Repetition as Reinforcement: Enhancing Sample Efficiency via Instant Episode Repetition in Reinforcement Learning

Hoda Yamani, Yuning Xing, Koen van Rijnsoever, Bruce A. MacDonald, Henry Williams

Keywords: Reinforcement Learning, Experience Replay, Episode Repetition, Sample Efficiency, Continuous Control, Self-Imitation Learning, TD3, SAC

Summary

Repetition is a central mechanism in biological learning, where revisiting successful experiences strengthens memory and stabilizes skill acquisition. Inspired by this principle, we propose **Instant Episode Repetition (IER)**, a simple mechanism that improves sample efficiency in reinforcement learning by actively re-executing action sequences from high-reward episodes during environment interaction. Unlike replay-buffer methods, which passively reuse stored transitions during policy updates, IER directly modifies the data collection process: when an episode achieves a higher cumulative reward than previously observed, the agent immediately re-executes the same action sequence under different initial conditions for a fixed number of repetitions. IER integrates seamlessly with off-policy continuous-control algorithms, including SAC and TD3, without architectural modifications. Experiments on MuJoCo, the DeepMind Control Suite, and a real-world robotic manipulation task show that IER improves learning efficiency and accelerates policy refinement compared with standard off-policy and SIL-enhanced baselines, suggesting that structured action-sequence repetition provides a simple yet effective mechanism for enhancing reinforcement learning.

Contribution(s)

1. We introduce **IER**, a biologically inspired mechanism that modifies the RL interaction loop by immediately re-executing the action sequence from a prior high-reward episode.
   Context: Unlike replay-buffer methods that reuse experience only after storage, IER intervenes during data collection by replacing policy-selected actions with actions from a successful prior episode.
2. IER introduces episode-level behavioural consolidation by repeating complete action sequences from high-reward episodes during environment interaction, rather than treating experience as independent transition samples.
   Context: This shifts experience reuse from passive replay-buffer sampling toward active episode-level action replay, enabling successful behaviours to be revisited while new transitions are still being collected.
3. We provide a simple and general integration of IER into off-policy continuous-control algorithms, demonstrating compatibility with SAC and TD3 without architectural modifications.
   Context: The method operates at the interaction level and does not require changes to network structures or loss functions.
4. Through evaluation on MuJoCo, the DeepMind Control Suite, and a real-world robotic manipulation task, we demonstrate that immediate episode repetition improves sample efficiency and accelerates convergence under consistent training settings.
   Context: The empirical analysis compares IER against standard off-policy baselines and self-imitation methods under consistent training settings.

Repetition as Reinforcement: Enhancing Sample Efficiency via Instant Episode Repetition in Reinforcement Learning

Hoda Yamani[1], **Yuning Xing**[1], **Koen van Rijnsoever**[1], **Bruce A. MacDonald**[1], **Henry Williams**[1]

hyam650@aucklanduni.ac.nz, yxin683@aucklanduni.ac.nz,
kvan910@aucklanduni.ac.nz, b.macdonald@auckland.ac.nz,
henry.williams@auckland.ac.nz

[1]**Robot Learning Team, Centre for Automation and Robotic Engineering Science (CARES), Department of Electrical, Computer and Software Engineering, University of Auckland, New Zealand**

Abstract

Repetition is a fundamental mechanism in human learning, where revisiting successful experiences strengthens memory, consolidates skills, and improves future performance. Motivated by this biological principle, we introduce Instant Episode Repetition (IER), a simple and novel mechanism that improves sample efficiency by immediately repeating action sequences from successful episodes during environment interaction. Unlike conventional approaches such as Experience Replay and Self-Imitation Learning (SIL), which passively reuse past experience during training updates, IER directly influences the data collection process. Upon identifying a high-reward episode, the agent repeats its action sequence for a fixed number of subsequent episodes, reinforcing valuable behaviours through renewed interaction with the environment. We integrate IER into state-of-the-art SAC and TD3 algorithms and evaluate its effectiveness on continuous-control benchmarks, including MuJoCo, the DeepMind Control Suite, and a real-world dynamic object translation task with a robotic manipulator. Experimental results demonstrate that this simple mechanism improves learning performance over standard and self-imitation-based baselines.

1 Introduction

Reinforcement Learning (RL) offers a powerful framework for training agents to make sequential decisions by interacting with their environment and learning from feedback Sutton et al. (1998). RL has achieved notable success in various domains, from robotic manipulation and autonomous driving to game playing Han et al. (2023); Silver et al. (2016); Kiran et al. (2021). However, one of the key challenges that continues to limit its broader applicability is sample efficiency, the ability to learn effective policies from a limited number of interactions Ding & Dong (2020). In contrast to biological agents, which can often learn complex behaviors from relatively few experiences, RL agents typically require millions of interactions to achieve comparable performance Lake et al. (2017); Bellemare et al. (2016). This inefficiency presents a significant barrier, especially in real-world environments where interactions are expensive and time-consuming Dulac-Arnold et al. (2021).

Neuro-scientific research highlights reward-driven repetition as a fundamental mechanism underlying efficient learning Schultz (2006); Doya (2008). When humans or animals experience a rewarding outcome, they are more likely to reengage in the same sequence of actions that led to that reward

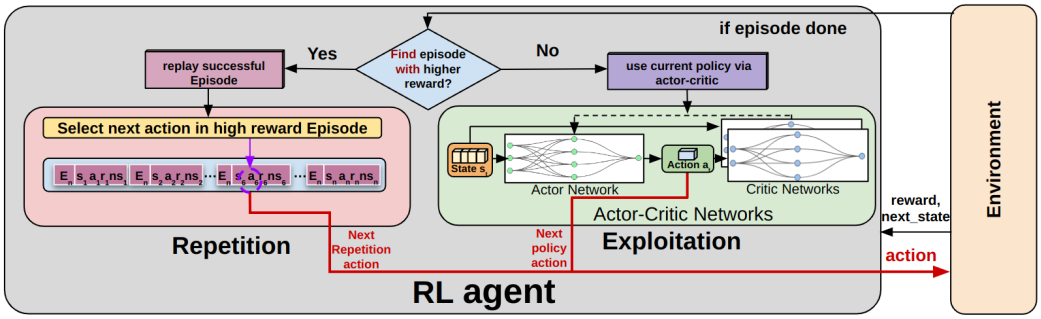


Figure 1: Illustration of the Instant Episode Repetition Algorithm: At the end of each episode (when done is true), if the agent identifies an **episode with a high reward**, it immediately initiates repetition by executing the actions from that episode sequentially, from start to end, for RN repetitions within the environment.

Kahana et al. (2024). This action repetition reinforces the link between actions and outcomes and supports the development of procedural memory and motor skills Hartley (2008); Graybiel (2008). For example, when learning to ride a bike, a person may initially struggle, trying different ways to balance, steer, and pedal. However, once they successfully ride a short distance, they instinctively repeat the same coordination of movements Ericsson et al. (1993); Kahana et al. (2024). Each repetition improves their competence, and the successful action pattern becomes more stable and efficient. At the neural level, this process strengthens synaptic connections through mechanisms such as synaptic plasticity and dopaminergic modulation, which support the consolidation and refinement of effective strategies over time Hebb (2005); Schultz et al. (1997).

Inspired by this, we investigate how repetition can be explicitly incorporated into RL frameworks to reinforce high-value experiences and improve learning. While existing techniques such as experience replay Lin (1992) and PER Schaul et al. (2015) enhance training efficiency by reusing stored transitions, they do so passively during policy updates and do not directly influence how the agent behaves during interaction. Consequently, even successful behaviors are rarely repeated in the environment, delaying their reinforcement.

Recent methods, such as Self-Imitation Learning (SIL) Oh et al. (2018), build on this idea by encouraging agents to imitate past experiences with high cumulative rewards obtained within an episode. SIL samples episodes from the replay buffer and prioritizes high returns, to guide learning. However, these approaches remain passive and do not directly alter the agent's interaction with the environment Dai et al. (2020).

We present **Instant Episode Repetition (IER)**, a novel and conceptually simple mechanism designed to improve sample efficiency in RL by reinforcing successful behaviors immediately upon discovery. In contrast to conventional off-policy approaches, which execute policy-generated actions during interaction and rely primarily on passive reuse of stored transitions in replay buffers, IER modifies the data collection process through repetition of action sequences from newly discovered high-reward episodes. When the agent identifies an episode achieving a new highest-reward episode, it stores the corresponding action sequence and repeats it immediately after the current episode for a fixed number of subsequent episodes, instead of sampling actions from the current policy. By generating additional samples around behaviours that have already demonstrated high value, IER increases the density of informative experience in promising regions of the state–action space. Importantly, this intervention affects only the interaction strategy and does not alter the underlying network architectures, objective functions, or optimization procedures.

