimum Std ( $\sigma_{min}$ )	0.05

To provide a comparative baseline, we also trained the agent using a **Uniform Domain Randomization (UDR)** strategy. Under this configuration, environmental parameters were sampled from a fixed, predefined range at the beginning of each episode. Specifically, the scaling factors for both **link masses** and **floor friction** were sampled from a uniform distribution  $U(0.2, 2.0)$ . This range ensures that the agent is exposed to physical variations ranging from 20% to 200% of the nominal values.

2) *Results*: The results demonstrate that DORAEMON is remarkably robust to torso mass shifts, maintaining high performance even when the parameter deviates significantly from the training default. In the most challenging scenario, where the torso mass was reduced by 6 kg, resulting in an actual mass of only 0.2502 kg, the agent still achieved a reward of 3045, which corresponds to 78.5% of its baseline performance.

While UDR also proves to be robust, maintaining a high percentage of its baseline performance across extreme shifts, its absolute reward remains consistently and significantly lower than DORAEMON's in every tested configuration.

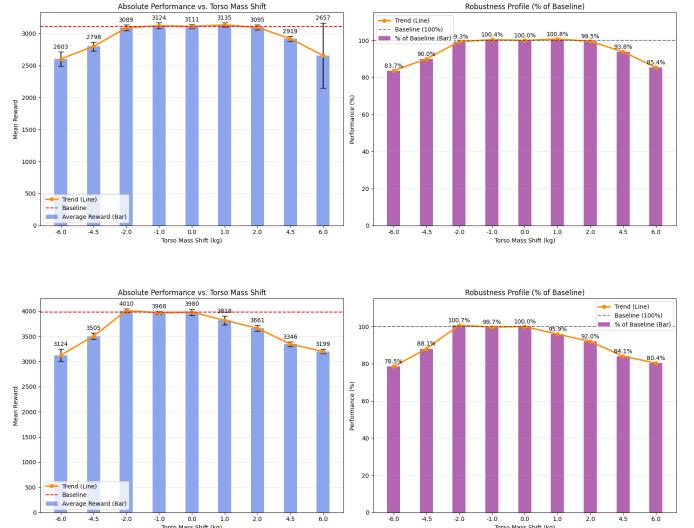


Fig. 4. Zero-Shot Robustness Analysis on Torso Mass. Comparison between UDR (top) and DORAEMON (bottom). While both exhibit robustness, DORAEMON achieves a higher performance ceiling ( $\approx 4000$  reward) and more stable generalization compared to the UDR baseline ( $\approx 3100$  reward).

The evaluation of friction shift (Figure 5) highlights a critical trade-off between peak performance and extreme out-of-distribution robustness.

DORAEMON demonstrates superior mastery in the proximity of nominal friction levels (shifts between -0.40 and +0.40). In this range, it achieves a peak reward of approximately 4000, significantly outperforming the UDR baseline which plateaus around 3167. As friction increases toward high-traction regimes (e.g., +1.00 shift), DORAEMON's performance begins to decline more sharply. At a +1.00 friction shift, DORAEMON retains 72.2% of its baseline, while UDR maintains a higher relative robustness of 81.9%.

UDR’s static and broad randomization forces the agent to learn a more “conservative” and slower gait from the beginning. While this results in lower absolute rewards, it provides the agent with a broader safety margin when faced with extreme friction levels that were never encountered during training.

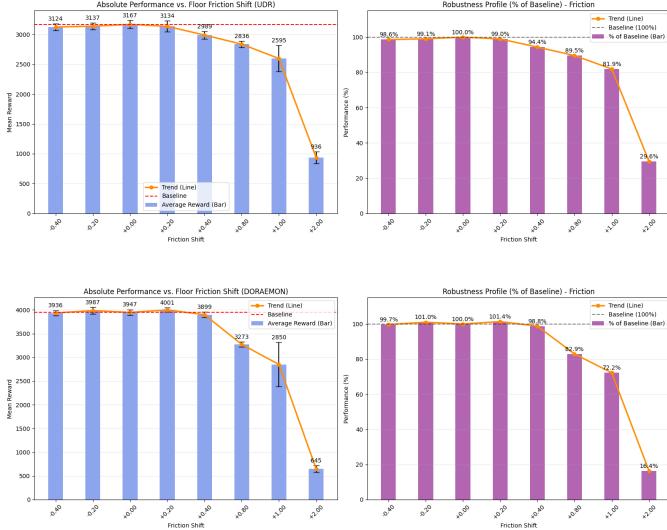


Fig. 5. Robustness Analysis on Friction shift. Comparison between UDR (top) and DORAEMON (bottom).

