ACSE: AGENTIC COLLABORATIVE SELF-EVOLUTION FOR EMBODIED INTELLIGENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot policies are typically trained on predefined task and data distributions, with improvements largely driven by data scaling. Despite gains in generalization, this paradigm offers limited support for proactive challenge discovery and continual capability expansion. We propose Agentic Collaborative Self-Evolution (ACSE), a capability-aware framework that couples task generation with solver adaptation through closed-loop interaction. A scene generator uses structured failure evidence to construct physically feasible, informative tasks near the solver's evolving capability boundary. The solver improves through online reinforcement learning, and its updated capabilities guide subsequent task generation, enabling training distributions and solver capabilities to co-evolve. ACSE is designed to accommodate standalone VLA and WAM models as well as agentic dual-brain architectures. In our AWAM-based implementation, ACSE couples experience-driven memory updates in the reasoning brain with parameter adaptation in the world–action brain. Unlike fixed-distribution training or predefined curricula, ACSE adapts the training distribution to evolving capabilities using interaction feedback. Experiments demonstrate sustained capability improvements across evolution rounds and performance gains over imitation learning, fixed-distribution RL, and failure-guided RL baselines, particularly in long-horizon manipulation and out-of-distribution generalization. These results highlight the potential of capability-aware task generation for continual embodied self-improvement.

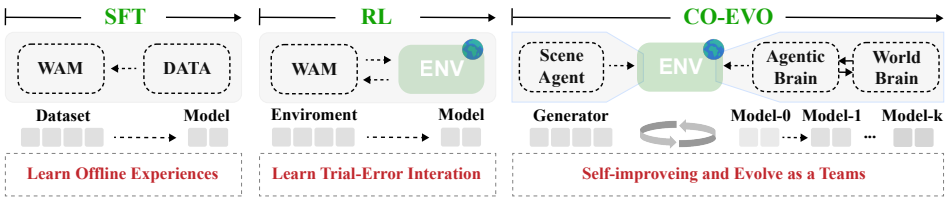


Figure 1: Comparison of learning paradigms. SFT learns from static data, RL from interaction, and ACSE through agentic co-evolution, jointly expanding scenes and agent capability boundaries.

1 INTRODUCTION

Recent advances in robot learning and foundation models have substantially expanded the capabilities of embodied models. Vision-language-action (VLA) models, such as π_0 (Black et al., 2024), $\pi_{0.5}$ (Black et al., 2025), and OpenVLA (Kim et al., 2024), use multimodal knowledge and robot interaction trajectories to generalize across manipulation tasks. Meanwhile, world action models (WAMs), such as DreamZero (Ye et al., 2026b), Fast-WAM (Yuan et al., 2026), and Fast-WAM (Zhao et al., 2026), incorporate predictive world modeling to guide action generation. To improve generalization under distribution shifts and enhance the reliability of long-horizon execution, agentic frameworks such as Harness VLA (Zhang et al., 2026b) and HarnessWAM (Gu et al., 2026) integrate high-level reasoning with VLA or WAM models for planning, execution monitoring, and failure recovery. Despite these advances, a central challenge remains: translating interaction experience into training tasks that match the solver's evolving capabilities. Improved execution performance alone does not establish a learning loop in which capability gaps guide task generation and newly acquired capabilities reshape the training distribution.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Embodied learning faces a mismatch between static training distributions and evolving solver capabilities. Mastered tasks offer diminishing learning value, while complex interactions expose new capability gaps. Curriculum learning and automatic environment design adjust task difficulty, while generative robot learning expands the task space (Wang et al., 2024a;b). However, predefined difficulty schedules cannot readily adapt to continually evolving solver capabilities, whereas simply maximizing task diversity or difficulty may generate tasks with limited learning value (Wang et al., 2019; 2020; Dennis et al., 2020; Jiang et al., 2021b). We argue that the key to continually evolving embodied intelligence is not to generate ever-harder tasks but to actively explore the solver's current capability boundary and construct tasks that are challenging, physically feasible, and informative for learning. This leads to our central question: Can a robotic solver continually identify and expand its own capability boundary through closed-loop interaction?