We integrate IER into standard off-policy frameworks and evaluate its effectiveness across continuous-control benchmarks in both simulated and real-world robotic environments. The exper-

imental results demonstrate that IER outperforms the baseline algorithms, highlighting its potential to improve learning efficiency and performance.

2 Background and Related Work

2.1 Behavioral and Neural Foundations of Repetition

Repetition and reinforcement are fundamental mechanisms of adaptive learning in biological systems Haleem et al. (2015). Reinforcement increases the likelihood of repeating actions associated with rewarding outcomes, a principle formalized in operant conditioning and supported by dopaminergic reward prediction signaling Thorndike (2017); Schultz et al. (1997). Repetition, in turn, stabilizes and refines reinforced behaviors through repeated activation of neural circuits and structured practice, strengthening synaptic efficacy and supporting long-term memory consolidation Hebb (2005); Bliss & Lømo (1973).

At the neural level, repeated activation of task-relevant circuits strengthens synaptic connections, a process captured by Hebbian learning Hebb (2005). Long-term potentiation provides physiological evidence that sustained co-activation enhances synaptic efficacy and supports memory consolidation Bliss & Lømo (1973). Through these mechanisms, repetition transforms transient successes into stable behavioral patterns and progressively increases execution efficiency Graybiel (2008).

Importantly, repetition is not mechanical duplication. Behavioral and motor learning research emphasizes "repetition without repetition," where each execution occurs under slightly varying conditions, promoting robustness and generalization Bernstein (1967); Latash (2012). Neuroimaging studies further demonstrate repetition suppression, reflecting increased processing efficiency as familiar patterns are re-engaged Henson (2003). Together, these findings indicate that effective learning arises from repeatedly re-engaging successful behaviors under natural variability.

These principles suggest that repetition consolidates success by stabilizing effective behaviors while preserving adaptability. Translating this insight to RL motivates the deliberate re-execution of high-reward episodes as a means of reinforcing valuable strategies during interaction. Motivated by this biological perspective, we introduce IER as an interaction-level repetition mechanism for policy learning.

2.2 Self-Imitation Learning (SIL)

SIL is a value-based RL technique that improves policy learning by leveraging past experiences with high discounted return Oh et al. (2018). It performs off-policy updates by increasing the likelihood of actions whose discounted returns exceed the current value estimate. However, SIL remains a passive method: although it updates the policy based on successful past episodes, the agent still samples actions from its current policy, and high-reward trajectories do not directly shape new experience. This separation between learning and behavior limits SIL's ability to reinforce successful strategies during interaction.

Recent extensions of SIL improve learning efficiency by integrating additional learning signals or auxiliary mechanisms. For example, Li et al. Li et al. (2023) present A-SILfD, which incorporates expert demonstrations and constrains updates using ensemble Q-functions to mitigate overestimation. However, their approach relies on expert data emphasizing key behaviors, limiting its applicability where such data is unavailable or difficult to obtain.

Andres et al. Andres et al. (2022) combine SIL with intrinsic motivation to guide exploration toward previously successful behaviors, with intrinsic signals further emphasizing the prioritization of high-reward episodes. Dai et al. Dai et al. (2020) introduce an episodic-level replay strategy combined with hindsight to improve policy learning. While useful in goal-conditioned settings, relying on Hindsight Experience Replay (HER) limits its use in unstructured or general-purpose environments. Kuang et al. Kuang et al. (2025) incorporate adversarial training to improve robustness within the

SIL framework. However, their method operates primarily through offline replay and does not influence action selection during interaction, reducing its effectiveness for real-time adaptation and behavior shaping.

In contrast, the proposed IER approach moves beyond purely policy-driven sampling by allowing the agent to re-execute entire successful trajectories, thereby embedding the recall of past successes directly into the interaction process. By reinforcing temporally coherent action sequences rather than isolated transitions, IER strengthens valuable behavioral patterns while enabling on-policy evaluation under dynamically evolving conditions. To our knowledge, this is a novel direction in self-imitation that can be integrated into a wide range of algorithms including both value-based and actor-critic methods, without requiring structural modifications or external demonstrations.

3 Methodology

In this section, we introduce IER and describe how it integrates into standard off-policy actor–critic RL. IER operates at episode boundaries and alters the interaction dynamics while preserving the underlying learning objective and optimization procedure.

3.1 Preliminaries

RL is formulated as a Markov Decision Process (MDP)

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma), \tag{1}$$

where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $P(s' \mid s, a)$ is the transition kernel, $R(s, a)$ the reward function, and $\gamma \in [0, 1)$ the discount factor.

An episode (trajectory) is defined as

$$\tau = \{(s_0, a_0, r_0), \ldots, (s_T, a_T, r_T)\}, \tag{2}$$

representing a complete interaction sequence from an initial state to termination.

In off-policy actor–critic methods such as TD3 and SAC, transitions (s_t, a_t, r_t, s_{t+1}) are stored in a replay buffer $\mathcal{M}$ and sampled to update the actor π_θ and critic Q networks through temporal-difference learning.

For episode-level selection in IER, we define the episode reward as the sum of rewards accumulated over an episode,

$$R_{\text{ep}}(\tau) = \sum_{t=0}^{T} r_t. \tag{3}$$

IER uses this $R_{\text{ep}}(\tau)$ as the criterion for determining whether an episode should be repeated.

3.2 Instant Episode Repetition

IER modifies the data collection phase of off-policy RL by temporarily reinforcing high-reward episodes. It does not alter the reward function, loss formulation, network architecture, or optimization procedure. Instead, it modifies how episodes are generated.

At the end of each episode (when the `done` flag is true), the agent computes the episode reward R_{ep} and updates the best observed value:

$$r_{\max} \leftarrow \max(r_{\max}, R_{\text{ep}}). \tag{4}$$

For the subsequent episode, the agent operates in one of two modes, as illustrated in Figure 1:

- **Repetition:** If the current episode achieves a new maximum episode reward, the full action sequence
$$\mathbf{a}^\star = (a_0^\star, \ldots, a_T^\star) \tag{5}$$
is stored. The agent then enters a repetition phase lasting RN consecutive episodes, where RN is a hyperparameter controlling repetition strength. During this phase, the stored action sequence $\mathbf{a}^\star$ is executed sequentially instead of sampling from the policy.
- **Exploitation:** If no new maximum-reward episode is detected, or once repetition ends, the agent resumes standard behavior by sampling actions from its current policy:
$$a_t \sim \pi_\theta(\cdot \mid s_t). \tag{6}$$

All transitions collected during both repetition and exploitation are inserted into the replay buffer $\mathcal{M}$ and used for standard off-policy updates. Consequently, IER modifies only the visitation distribution of newly collected episodes while preserving the underlying learning dynamics. The complete training procedure is summarized in pseudo-code in the supplementary material.

3.3 Re-execution Under New Initial Conditions

A key property of IER is that repetition does not duplicate identical episodes. Although several benchmark environments are deterministic in their transition dynamics, each episode starts from an initial state sampled from a reset distribution p_0. As a result, re-executing the same stored action sequence $\mathbf{a}^\star$ does not reproduce the original episode exactly. Instead, it generates a family of locally related episodes in the neighborhood of the previously observed high-reward behavior.

Rather than cloning past experience, IER replays successful action sequences under perturbed initial conditions. While the action sequence remains fixed, the resulting trajectories differ at the state level due to variations in the starting state and subsequent state evolution. Consequently, repetition increases sampling density in regions of the state space that have empirically yielded high rewards, without duplicating identical transitions. This concentrates visitation around informative regions while avoiding collapse to a single deterministic trajectory.

In stochastic environments or real-world robotic systems, additional variability from sensor noise, contact dynamics, and actuation uncertainty further broadens this local sampling distribution. Thus, IER preserves replay diversity while reinforcing behaviorally salient regions, supporting stable and generalizable learning.

4 Experiments

We evaluate the proposed IER method on a diverse set of high-dimensional continuous control tasks in both simulated and real-world environments. The goal is to examine how episodic repetition affects learning performance when combined with state-of-the-art RL algorithms. The following subsections introduce the baseline algorithms used for comparison and describe the simulated and real-world tasks employed in the experiments.