This trend is further illustrated by the cross-evaluation heatmaps (Figure 6), which investigate the coupling effects between mass and friction variations. The DORAEMON heatmap is characterized by intense, saturated green regions, indicating a high reward plateau (near 4000) centered around the training domain. This map reveals a wide area of high proficiency with sharp color transitions, reflecting a policy that has successfully mastered a specific parameter distribution through adaptive entropy maximization.

As we move toward the top-right corner (high mass shift combined with high friction), we observe a relatively sharp transition to warmer colors (lower rewards). This confirms that DORAEMON prioritizes peak efficiency, which can lead to steeper performance drops when multiple environmental variables simultaneously reach extreme out-of-distribution values. In contrast, the UDR heatmap presents a more diffuse color range. The baseline rewards are consistently lower (plateauing around 3100), resulting in a lighter shade of green across the center. Because UDR randomizes parameters uniformly without preferring specific configurations. Notably, the transition toward the extreme limits (the edges of the map) is more gradual. In “limit” configurations where DORAEMON’s performance begins to collapse, UDR maintains a higher relative reward.

## V. CONCLUSION

In this work, we investigated the efficacy of DORAEMON, an active domain randomization framework, in bridging the reality gap for complex robotic locomotion tasks. Our

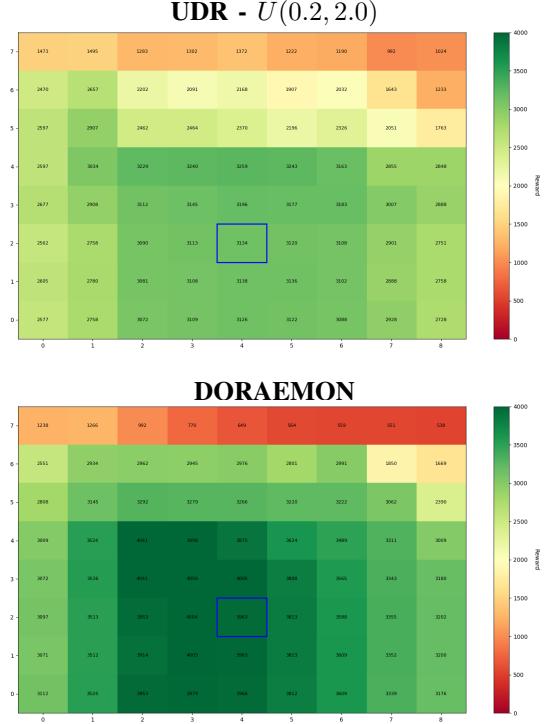


Fig. 6. Cross-evaluation heatmaps for joint Torso Mass (x-axis) and Floor Friction (y-axis) shifts. The top plot displays the **UDR baseline**, showing a diffuse, lower-reward profile that maintains marginal stability at extreme boundaries. The bottom plot represents **DORAEMON**, highlighting a saturated high-performance plateau (approx. 4000 reward) within the learned manifold, followed by a sharper performance decay at extreme out-of-distribution couplings.

experiments on the Hopper and HalfCheetah environments demonstrate that DORAEMON can outperform the traditional Uniform Domain Randomization (UDR) baseline in terms of absolute performance and policy efficiency. The core strength of DORAEMON lies in its constrained entropy maximization strategy. Unlike UDR, which forces the agent to handle a static, often overwhelming range of perturbations from the beginning, DORAEMON promotes a “prudent exploration” of the parameter space. By adaptively expanding the randomization range only when the agent satisfies a minimum performance threshold, the framework ensures that the policy maintains high efficiency while gradually increasing its robustness. This allows the agent to master the task around the most probable environmental configurations before cautiously venturing into more extreme regimes. Our results highlight that this adaptive curriculum is particularly effective in high-dimensional randomization spaces. DORAEMON successfully expanded its sampling distribution across all dimensions without suffering from the performance collapse often seen in wide-range UDR.

## REFERENCES

- [1] X. Chen, “Understanding domain randomization for sim-to-real transfer,” 2022. [Online]. Available: <https://arxiv.org/abs/2110.03239>

- [2] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas *et al.*, “Solving rubik’s cube with a robot hand,” *arXiv preprint arXiv:1910.07113*, 2019.
- [3] G. Tiboni, P. Klink, J. Peters, T. Tommasi, C. D’Eramo, and G. Chalvatzaki, “Domain randomization via entropy maximization,” in *International Conference on Learning Representations (ICLR)*, 2024. [Online]. Available: <https://arxiv.org/abs/2311.01885>
- [4] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum learning,” in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.
- [5] P. Klink, C. D’Eramo, J. R. Peters, and J. Pajarinen, “Self-paced deep reinforcement learning,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 9216–9227, 2020.