To this end, we introduce Agentic Collaborative Self-Evolution (ACSE), a framework that couples scene generation with solver learning in a continual closed-loop process. As illustrated in Fig. 2, ACSE comprises a scene generator and a robotic solver that co-evolve through closed-loop interaction. The robotic solver is an embodied model that learns from generated tasks and interaction feedback to continually improve its capabilities. Based on the solver's closed-loop interaction outcomes, the scene generator continually identifies unresolved capability gaps and generates informative tasks near the solver's current capability boundary. These tasks are instantiated in a closed-loop simulator, where the solver improves through reinforcement learning. As the solver progressively overcomes existing challenges, its capability boundary expands, which in turn drives the generator to discover new capability gaps. This process forms a closed loop in which the task distribution and the solver's capabilities continually co-evolve.

ACSE enables collaborative self-evolution beyond static data scaling and predefined curricula. In each round, the generator targets the solver's capability gaps, while the solver learns to overcome these challenges and guides subsequent task generation. Grounded in physical feedback, this adversarial interaction drives task distributions and solver capabilities to co-evolve. To support generalization and reliable long-horizon execution, embodied architectures are expanding from direct-action VLAs to predictive WAMs and further to Agent–WAM dual-brain systems integrating reasoning, world modeling, and action generation. For these architectures, ACSE couples experience-driven memory evolution with world–action model adaptation, jointly improving high-level reasoning and low-level execution. Our main contributions are summarized as follows:

- We propose Agentic Collaborative Self-Evolution (ACSE), a capability-aware framework that couples task generation with solver evolution through closed-loop interaction. ACSE supports both standalone VLA and WAM models as well as agentic dual-brain architectures, enabling training distributions and solver capabilities to co-evolve.
- We develop a failure-guided task generation mechanism that combines structured failure attribution, constrained skill composition, and simulation-based validation to construct executable task and scene variants targeting the solver's identified capability gaps.
- Extensive experiments demonstrate that ACSE achieves sustained capability improvements across multiple evolution rounds and outperforms imitation learning, fixed-distribution RL, and failure-guided RL baselines, particularly in long-horizon manipulation and out-of-distribution generalization.

2 Related Work

2.1 Foundation Models for Embodied Intelligence

Large-scale robot data and vision-language foundation models have advanced embodied learning from task-specific policies toward generalist VLA models. By jointly modeling vision, language, and actions, VLA models exhibit improved cross-task, cross-embodiment, and open-world generalization (Brohan et al., 2022; Zitkovich et al., 2023; Octo Model Team et al., 2024; Kim et al., 2024; Black et al., 2024; 2025). However, they primarily learn observation-to-action mappings through action supervision, leaving physical dynamics largely implicit.

This limitation has motivated the development of world action models (WAMs), which move beyond direct action prediction to jointly model world dynamics and generate actions, making predic-

tive physical representations an integral component of embodied decision-making (Ye et al., 2026b; Kim et al., 2026; Yuan et al., 2026; Zhao et al., 2026). As world modeling improves physical prediction and action generation, recent research has increasingly focused on orchestrating these capabilities for long-horizon tasks. Emerging agentic harness approaches augment pretrained policies with reasoning, memory, task decomposition, capability modeling, and failure recovery, thereby orchestrating frozen VLA models, WAMs, or heterogeneous robot policies as reusable embodied capabilities (Zhang et al., 2026b; Li et al., 2026b; Huang et al., 2026; Gu et al., 2026). These methods cover vision-language modeling, world modeling, and agentic reasoning, yet leave task–capability co-evolution underexplored. ACSE couples failure-guided task generation with solver adaptation, using closed-loop feedback to align the training distribution with evolving solver capabilities.

2.2 Self-Evolving Embodied Learning

Adapting training experiences to evolving agent capabilities has long been a central goal of curriculum learning and open-ended learning. Early approaches organized or automatically adapted task difficulty to construct progressive curricula (Bengio et al., 2009; Graves et al., 2017; Portelas et al., 2020), while subsequent work on open-ended learning and unsupervised environment design (UED) coupled environment generation with agent learning, allowing training environments to evolve alongside the learner (Wang et al., 2019; 2020; Dennis et al., 2020; Jiang et al., 2021b;a; Parker-Holder et al., 2022). These approaches have laid the foundation for adaptive task generation but largely focus on procedurally generated reinforcement learning environments rather than semantically rich embodied tasks with complex physical constraints.