4.1 Baseline Algorithms and Variants

We integrate IER into two widely used off-policy RL algorithms: TD3 Fujimoto et al. (2018) and SAC Haarnoja et al. (2018), yielding the variants **IER-TD3** and **IER-SAC**. To isolate the effect of IER, we compare these agents with:

- **TD3 and SAC:** Standard implementations with no episode repetition.
- **SIL-TD3** and **SIL-SAC**: We integrate SIL Oh et al. (2018) with TD3 following the implementation in Hu et al. (2021), and extend the same formulation to SAC for a fair comparison. We deliberately use the standard SIL formulation rather than newer variants to avoid introducing additional

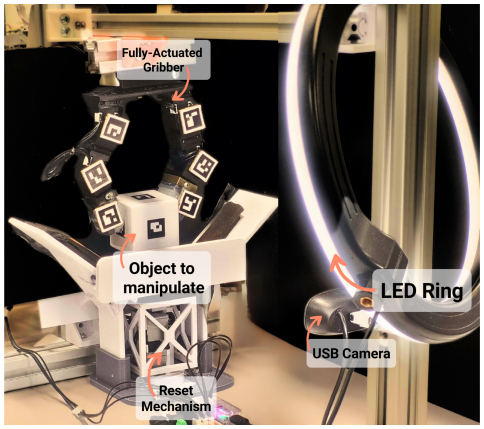


Figure 2: Real-world setup for dynamic object translation. The cube must be moved laterally while suspended in the air by a custom-built gripper.

mechanisms (e.g., demonstrations, intrinsic rewards) discussed in Background and Related Works section, which could confound the comparison. This provides a clean baseline to highlight the behavioral difference between passive experience reuse in SIL and active episodic repetition in IER.

4.2 Simulation Environments

We evaluated IER and baseline algorithms on eight continuous control tasks spanning both the **DeepMind Control Suite** Tassa et al. (2018) and **MuJoCo** Todorov et al. (2012). These simulation platforms are widely recognized benchmarks for RL, providing diverse dynamics, varying levels of task complexity, and high-dimensional state-action spaces that closely resemble real-world continuous control challenges. By selecting tasks from both environments, we ensure a comprehensive evaluation of IER's robustness and generalization across different domains and difficulty levels. Detailed descriptions and configurations of the specific tasks used in our experiments are provided in the supplementary material.

4.3 Real-World Task: Dynamic Object Translation via In-Hand Manipulation

We evaluated our method on a stochastic dexterous in-hand manipulation task that requires moving a cube sideways while maintaining a stable grasp. As shown in Figure 2, each episode starts with a reset mechanism that autonomously repositions the cube to ensure consistent trial conditions and minimize human intervention. The gripper must perform precise and coordinated movements to keep the cube from slipping or falling, relying entirely on its dexterity without external support.

To provide a reliable perception, the workspace is uniformly illuminated by a ring of LEDs, while a camera tracks the pose of the cube in real time using ArUco markers. The agent receives rewards for maintaining a stable grasp and increasing lateral displacement between time steps, encouraging fine-grained control while preserving grasp integrity. This setup captures key real-world challenges, including contact dynamics, gravitational effects, sensor noise, and constraints on data collection, while supporting scalable, physically grounded experimentation. Additional details on the hardware, reward function, and implementation are provided in the supplementary material.

5 Experimental Results

We evaluate the performance of the proposed repetition-based methods, **IER-SAC** and **IER-TD3**, across eight simulated continuous-control tasks and a real-world robotic task. These variants are compared against the standard SAC and TD3 baselines, along with their respective self-imitation

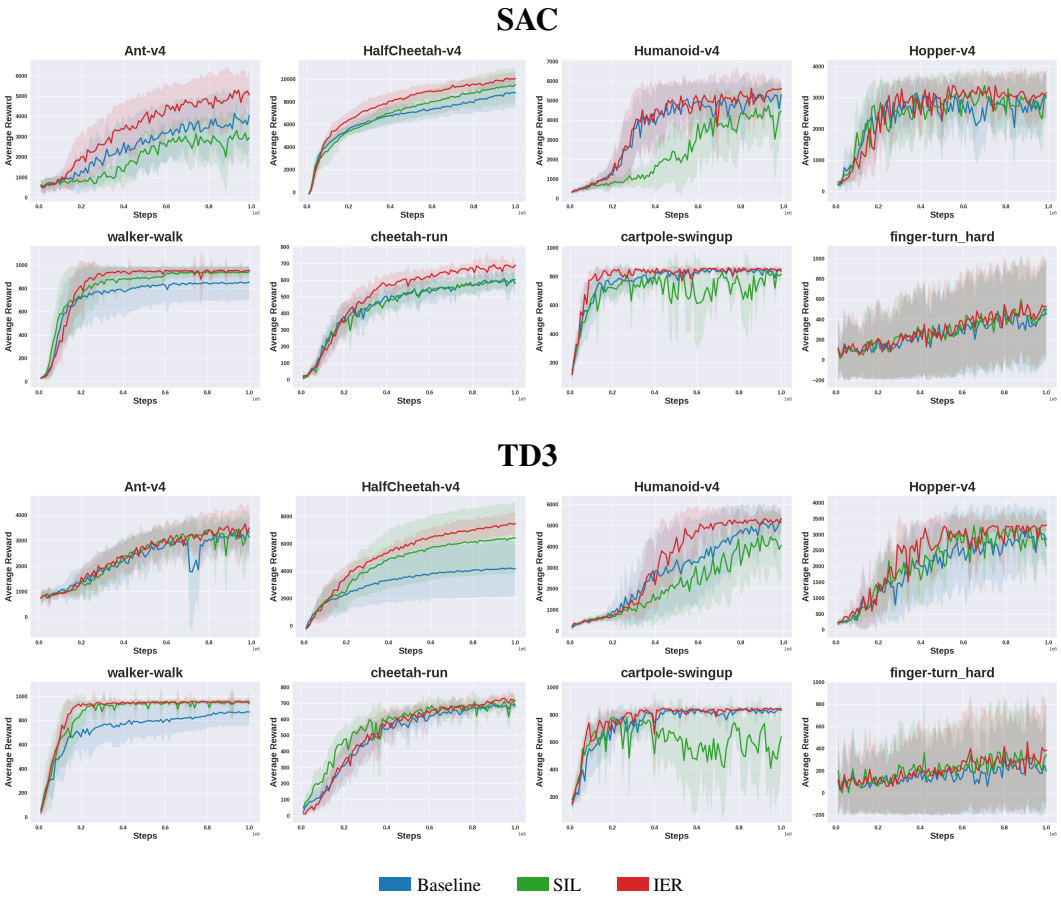


Figure 3: Performance comparison of Baseline, SIL, and the proposed IER method across eight continuous-control tasks under SAC (top) and TD3 (bottom). Curves show the mean episodic return over five independent seeds, and shaded regions represent the standard deviation across five random seeds.

learning variants (SIL-SAC and SIL-TD3). The learning curves for simulated tasks are shown in Figure 3, which reports SAC-based methods (top) and TD3-based methods (bottom). Results for the real-world setup are presented in Figure 4.

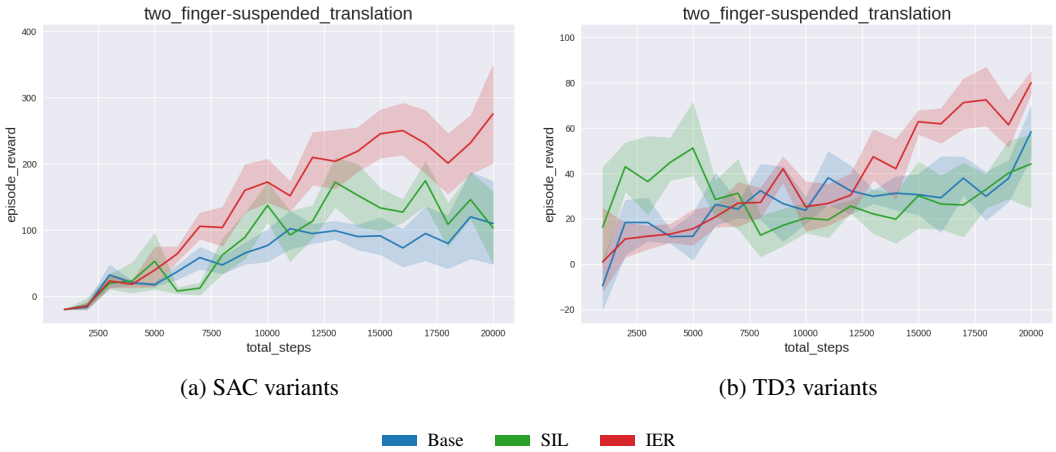
Across most tasks, **IER-SAC** and **IER-TD3** consistently outperform their respective baselines, demonstrating that repetition of high-reward episodes improves learning efficiency and convergence speed with minimal additional complexity.

In some environments, baseline performance is slightly lower than values reported in the original publications, likely due to simulator stochasticity and random seed variation. However, relative performance differences between competing methods remain consistent across seeds. Our evaluation follows established best practices for fair RL benchmarking, as outlined by Henderson et al. Henderson et al. (2018).

We begin by presenting results from simulated environments for both SAC- and TD3-based variants, followed by evaluation in a real-world setting. Finally, we conduct a sensitivity analysis to examine the impact of RN.

5.1 Performance in Simulated Environments

As shown in Figure 3, **IER-SAC** improves over the SAC baseline in six out of eight tasks and performs comparably in the remaining two. The strongest gains occur in dynamic locomotion tasks



(a) SAC variants

(b) TD3 variants

Figure 4: Performance comparison of standard, SIL-enhanced, and IER-enhanced SAC and TD3 variants in the real-world gripper manipulation task. Curves show the mean episodic return over five independent seeds, and shaded regions represent the standard deviation across five random seeds.

such as *Ant-v4*, *HalfCheetah-v4*, *Walker-Walk*, and *Cheetah-Run*, where agents must learn coordinated and temporally extended movement patterns. In these domains, repetition reinforces stable behavioral sequences, leading to faster convergence and improved long-term rewards.

SIL-SAC performs competitively in selected environments, including *HalfCheetah-v4* and *Walker-Walk*, but shows inconsistent performance in more unstable tasks such as *Humanoid-v4* and *Cartpole-Swingup*. In some cases, performance falls below the SAC baseline, reflecting a known limitation of self-imitation learning: the risk of reinforcing suboptimal trajectories when exploration remains incomplete Oh et al. (2018); Guo et al. (2019).

Under TD3 (bottom row of Figure 3), **IER-TD3** outperforms standard TD3 in six out of eight tasks and exceeds SIL-TD3 in five tasks, with notable improvements in *HalfCheetah-v4*, *Humanoid-v4*, and *Walker-Walk*. These results further confirm that repetition strengthens beneficial trajectory segments that contribute to improved cumulative reward.

While both IER variants outperform their respective baselines, the relative improvements observed with **IER-TD3** are generally smaller than those seen with **IER-SAC**. This difference may stem from the underlying exploration mechanisms. SAC incorporates entropy regularization, promoting broader exploration and maintaining policy diversity. The repetition mechanism complements this objective by reinforcing effective behaviors without substantially limiting exploration.

In contrast, TD3 employs deterministic policy updates, which may be more sensitive to repeated behavior sequences and thus more prone to premature convergence if repetition is excessive. This architectural difference likely explains the comparatively smaller but still consistent gains observed for **IER-TD3**.

5.2 Real-World Evaluation

To evaluate the robustness of the proposed IER method under real-world stochasticity, we conduct experiments on a physical robotic manipulation task (see Experiments Section). Unlike simulated benchmarks, real-world environments introduce unavoidable sources of variability, including sensor noise, friction and contact inconsistencies, and actuation delays. These factors make trajectory execution inherently stochastic, even when identical action commands are applied.

As shown in Figure 4, **IER-SAC** achieves faster convergence and higher final performance compared to both SAC and SIL-SAC. A similar pattern is observed in the TD3-based setting, where **IER-TD3** outperforms TD3 and SIL-TD3, although with narrower margins. While SIL-TD3 exhibits a

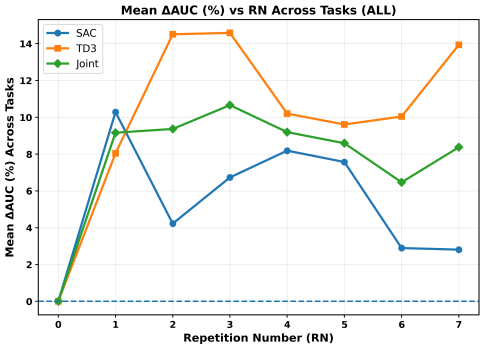


Figure 5: Mean ΔAUC% across all eight tasks as a function of the RN, jointly averaged over SAC and TD3.

temporary early improvement, its performance quickly saturates and fails to sustain gains, allowing both TD3 and IER-TD3 to surpass it after relatively few environment interactions.

These results indicate that the repetition strategy remains effective under real-world stochasticity. In particular, IER improves data efficiency and promotes more stable learning dynamics in this setting.

5.3 Sensitivity Analysis of the Repetition Number (RN)

This section examines how the RN affects IER performance. RN specifies how many consecutive episodes re-execute a high-reward episode once it is identified. We evaluate RN values from $\{0, 1, \ldots, 7\}$ both IER-SAC and IER-TD3 across all benchmark tasks. Performance is measured using the normalized Area Under the Learning Curve (AUC), capturing both learning speed and stability. We report the relative improvement over RN=0 (no repetition):

$$\Delta\text{AUC}\% = \frac{\text{AUC}_{\text{RN}} - \text{AUC}_{\text{RN0}}}{\text{AUC}_{\text{RN0}}} \times 100. \tag{7}$$

5.3.1 Aggregate Trends Across Tasks

Figure 5 shows the mean ΔAUC% across all eight tasks as a function of RN, jointly averaged over SAC and TD3. Performance peaks at RN3 (10.66%), followed by RN2 and RN4, and declines at larger repetition levels. The resulting unimodal pattern indicates that moderate repetition yields the strongest overall acceleration of learning. Small RN values may insufficiently reinforce high-reward episodes, whereas excessively large RN values can reduce state diversity, leading to diminishing returns.

We report the joint average across SAC and TD3 to provide a unified view of repetition effects across baseline algorithms. Although task-wise analyzes are presented separately, the aggregate curve highlights repetition dynamics that remain consistent across settings and provides a compact estimate of the most effective repetition range.

5.3.2 Task-Specific Trends and Algorithm Sensitivity

Figure 6 presents the per-task ΔAUC% as a function of the RN for both TD3 and SAC. Results suggest that environments with higher instability or reward variance tend to benefit more strongly from repetition. For example, **Finger-Turn-Hard** and **Walker-Walk** show substantial gains at moderate or higher RN values. In **Walker-Walk** and **Cheetah-Run**, performance often improves with increasing RN and typically peaks at intermediate levels (RN3–RN5). In contrast, tasks such as **Hopper-v4** and **Cartpole-Swingup** exhibit smaller and more variable changes, consistent with their comparatively stable reward dynamics.

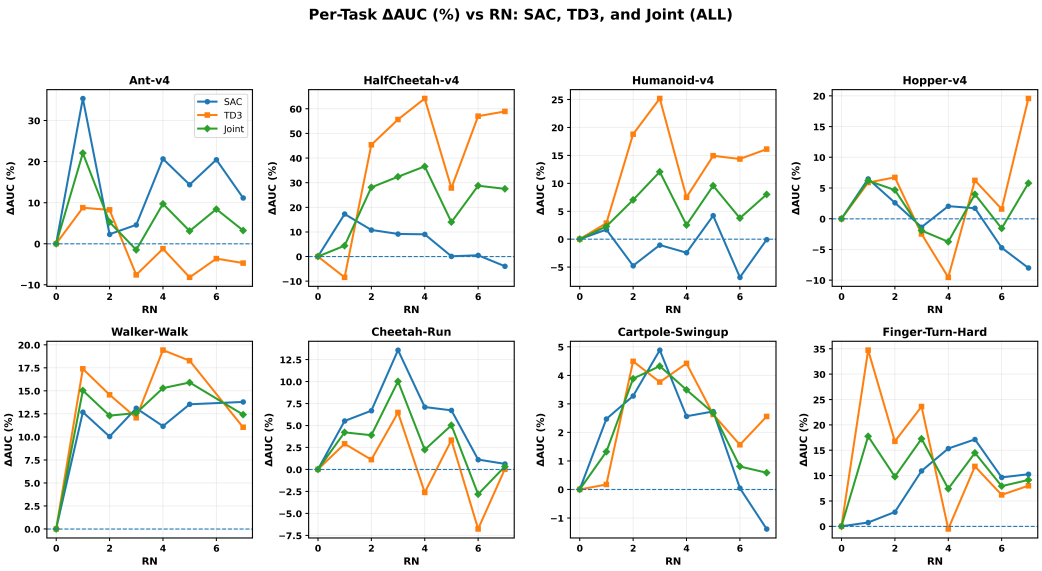


Figure 6: Per-task ΔAUC% as a function of the Repetition Number (RN), from RN0 (no repetition) to RN7, for SAC and TD3.

The impact of repetition varies across baseline algorithms. For TD3, performance commonly peaks at moderate RN values (RN2–RN3), yielding consistent improvements across multiple tasks. In contrast, SAC tends to benefit from smaller RN values, with performance exhibiting greater sensitivity at larger repetition levels. This difference reflects their respective update mechanisms: TD3 gains from stronger reinforcement of high-reward episodes, while SAC's entropy regularization inherently promotes exploration, increasing its sensitivity to excessive repetition.

Although the optimal RN varies across environments and occasional performance declines are observed at certain repetition levels, moderate repetition (RN2–RN4) often achieves competitive and near-peak performance. These results indicate that RN should be treated as a tunable hyperparameter rather than a fixed setting.

Additional analyses, including complete RN0–RN7 performance curves, domain-level comparisons, and agreement analysis between TD3 and SAC, are provided in the supplementary material. These extended results further demonstrate the robustness of moderate repetition and confirm its effectiveness across both MuJoCo and DeepMind Control environments.