Foundation models have extended this automation to the generation of robot tasks, scenes, rewards, and data, shifting robot learning from fixed data collection toward autonomous experience generation (Wang et al., 2024a;b; Ma et al., 2024a;b; Pfaff et al., 2026). In parallel, autonomous data collection, continual reinforcement learning, and world-model-based imagination have enable models to progressively improve via experience gained from their own interactions (Bousmalis et al., 2023; Lei et al., 2026; Zeng et al., 2026; Liu et al., 2025; Yang et al., 2026; Jiang et al., 2026). The former expands training experience, while the latter improves models through interaction data. Yet these works remain largely decoupled, lacking a closed loop driven by evolving agent capabilities. ACSE generates tasks near the solver's evolving capability boundary, as revealed through closed-loop interactions. These tasks drive solver improvement, while its updated capabilities guide subsequent task generation, enabling the task distribution and solver capabilities to co-evolve.

3 Method

3.1 Agentic Self-Evolution Formulation

We formulate continual embodied self-evolution as a two-player game between a scene generator $\mathcal{G}_\phi$ and a robotic solver $\mathcal{S}_\theta$, as shown in Fig 2. The generator induces a distribution $q_\phi(T)$ over embodied tasks $T \in \mathcal{T}$, while the solver executes a policy π_θ through closed-loop interaction with the environment. Given a task T, the solver generates a trajectory $\tau \sim p_\theta(\cdot \mid T)$ and receives an episodic return $R(T, \tau)$. We define the expected closed-loop utility as $V(\theta, \phi) = \mathbb{E}_{T\sim q_\phi,\, \tau\sim p_\theta(\cdot|T)}\,[R(T, \tau)]$.

Unlike embodied learning over fixed or predefined task distributions, ACSE evolves the training distribution alongside the solver's capabilities. At round k, the generator operates within a capability-aware strategy space Φ^k, inducing a task distribution focused on physically feasible, capability-relevant challenges. The round-wise generator–solver game is formulated as:

$$\mathcal{J}^k \triangleq \max_{\theta\in\Theta}\ \min_{\phi\in\Phi^k}\ \mathbb{E}_{T\sim q_\phi,\ \tau\sim p_\theta(\cdot|T)}[R(T,\tau)]\,. \tag{1}$$

where Θ denotes the solver strategy space. The generator targets the incumbent solver's capability gaps, while the solver maximizes its return on the generated tasks, making their interaction adversarial within each round. Across rounds, solver improvements reshape the capability-aware generator strategy space, exposing new gaps and driving task distributions and solver capabilities to co-evolve.

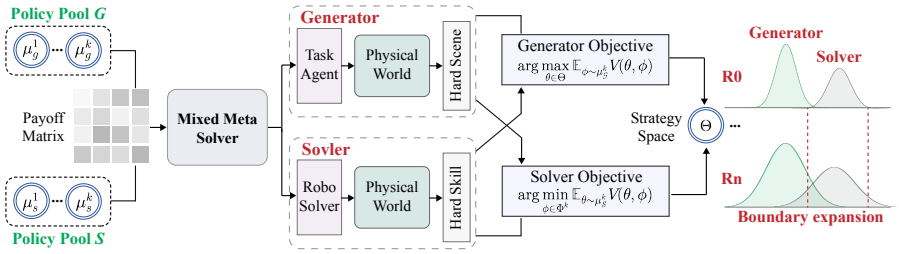


Figure 2: Overview of Agentic Collaborative Self-Evolution. It couples the task agent and robotic solver through a population-based meta-game. The generator explores capability-boundary challenges, while the solver learns to overcome them, driving task–capability co-evolution.

3.2 Capability-Aware Scene Generator

The generator determines what the solver should learn next. Each strategy ϕ in Eq. 1 induces a task distribution $q_\phi(T)$. ACSE uses closed-loop evidence to target capability gaps, verify physical feasibility, and select challenges matched to the solver's current capabilities.

Capability-Conditioned Task Generation. At round k, the solver $\mathcal{S}_{\theta^k}$ produces trajectories $\tau \sim p_{\theta^k}(\cdot \mid T)$. An agentic analyzer extracts evidence $\xi = \mathcal{A}(T, \tau)$ covering skill outcomes, object states, unmet conditions, and temporal dependencies. Successful executions provide reference conditions, while failures indicate potential capability gaps. Let ν^k denote the empirical evidence distribution. The induced task distribution is $q_\phi(T) = \mathbb{E}_{\xi\sim\nu^k}[q_\phi(T \mid \xi)]$, where ξ includes the source task and its constraints. This evidence guides mutations in task structure, scene configuration, language expression, and visual observation. Operators specify the changes, while closed-loop evaluation estimates the difficulty of the resulting tasks for the current solver. Task mutations remain within the supported base-task scope, while language and visual mutations preserve task semantics.