6 Conclusion

This work introduced IER, a repetition-based mechanism that enhances RL by directly reinforcing action sequences from high-reward episodes through immediate re-execution. Unlike traditional approaches that rely solely on replayed transitions, IER intervenes during data collection, enabling agents to revisit successful behaviours during data collection.

IER provides a simple, general, and biologically inspired extension to off-policy RL that requires no architectural modifications. Empirical results across simulated benchmarks and real-world robotic tasks demonstrate consistent improvements in data efficiency and training stability.

References

Alain Andres, Esther Villar-Rodriguez, and Javier Del Ser. Towards improving exploration in self-imitation learning using intrinsic motivation. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 890–899. IEEE, 2022.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Nicholas Bernstein. The coordination and regulation of movements. *(No Title)*, 1967.

Tim VP Bliss and Terje Lømo. Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *The Journal of physiology*, 232(2):331–356, 1973.

Nikhil Chavan Dafle, Alberto Rodriguez, Robert Paolini, Bowei Tang, Siddhartha S Srinivasa, Michael Erdmann, Matthew T Mason, Ivan Lundberg, Harald Staab, and Thomas Fuhlbrigge. Extrinsic dexterity: In-hand manipulation with external forces. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1578–1585. IEEE, 2014.

Tianhong Dai, Hengyan Liu, and Anil Anthony Bharath. Episodic self-imitation learning with hindsight. *Electronics*, 9(10):1742, 2020.

Zihan Ding and Hao Dong. Challenges of reinforcement learning. *Deep Reinforcement Learning: Fundamentals, Research and Applications*, pp. 249–272, 2020.

Kenji Doya. Modulators of decision making. *Nature neuroscience*, 11(4):410–416, 2008.

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

K Anders Ericsson, Ralf T Krampe, and Clemens Tesch-Römer. The role of deliberate practice in the acquisition of expert performance. *Psychological review*, 100(3):363, 1993.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Ann M Graybiel. Habits, rituals, and the evaluative brain. *Annu. Rev. Neurosci.*, 31(1):359–387, 2008.

Yijie Guo, Jongwook Choi, Marcin Moczulski, Samy Bengio, Mohammad Norouzi, and Honglak Lee. Self-imitation learning via trajectory-conditioned policy for hard-exploration tasks. 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Abdul Haleem, Muhammad Khalil Khan, Shamta Sufia, Saima Chaudhry, Muhammad Irfanullah Siddiqui, and Ayyaz Ali Khan. The role of repetition and reinforcement in school-based oral health education-a cluster randomized controlled trial. *BMC Public Health*, 16:1–11, 2015.

Dong Han, Beni Mulyana, Vladimir Stankovic, and Samuel Cheng. A survey on deep reinforcement learning algorithms for robotic manipulation. *Sensors*, 23(7):3762, 2023.

James Hartley. Learning and studying: A research perspective. 2008.

Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology press, 2005.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Richard NA Henson. Neuroimaging studies of priming. *Progress in neurobiology*, 70(1):53–81, 2003.

Hao Hu, Jianing Ye, Guangxiang Zhu, Zhizhou Ren, and Chongjie Zhang. Generalizable episodic memory for deep reinforcement learning. *arXiv preprint arXiv:2103.06469*, 2021.

Michael J Kahana, Nicholas B Diamond, and Ada Aka. Laws of human memory. *The Oxford Handbook of Human Memory, Two Volume Pack: Foundations and Applications*, pp. 29–63, 2024.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE transactions on intelligent transportation systems*, 23(6):4909–4926, 2021.

Yingyi Kuang, Luis J Manso, and George Vogiatzis. Goal-based self-adaptive generative adversarial imitation learning (goal-sagail) for multi-goal robotic manipulation tasks. *arXiv preprint arXiv:2506.12676*, 2025.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017.

Mark L Latash. The bliss (not the problem) of motor abundance (not redundancy). *Experimental brain research*, 217(1):1–5, 2012.

Chao Li, Fengge Wu, and Junsuo Zhao. Accelerating self-imitation learning from demonstrations via policy constraints and q-ensemble. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2023.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8:293–321, 1992.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International conference on machine learning*, pp. 3878–3887. PMLR, 2018.

Youngmin Oh, Jinwoo Shin, Eunho Yang, and Sung Ju Hwang. Model-augmented prioritized experience replay. In *International Conference on Learning Representations*, 2021.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Wolfram Schultz. Behavioral theories and the neurophysiology of reward. *Annu. Rev. Psychol.*, 57 (1):87–115, 2006.

Wolfram Schultz, Peter Dayan, and P Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Edward Thorndike. *Animal intelligence: Experimental studies*. Routledge, 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

David Valencia, Henry Williams, Yuning Xing, Trevor Gee, Minas Liarokapis, and Bruce A. MacDonald. Image-based deep reinforcement learning with intrinsically motivated stimuli: On the execution of complex robotic tasks. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2024. URL https://arxiv.org/abs/2407.21338.

Supplementary Materials

The following content was not necessarily subject to peer review.

This supplementary material provides additional methodological details and experimental results that complement the main paper. The content is organized as follows:

- **Appendix A** – Pseudo-code of the proposed IER algorithm.
- **Appendix B** – Extended Analysis of the Repetition Number (RN)
- **Appendix C** – Description of the experimental environments, including MuJoCo tasks, the DeepMind Control Suite, and the real-world *Dynamic Object Translation with Gripper Manipulation* setup.
- **Appendix D** – Detailed hyperparameter configurations for IER and baseline algorithms.

Appendix A: IER Algorithm

In this section, we present the pseudocode of the Instant Episode Repetition (IER) algorithm, as shown in Algorithm 1.

Algorithm 1: Policy Training with Instant Episode Repetition (IER)

Initialize: Policy π_θ, environment $\mathcal{E}$, buffer $\mathcal{M}$, exploration steps T_{explore}, training steps T_{train}, repetition length R_N
$R_{\text{max}} \leftarrow 0$, $r \leftarrow 0$, $\mathbf{a}^* \leftarrow []$, $s_0 \leftarrow \mathcal{E}$.reset(), $R_e \leftarrow 0$, step $\leftarrow 0$
for $t = 1$ to T_{train} **do**
  if $t \leq T_{\text{explore}}$ **then**
    $a_t \sim \mathcal{U}(\mathcal{A})$
  else if $r > 0$ **then**
    $a_t \leftarrow \mathbf{a}^*[\text{step}]$
  else
    $a_t \leftarrow \pi_\theta(s_t) + \epsilon_t$
  end if
  Execute a_t, observe $s_{t+1}, r_t,$ done
  Store $(s_t, a_t, r_t, s_{t+1},$ done$)$ in $\mathcal{M}$
  $R_e \leftarrow R_e + r_t$, step $\leftarrow$ step $+ 1$
  Update policy every K steps
  if done or truncated **then**
    if $R_e > R_{\text{max}}$ **then**
      $R_{\text{max}} \leftarrow R_e$, $\mathbf{a}^* \leftarrow$ actions from episode
      $r \leftarrow R_N$
    else if $r > 0$ **then**
      $r \leftarrow r - 1$
    end if
    Reset: $s_0 \leftarrow \mathcal{E}$.reset(), $R_e \leftarrow 0$, step $\leftarrow 0$
  end if
end for

The algorithm begins by initializing the policy, environment, replay buffer, and relevant parameters, such as the number of exploration steps T_{explore}, total training steps T_{train}, and repetition length R_N. During the exploration phase, actions are sampled uniformly from the action space to promote state diversity. After the exploration phase, the policy's actions are selected with added exploration noise.

When a new episode achieves a total reward R_e greater than the maximum episode reward R_{max}, the sequence of actions from that episode is stored as $\mathbf{a}^*$. This sequence is then repeated for the next R_N episodes. Repetition ensures that the policy can consistently revisit and learn from the most successful trajectories discovered so far. The variable r tracks the number of remaining repeated episodes.

At each step, the agent interacts with the environment, stores transitions in the replay buffer, and updates the policy at a fixed interval. When an episode terminates, the algorithm checks whether the episode reward exceeds $R_{\max}$ and either updates the best sequence or decrements the repetition counter. This process continues until T_{train} steps are completed.

Appendix B: Extended Analysis of the Repetition Number (RN)

This appendix provides additional experimental results analyzing the impact of the RN used in the IER algorithm on agent performance.

Figures 7 and 8 present complete performance curves across all RN values and tasks, allowing for the comparison of performance trends under varying repetition frequencies.

Across nearly all tasks, RN=0 consistently results in one of the weakest performances, confirming that learning without repetition underutilizes successful trajectories. Even a small amount of repetition (e.g., RN=1) consistently improves performance, indicating that revisiting high-reward episodes enhances learning efficiency. Higher RNs, such as RN=6, continue to provide strong performance in many tasks; however, performance occasionally declines at RN=7, likely due to overexploitation or reduced diversity in the sampled experiences.

Domain-Level Analysis. Figures 10a and 10b separate MuJoCo and DeepMind Control tasks.

In MuJoCo locomotion environments, improvements are strongest at moderate repetition levels, particularly under TD3. In DeepMind Control tasks, repetition effects are smoother and less sensitive to RN magnitude. This suggests that repetition interacts with environment dynamics, with higher-dimensional locomotion tasks benefiting more strongly from moderate replay.

Agreement Between Baseline Algorithms. To evaluate robustness beyond single-task optima, we first identify, for each task and each baseline (IER-SAC and IER-TD3), the set of RN values whose normalized AUC is within 95% of that task's best AUC. We refer to these as the near-optimal RN sets.

Agreement is defined as selecting an RN that performs well for both baselines on the same task. When the near-optimal RN sets of SAC and TD3 overlap, we choose the RN from their intersection, i.e., an RN that is simultaneously near-optimal for both algorithms.

If no intersection exists, we apply a rank-based criterion. Specifically, RN values are ranked according to their normalized AUC under each baseline, and we select the RN with the best combined rank across SAC and TD3, ensuring a balanced compromise rather than favoring a single algorithm (Table 1).

Several tasks exhibit direct intersection-based agreement, while others require rank-based compromise selection. Figure 11 summarizes the frequency of selected agreement RN values across tasks. RN2, RN3, and RN5 appear most frequently, indicating that moderate repetition levels tend to generalize across algorithms.

The results suggest that moderate repetition provides the most reliable performance across tasks and baseline algorithms. Very small RN values underutilize successful trajectories, whereas excessively large RN values yield diminishing returns. Across MuJoCo and DeepMind Control environments, RN2–RN4 offers a stable and effective operating range for IER.

Appendix C: Environment Descriptions

We evaluate our approach across a diverse set of continuous control environments from MuJoCo Todorov et al. (2012) and the DeepMind Control Suite (DMC) Tassa et al. (2018), two widely adopted platforms for studying motor control in physically realistic simulations. All tasks are con-

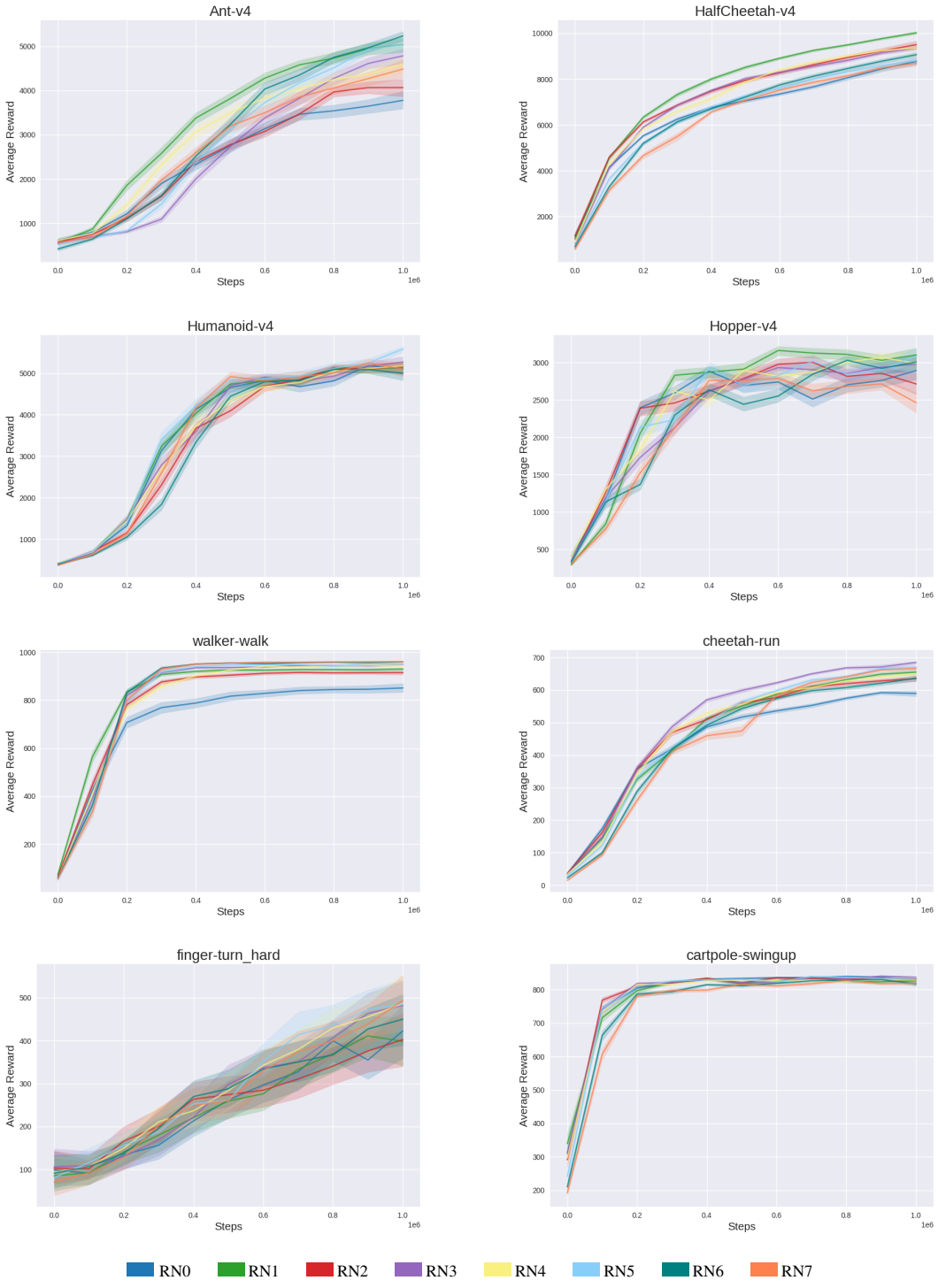


Figure 7: Impact of repetition number RN on **IER-SAC** performance across all tasks, showing how repeated execution of the highest-reward episode affects learning. Curves show the mean episodic return over ten independent seeds and are smoothed using a sliding window of size 5 for visual clarity.