Executable Task Construction. An LLM translates the selected mutation into a structured specification defining the instruction, object roles, goal, and constraints. Object roles are bound to compatible assets, and the scene is initialized with specified poses, joint states, and spatial relations. The task is organized as a graph of supported skills, each expanded into a grounded primitive sequence with explicit arm assignments and execution dependencies. Success conditions are instantiated using supported simulator predicates and evaluators. Scene initialization and reference execution check scene validity, primitive completion, and task feasibility. Invalid candidates are revised or rejected. The current solver then evaluates valid candidates to estimate their difficulty relative to its capabilities. The mutation operators and generation pipeline are detailed in Appendix A.1 and Appendix A.2.

Capability-Aware Strategy Space. Physical feasibility does not imply suitability for learning. Let $c_{\theta^k}(T)$ denote the current solver's completion probability, defined in Sec. 3.3. We define the capability boundary $\mathcal{B}^k$ as the set of physically valid tasks satisfying $c_{\min} \le c_{\theta^k}(T) \le c_{\max}$, where $0 < c_{\min} \le c_{\max} < 1$. This selects tasks that are neither consistently mastered nor currently beyond the solver's reach. The ideal admissible generator strategy space is $\Phi^k \triangleq \{\phi \in \Phi \mid q_\phi(\mathcal{B}^k) \ge 1-\delta\}$, where Φ is the full generator strategy space, $q_\phi(\mathcal{B}^k)$ denotes the probability mass assigned to $\mathcal{B}^k$, and $\delta \in [0, 1)$ controls the mass allowed outside it. In practice, we approximate this restriction by filtering validated candidates using empirical completion estimates $\hat{c}_{\theta^k}(T)$ and capability thresholds. Selected tasks enter the training pool, while interaction feedback guides generation toward the solver's evolving capability boundary without a predefined difficulty schedule.

3.3 Agentic World-Action Solver

We use an Agentic World Action Model as the solver. It is a dual-brain architecture comprising an Agentic Reasoning Brain based on Qwen3.5 Qwen Team (2026) and a World-Action Brain based on Wan2.2 Team Wan et al. (2025). The former uses task context and accumulated memory to generate high-level intents, specifying *what should be achieved next*. The latter combines predictive world modeling with action decoding to determine *how these intents can be physically realized* and generates executable robot actions. This design decomposes the solver into two coupled modules

for agentic reasoning and world–action modeling, enabling continual evolution through experience-driven memory updates and model parameter adaptation.

Agentic world-action policy. For a task T, let $x_t = (T, o_t, h_t)$ denote the decision context at time step t, where o_t is the current observation and h_t summarizes the interaction history. Let m denote the solver's persistent memory, updated from interaction experience. The Agentic Reasoning Brain generates a structured intent $z_t \sim \pi_{\mathrm{A}}(\cdot \mid x_t; m)$, where $z_t \in \mathcal{Z}$ and $\mathcal{Z}$ denotes the discrete intent space. Conditioned on this intent, the World-Action Brain predicts a continuous world representation $w_t \sim \pi_{\mathrm{W}}(\cdot \mid o_t, z_t; \omega)$, from which an action $a_t \sim \pi_{\mathrm{C}}(\cdot \mid w_t; \omega)$ is decoded. Marginalizing over these intermediate variables yields the effective solver policy:

$$\pi_\theta(a_t \mid x_t) = \sum_{z_t \in \mathcal{Z}} \pi_{\mathrm{A}}(z_t \mid x_t; m) \int \pi_{\mathrm{C}}(a_t \mid w_t; \omega)\, \pi_{\mathrm{W}}(w_t \mid o_t, z_t; \omega)\, \mathrm{d}w_t, \theta = (m, \omega) \in \Theta. \quad (2)$$

Here, θ denotes the solver's evolvable configuration, with m updated through interaction experience and ω adapted through online reinforcement learning. Actions are decoded solely from the world representation w_t, with intents indirectly guiding action generation through w_t.