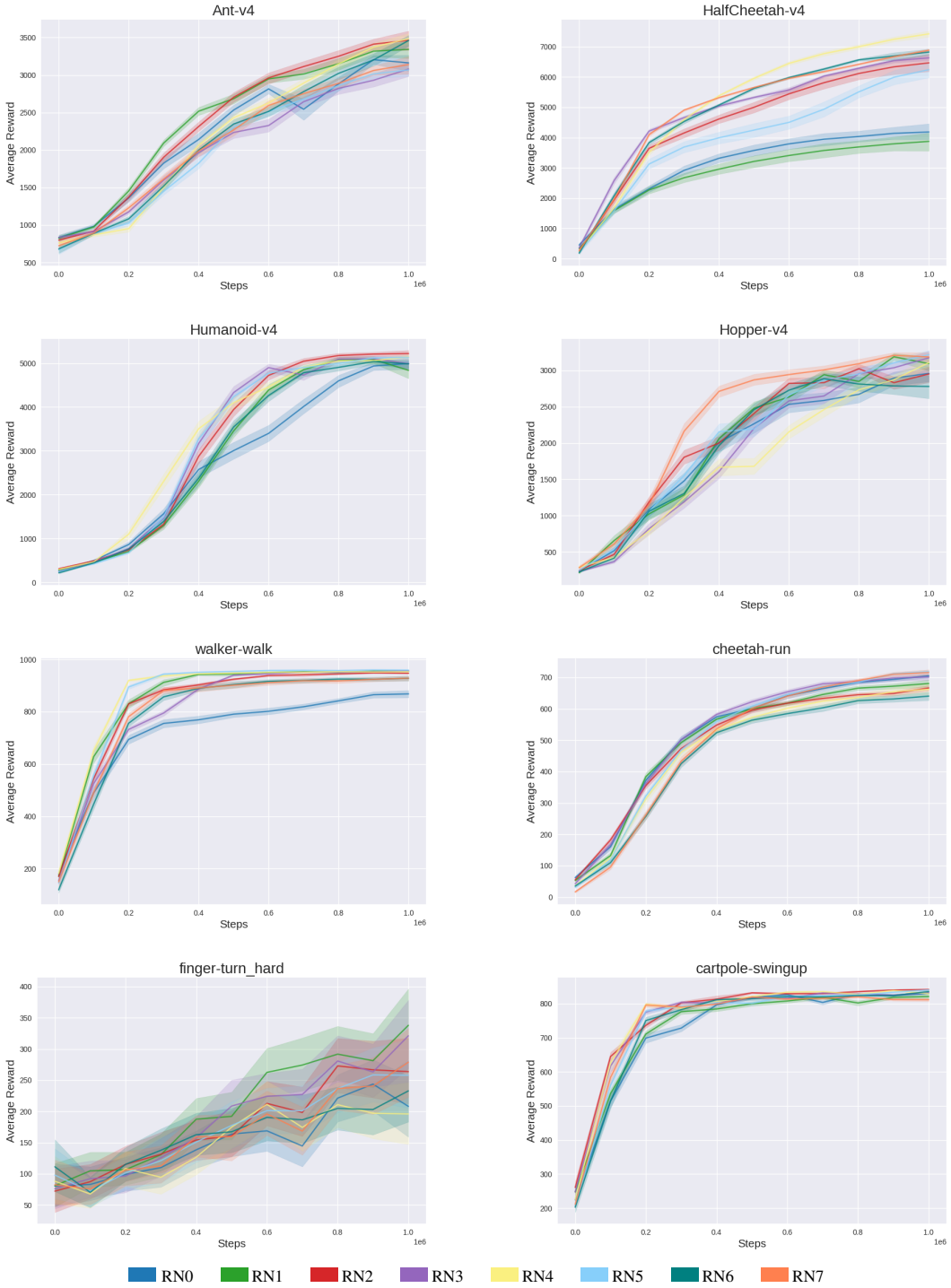
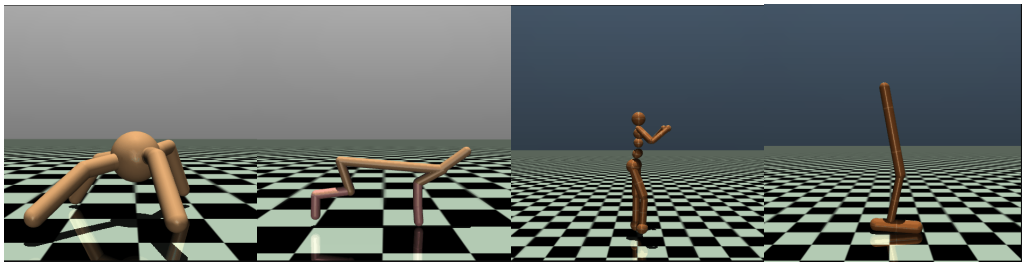
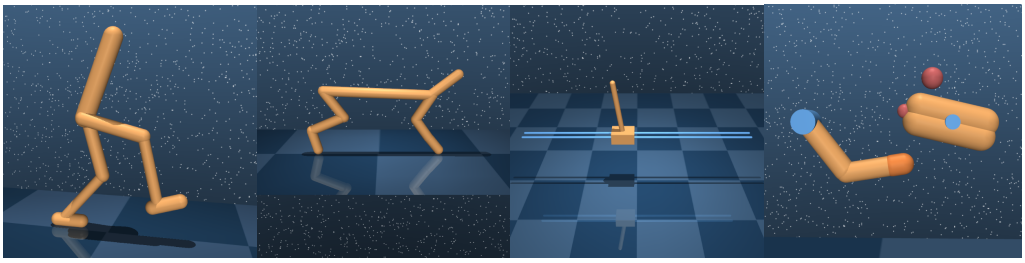


Figure 8: Impact of repetition number RN on **IER-TD3** performance across all tasks, showing how repeated execution of the highest-reward episode affects learning. Curves show the mean episodic return over ten independent seeds and are smoothed using a sliding window of size 5 for visual clarity.

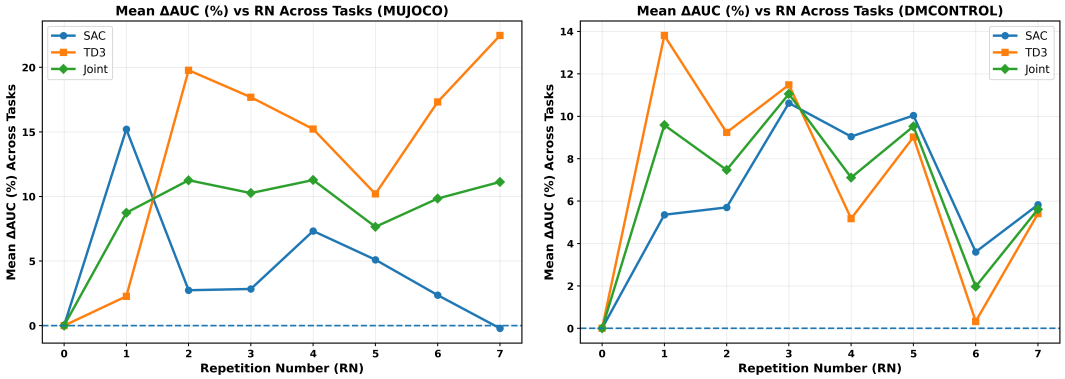


(a) Ant-v4 (b) HalfCheetah-v4 (c) Humanoid-v4 (d) Hopper-v4



(e) Walker-Walk (f) Cheetah-Run (g) Cartpole-Swingup (h) Finger-Turn-Hard

Figure 9: tasks used in our experiments.



(a) Mean ΔAUC% across MuJoCo tasks. Moderate repetition (RN3–RN4) yields the strongest gains, particularly under TD3.

(b) Mean ΔAUC% across DeepMind Control tasks. Improvements are more uniform and less sensitive to RN magnitude.

Figure 10: Mean ΔAUC% across environment groups. Left: MuJoCo tasks. Right: DeepMind Control tasks.

Table 1: Per-task RN agreement based on 95% near-best criterion.

Task	**Agreement RN**	**Selection Type**
Ant-v4	RN1	Intersection
Cheetah-Run	RN3	Intersection
Walker-Walk	RN5	Intersection
Cartpole-Swingup	RN2	Intersection
HalfCheetah-v4	RN4	Rank-based
Hopper-v4	RN2	Rank-based
Humanoid-v4	RN3	Rank-based
Finger-Turn-Hard	RN5	Rank-based

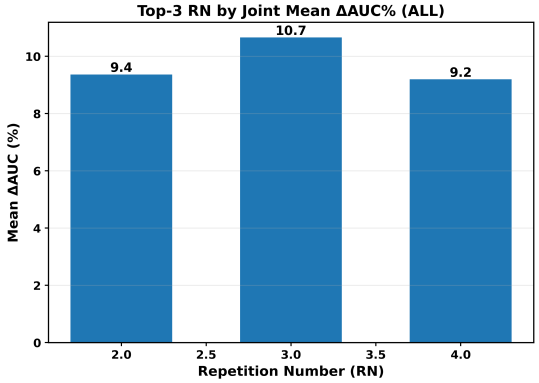


Figure 11: Frequency of RN values selected as robust joint candidates across tasks. RN2, RN3, and RN5 appear most frequently.

Table 2: Summary of environments used in evaluation.

Environment	**Suite**	**Task Type**	**Description**	**Approx. Difficulty**
Ant-v4	MuJoCo	Locomotion	Quadruped; coordination-heavy with complex contact dynamics.	High
HalfCheetah-v4	MuJoCo	Locomotion	Planar biped; dense unbounded rewards support long-term learning.	Medium
Humanoid-v4	MuJoCo	Locomotion	Biped with high DOF; unstable dynamics and sensitive to control noise.	High
Hopper-v4	MuJoCo	Locomotion	Single-legged hopping; requires balance and precise control, with early termination on failure.	Medium
Walker-Walk	DMC	Locomotion	Bipedal walking; fewer DOF than Humanoid, but still unstable.	Medium
Cheetah-Run	DMC	Locomotion	Fast planar cheetah; velocity-based control under bounded rewards.	Low
Cartpole-Swingup	DMC	Classic Control	Swing-up and balance pole on cart; sparse rewards, underactuated.	Medium
Finger-TurnHard	DMC	Manipulation	Rotate object with a single finger; fine-grained actuation required.	High

Table 3: Observation and action space dimensions and episode horizon for each environment. $\mathbb{R}^d$ denotes a d-dimensional real vector; $[-1, 1]^d$ denotes a bounded action space.

Environment	Observation Space	Action Space	Horizon
MuJoCo (vector-based)			
Ant-v4	$\mathbb{R}^{111}$	$[-1, 1]^8$	1000 (may terminate early due to instability)
HalfCheetah-v4	$\mathbb{R}^{17}$	$[-1, 1]^6$	1000
Humanoid-v4	$\mathbb{R}^{376}$	$[-1, 1]^{17}$	1000 (may terminate early due to instability)
Hopper-v4	$\mathbb{R}^{11}$	$[-1, 1]^3$	1000
DeepMind Control Suite (vector-based)			
Walker-Walk	$\mathbb{R}^{24}$	$[-1, 1]^6$	1000
Cheetah-Run	$\mathbb{R}^{17}$	$[-1, 1]^6$	1000
Cartpole-Swingup	$\mathbb{R}^5$	$[-1, 1]^1$	1000
Finger-TurnHard	$\mathbb{R}^{12}$	$[-1, 1]^2$	1000

figured with low-dimensional, vector-based observations consisting of joint positions, velocities, orientations, and other proprioceptive signals—excluding high-dimensional visual input.