Closed-Loop Capability. The policy in Eq. 2 governs the solver's closed-loop interaction with the environment. Let $\mathcal{P}(s_{t+1} \mid s_t, a_t, T)$ denote the environment dynamics and $\rho_0(s_0 \mid T)$ the initial-state distribution. The resulting trajectory distribution for task T is as follows:

$$p_\theta(\tau \mid T) = \rho_0(s_0 \mid T) \prod_{t=0}^{H-1} \pi_\theta(a_t \mid x_t)\, \mathcal{P}(s_{t+1} \mid s_t, a_t, T). \quad (3)$$

For simplicity, we do not explicitly include the observation kernel. The utility $V(\theta, \phi)$ reflects the solver's closed-loop behavior rather than isolated action predictions. If $C(T, \tau) \in [0, 1]$ measures task completion, we define the solver's capability on task T and its empirical estimate as follows:

$$c_\theta(T) \triangleq \mathbb{E}_{\tau \sim p_\theta(\cdot \mid T)}[C(T, \tau)], \quad \hat{c}_\theta(T) = \frac{1}{N} \sum_{n=1}^{N} C(T, \tau^{(n)}), \quad (4)$$

where $\tau^{(n)} \sim p_\theta(\cdot \mid T)$ and θ is held fixed during evaluation. The estimate $\hat{c}_\theta(T)$ links solver evaluation to the capability-aware generator. The capability boundary is therefore identified from the current solver's closed-loop behavior rather than specified independently by the generator.

Dual-level Solver Evolution. The decomposition $\theta = (m, \omega)$ defines two complementary modes of solver evolution. Memory m is updated with successful strategies, failure patterns, and reusable experience to guide agentic decisions while keeping the reasoning model's parameters frozen. We refer to this process as Harness evolution. Meanwhile, the World-Action Brain parameters ω are adapted through online reinforcement learning using closed-loop interaction, improving execution capabilities and reliability. Together, these updates enable complementary evolution at the experience and model levels.

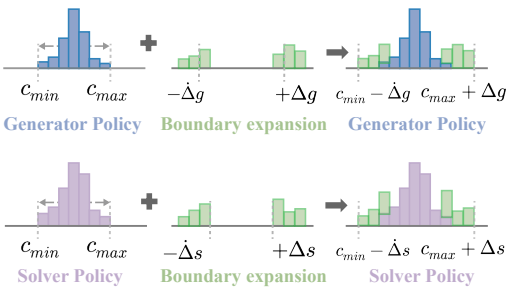


Figure 3: Generator and solver policy boundary expansion during collaborative self-evolution.

This dual-level evolution supports the collaborative ACSE loop. At round k, the solver $\theta^k = (m^k, \omega^k)$ induces the trajectory distribution $p_{\theta^k}(\tau \mid T)$ and capability function $c_{\theta^k}(T)$, which inform capability boundary estimation and the admissible generator strategy space Φ^k, as shown in Fig 3. Updating memory and model parameters yields θ^{k+1}, whose updated capabilities guide task generation in the next round. It provides a concrete implementation of this process within the general ACSE framework. We next formalize the resulting reciprocal updates as an iterative procedure for solving the generator–solver game.

3.4 Game-Theoretic Collaborative Self-Evolution

The preceding sections define the players in Eq. 1 as the scene generator and the robotic solver. We now formalize their repeated interaction as a population-based game. Unconstrained adversarial generation may favor infeasible or nearly unsolvable tasks that offer little learning value. ACSE restricts the search to Φ^k, targeting capability gaps while reducing the risk that all rollouts fail.

Generator–Solver Meta-Game. At evolution round k, let $\mathbb{S}^k = \{\theta_1, \ldots, \theta_n\} \subset \Theta$ and $\mathbb{G}^k = \{\phi_1, \ldots, \phi_m\} \subset \Phi^k$ denote the solver and generator populations, respectively. Historical generator strategies are included in the current population only if they still belong to Φ^k. Each solver strategy fixes its memory and model parameters. Pairwise closed-loop utilities define the payoff matrix M^k, with $M^k_{ij} = V(\theta_i, \phi_j)$. We formulate the restricted zero-sum game as follows:

$$v^k = \max_{\mu_{\mathcal{S}} \in \Delta(\mathbb{S}^k)} \min_{\mu_{\mathcal{G}} \in \Delta(\mathbb{G}^k)} \mu_{\mathcal{S}}^\top M^k \mu_{\mathcal{G}} \tag{5}$$

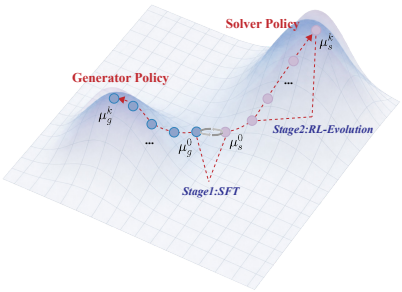


Figure 4: Illustration of generator–solver co-evolution along an expanding capability boundary.

where $\mu_{\mathcal{S}}$ and $\mu_{\mathcal{G}}$ denote the solver and generator meta-strategies, respectively, and $\Delta(\cdot)$ denotes the probability simplex over a population. Let $\mu^k_{\mathcal{S}}$ and $\mu^k_{\mathcal{G}}$ denote the meta-strategies obtained for round k. The strategy space Φ^k is constructed based on the current solver's estimated capability boundary at the start of the round and remains fixed during meta-strategy optimization and response computation. The populations are then expanded by adding strategies that respond to the current opponent mixtures, with ideal best responses defined as follows:

$$\theta^k_{\text{br}} \in \arg\max_{\theta \in \Theta} \mathbb{E}_{\phi \sim \mu^k_{\mathcal{G}}} V(\theta, \phi), \phi^k_{\text{br}} \quad \in \arg\min_{\phi \in \Phi^k} \mathbb{E}_{\theta \sim \mu^k_{\mathcal{S}}} V(\theta, \phi) \tag{6}$$

In practice, these responses are approximated through solver updates and constrained task generation. The solver searches over Θ, whereas the generator's response is restricted to Φ^k. This asymmetry preserves adversarial pressure while limiting the generation of excessively difficult tasks.

Capability-Boundary Feedback. Let $C(T, \tau) \in \{0, 1\}$ denote the task-completion indicator and $c_{\theta^k}(T)$ the corresponding closed-loop completion probability of the current solver. We characterize a generator's completion-feedback variability as follows:

$$\Gamma^k(\phi) = \mathbb{E}_{T \sim q_\phi}\left[\text{Var}_{\tau \sim p_{\theta^k}(\cdot|T)}[C(T, \tau)]\right] = \mathbb{E}_{T \sim q_\phi}\left[c_{\theta^k}(T)(1 - c_{\theta^k}(T))\right] \tag{7}$$

The equality follows from $C(T, \tau)^2 = C(T, \tau)$. For a fixed task, the variance peaks at a completion probability of $1/2$ and vanishes at zero or one, motivating physically feasible tasks with intermediate completion probabilities. ACSE favors such tasks using empirical capability estimates, with feedback variance serving as a sampling rationale rather than a measure of learning progress.

Capability-Aware Meta-Game Optimization. We next characterize optimization of the population game held fixed within each round. For mixed strategies, define $U^k(\mu_{\mathcal{S}}, \mu_{\mathcal{G}}) = \mathbb{E}_{\theta \sim \mu_{\mathcal{S}}, \phi \sim \mu_{\mathcal{G}}}[V(\theta, \phi)]$. We measure convergence using the restricted duality gap as follows:

$$\text{Gap}_k(\mu_{\mathcal{S}}, \mu_{\mathcal{G}}) = \max_{\theta \in \mathbb{S}^k} U^k(\theta, \mu_{\mathcal{G}}) - \min_{\phi \in \mathbb{G}^k} U^k(\mu_{\mathcal{S}}, \phi) \tag{8}$$

Pure-strategy arguments in U^k denote point-mass distributions. The gap sums the players' maximum unilateral gains and vanishes at a nash equilibrium of the restricted game.

For fixed populations and payoffs in $[0, 1]$, let $\bar{\mu}^L_{\mathcal{S}}$ and $\bar{\mu}^L_{\mathcal{G}}$ denote meta-strategies averaged over L iterations. Full-information multiplicative-weights updates with suitable learning rates give the following bound via cumulative external regret bounds $\mathcal{R}^L_{\mathcal{S}}$ and $\mathcal{R}^L_{\mathcal{G}}$:

$$\text{Gap}_k\left(\bar{\mu}^L_{\mathcal{S}}, \bar{\mu}^L_{\mathcal{G}}\right) \leq \frac{\mathcal{R}^L_{\mathcal{S}} + \mathcal{R}^L_{\mathcal{G}}}{L} = O\left(\sqrt{\frac{\log|\mathbb{S}^k|}{L}} + \sqrt{\frac{\log|\mathbb{G}^k|}{L}}\right) \tag{9}$$

Adding the regret bounds cancels the realized payoffs, and bilinearity yields the bound for averaged meta-strategies. Thus, the restricted gap vanishes as $L \to \infty$. The same analysis applies to the empirical gap $\widehat{\text{Gap}}_k$ of a fixed rollout-estimated matrix $\widehat{M}^k$, with accuracy relative to the true gap depending on payoff estimation. This guarantees convergence within a fixed population game, not across evolving populations or capability-dependent games.