MuJoCo Environments. MuJoCo tasks, accessed via the Gym interface, focus on high-speed and dynamic locomotion with varying complexity:

- **Ant-v4:** A quadrupedal agent learns to walk using a high-dimensional action space. Coordination and contact dynamics make this task nontrivial.
- **HalfCheetah-v4:** A planar biped with a flexible spine learns to run. It offers dense, unbounded rewards, with cumulative rewards often exceeding 12,000, making it suitable for evaluating long-horizon learning and experience prioritization.
- **Humanoid-v4:** A high-DOF biped must learn to walk upright. Its instability and dimensionality pose significant challenges for balance, control, and exploration.
- **Hopper-v4:** A single-legged agent must learn to hop forward while maintaining balance. The task requires precise control and stability, as failure to maintain upright posture results in early episode termination. Its sensitivity to instability makes it useful for assessing learning robustness.

DeepMind Control Suite Environments. DMC tasks are designed for general motor control using structured state vectors. Unlike MuJoCo, rewards are typically bounded in $[0, 1000]$, focusing more on precision than accumulation:

- **Walker-Walk:** A biped learns forward locomotion while maintaining balance. It involves moderately complex dynamics and fewer degrees of freedom than Humanoid.
- **Cheetah-Run:** A planar cheetah learns forward locomotion under aggressive dynamics and bounded rewards. Although structurally similar to HalfCheetah-v4, it exhibits greater control sensitivity and performance saturation. We deliberately include this task to evaluate robustness and generalization in environments with similar morphology but different reward scales and control challenges.
- **Cartpole-Swingup:** An underactuated classic control problem requiring swing-up and stabilization of a pole mounted on a cart. Sparse rewards and nonlinear dynamics make learning nontrivial.
- **Finger-TurnHard:** A single robotic finger must rotate an object to a target orientation. Unlike the easy variant, it requires more precise torque control and sustained actuation under stricter success conditions.

Table 4: Real-World Robotic Gripper Task Characteristics

Feature	Description
Task Overview	
Task Type	Dynamic cube translation with robotic gripper
Object	Cube tracked via ArUco markers
Goal	Translate cube as far as possible within camera view
Reward Components	Force-closure grasp + lateral displacement
Hardware Setup	
Gripper Dexterity	Stable force closure; no extrinsic dexterity or caging
Workspace	Camera-bounded area (blue rectangle, Fig. 12)
Materials	Consumer-grade aluminium and 3D-printed parts
Sensors	Logitech webcam with ArUco tracking
Actuation	Dynamixel servos
Control and Learning	
State Space	Servo angles; cube current/previous positions (x, y mm)
Action Space	Target servo joint angles
Episode Length	20 real-world steps
Training Steps	20,000
Reset Mechanism	Rack-and-pinion elevator for automatic repositioning
System Specifications	
PC Specs	Intel i9-10900, 128GB RAM, Nvidia RTX 3080

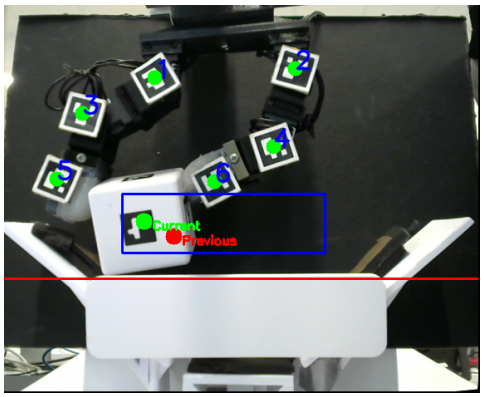


Figure 12: Workspace boundaries and cube tracking for reward evaluation. Green and red markers indicate current and previous cube positions used to compute displacement-based rewards.

Table 5: Hyperparameters used in training. Shared parameters apply to both SAC and TD3; SAC, TD3, and SIL sections list algorithm-specific settings.

Parameter	Value (Description)
	Shared Parameters
batch_size	256 (Mini-batch size)
buffer_size	10^6 (Replay buffer size)
max_steps_exploration	10^3 (Exploration steps before training)
max_steps_training	10^6 (Total training steps)
number_steps_per_evaluation	10^4 (Steps between evaluations)
number_eval_episodes	10 (Episodes per evaluation)
number_steps_per_train_policy	1 (Steps per policy update)
min_noise	0.0 (Min exploration noise)
noise_scale	0.1 (Initial Gaussian noise scale)
noise_decay	1.0 (Exploration noise decay)
actor_lr	3e-4 (Actor learning rate)
critic_lr	3e-4 (Critic learning rate)
gamma	0.99 (Discount factor)
tau	0.005 (Target smoothing)
hidden_size_actor	[256, 256] (Actor hidden layers)
hidden_size_critic	[256, 256] (Critic hidden layers)
optimizer	Adam (Training optimizer)
nonlinearity	ReLU (Network activation)
	TD3-Specific
policy_update_freq	2 (Policy update frequency)
	SAC-Specific
alpha_lr	3e-4 (Entropy temperature LR)
reward_scale	1.0 (Reward scaling)
log_std_bounds	[-20, 2] (Log-std bounds)
policy_update_freq	1 (Policy update frequency)
target_update_freq	1 (Target update frequency)
	SIL-Specific
initial_per_exponents	(0.7, 0.4) (Initial PER exponents (α, β))
replay_period	64 (Replay sampling frequency)
gradient_step	64 (Gradient updates per sampling)
max_nlogp	5 (Max negative log probability)

Real-World Robotic Task

This section describes the Dynamic Object Translation task involving gripper-based manipulation, including the hardware setup, sensing modalities, and key challenges.

The task, illustrated in Fig. 12, is a dexterous in-hand manipulation challenge requiring the gripper to securely grasp a cube and manipulate it to a desired position within image space. Unlike tasks leveraging extrinsic dexterity Dafle et al. (2014) or caging strategies, this setup demands sustained force closure to maintain grasp stability throughout the manipulation.

This task builds upon a prior two-finger push manipulation setup from Valencia et al. (2024), with a 90-degree rotation along the x-axis that orients the gripper downward. This modification eliminates the supporting base beneath the cube and intensifies gravitational effects, forcing the gripper to develop stable grasp and finger manipulation strategies to avoid dropping the object.

Success in this task is defined by continuously manipulating the cube, maximizing its displacement while maintaining a stable grasp. The task pushes beyond simulation by involving intrinsic gripper dexterity in a physical environment, introducing real-world challenges such as limited data collection and hardware constraints.

Fig. 12 shows the workspace from a camera viewpoint. The blue rectangle denotes the gripper's operational area. The cube's current and previous positions are tracked using ArUco markers, visualized as green and red circles respectively, providing intuitive feedback for monitoring and debugging. The hardware combines aluminium extrusions and 3D-printed joints, with a Logitech webcam capturing the scene. Experiments run on a single PC, ensuring real-time control. The gripper hardware is built with consumer-level components and extensive 3D printing, yielding a cost-effective setup. An automated reset mechanism is integrated to facilitate efficient data collection and consistent training. A rack-and-pinion elevator lifts the cube to a fixed start position between the gripper's fingertips at the end of each episode, enabling rapid and repeatable task restarts. A detailed summary of the task's characteristics is provided in Table 4.

Reward Structure: Rewards are computed based on two main criteria: (1) successful force-closure grasping of the object, and (2) the magnitude of lateral displacement of the cube between consecutive time steps. This reward structure encourages the agent to maintain grip stability while steadily moving the object over longer distances.

State and Action Spaces: The **state space** is a vector consisting of the positions of the servos, Aruco markers in the image (xy), cube positions (xy). Similarly, the action space is represented as a vector containing the desired positions of each servo joint. Each position value in this vector corresponds to a specific angular position for the Dynamixel servos. The vector's elements directly control the orientation and configuration of the gripper, allowing for precise manipulation of objects.

Training Setup: Agents are trained for 20,000 interaction steps. Each episode comprises 20 real-time steps, reflecting physical environment constraints.

Appendix D: Configuration and Hyperparameter Settings

This appendix provides the complete configuration and hyperparameter settings used in all experiments. All agents were implemented in PyTorch, following architectures and optimization settings consistent with prior work Fujimoto et al. (2018); Oh et al. (2021). To ensure fair comparisons, all tasks were trained under the same configuration unless specified otherwise. Experiments in the main study were averaged over five random seeds, while the ablation study on RN used ten seeds to improve statistical reliability.

Table 5 summarizes the hyper-parameters used in training. Shared parameters apply to both SAC and TD3, while algorithm-specific settings for SAC, TD3, and the SIL extension are listed separately. These configurations were kept consistent across tasks to isolate the effects of algorithmic modifications, such as varying the RN or incorporating intrinsic motivation mechanisms.