Approximate Best-Response Optimization. Exact best responses in Eq. 6 are generally intractable in high-dimensional task and policy spaces, so we approximate both operators. The scene generator

Algorithm 1 Agentic Collaborative Self-Evolution

Require: Generator $\mathcal{G}$, Solver $\theta^0 = (m^0, \omega^0)$, simulator $\mathcal{E}$
Initialize Solver and Generator populations Θ^0, Φ^0 and training pool $\mathcal{P}^0$
for $k = 0, \ldots, K-1$ **do**
  $\mathcal{D}^k \leftarrow \textsc{Rollout}(\mathcal{E}, \theta^k)$; $\nu^k \leftarrow \mathcal{A}(\mathcal{D}^k)$ *// Capability Analysis*
  $\widetilde{\mathcal{T}}^k \leftarrow \textsc{Generate}(\mathcal{G}, \nu^k)$ *// Task Generation*
  Retain valid tasks $\mathcal{T}_+^k$ with $c_{\min} \le \hat{c}_{\theta^k}(T) \le c_{\max}$ *// Task Filtering*
  Update training pool $\mathcal{P}^{k+1}$ with $\mathcal{T}_+^k$ and construct admissible population Φ^k
  $M_{ij}^k \leftarrow V(\theta_i, \phi_j), \theta_i \in \Theta^k,\ \phi_j \in \Phi^k$ *// Payoff Evaluation*
  $(\bar{\mu}_{\mathcal{S}}^k, \bar{\mu}_{\mathcal{G}}^k) \leftarrow \mathrm{MWU}(M^k)$ with populations fixed *// Meta-Strategy Optimization*
  Update ω^{k+1} via RL against $\bar{\mu}_{\mathcal{G}}^k$, holding m^k fixed *// World-Action Adaptation*
  $m^{k+1} \leftarrow \textsc{Consolidate}(m^k, \text{collected interactions})$ *// Harness Evolution*
  $\theta^{k+1} \leftarrow (m^{k+1}, \omega^{k+1})$; update populations
end for
return θ^K

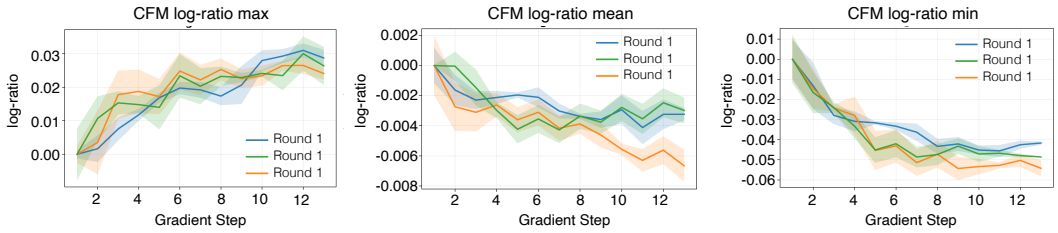


Figure 5: Policy update dynamics across three evolution rounds. Maximum, mean, and minimum CFM log-ratios are plotted against gradient steps, with each color representing one round.

in Sec. 3.2 uses capability evidence ν^k to synthesize diverse tasks within Φ^k, providing a heuristic generator response without model parameter optimization. As illustrated in Fig. 4, the generator and solver co-evolve after SFT initialization, progressively expanding the capability boundary. The overall procedure is summarized in Algorithm 1.

In our implementation, Agentic World-Action Model serves as the solver. Sampling generator strategies from $\mu_{\mathcal{G}}^k$ induces the task distribution for online reinforcement learning. With Harness memory m^k held fixed during the parameter-update stage, the World-Action Brain is trained to approximately maximize the following expected-return objective:

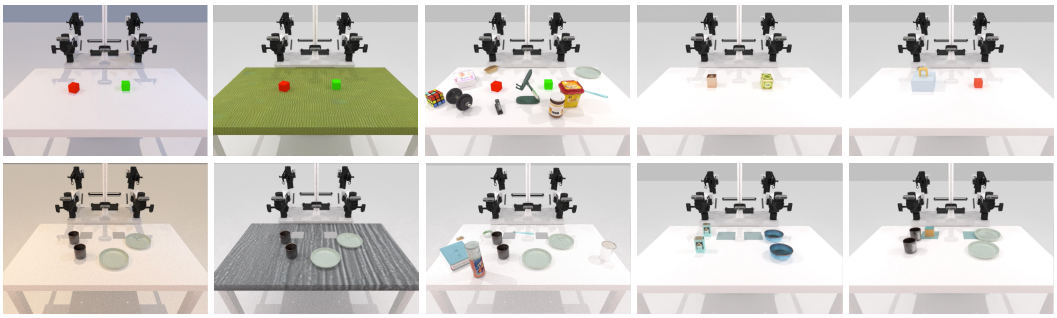
$$\max_{\omega} \mathbb{E}_{\phi \sim \mu_{\mathcal{G}}^k,\ T \sim q_\phi,\ \tau \sim p_{(m^k,\omega)}(\cdot|T)} \left[R(T, \tau) \right]. \tag{10}$$

Online reinforcement learning updates ω^{k+1}, while interaction experience is consolidated into m^{k+1}, yielding $\theta^{k+1} = (m^{k+1}, \omega^{k+1})$. Evaluating the updated solver informs its capability boundary and the next strategy space Φ^{k+1}. Adversarial optimization fixes the strategy space within each round, while solver adaptation guides future tasks, linking within-round competition to cross-round collaborative self-evolution. To characterize the reinforcement learning updates within this process, we track CFM log-ratios across three evolution rounds as shown in Fig 5. Their range generally widens with gradient steps, while the mean exhibits a small negative shift, providing a local view of policy update dynamics.

4 Experiments

We evaluate ACSE on multiple benchmarks for long-horizon manipulation, skill composition, and out-of-distribution generalization. We track performance across evolution rounds and conduct ablations on task generation, population-based optimization, Harness evolution, and World-Action adaptation to assess their effects on learning efficiency, failure recovery, and generalization.

Benchmarks. We evaluate ACSE on three complementary manipulation benchmarks. RoboDojo tests memory, precision, instruction following, and long-horizon generalization (Chen et al., 2026).



(a) Lighting Variation (b) Background Variation (c) Distractor Object Addition (d) Asset Replacement (e) Novel Task Introduction

Figure 6: Generated task variants for block (top) and tableware (bottom) manipulation, covering lighting, backgrounds, distractors, asset replacement, and novel tasks.

Table 1: ACSE performance on the RoboDojo simulation benchmark. Entries report Score/Sucess Rate; higher is better. The init solver method is agentic world-action model.

Method	Type	Generalization	Precision	Long-Horizon	Memory	Open	Average
ACT	IL	0.69/0.56	0.85/0.00	1.73/0.92	1.65/0.13	0.00/0.00	0.98/0.32
π_0	VLA	3.94/2.56	3.56/0.75	6.19/2.00	3.47/2.11	0.25/0.25	3.48/1.53
RDT-1B	VLA	0.56/0.33	0.38/0.00	1.13/0.00	0.49/0.33	0.00/0.00	0.51/0.13
X-VLA	VLA	10.48/6.78	18.32/12.00	16.53/9.75	4.76/3.56	0.55/0.50	10.13/6.52
$\pi_{0.5}$	VLA	13.37/8.17	12.40/5.50	23.54/14.67	5.78/4.56	1.98/1.67	11.41/6.91
Spatial Forcing	VLA	14.12/9.33	17.33/10.58	23.26/14.58	5.43/4.11	1.78/1.58	12.38/8.04
Hy-Embodied-0.5-VLA	VLA	11.77/8.39	13.81/8.00	25.74/14.92	13.37/12.11	0.65/0.58	13.07/8.80
Fast-WAM	WAM	2.34/1.11	1.96/0.00	9.14/5.17	3.55/3.44	0.42/0.42	3.48/2.03
GigaWorld-Policy	WAM	5.34/2.89	6.15/1.83	15.51/8.92	3.46/2.22	0.54/0.50	6.20/3.27
X-WAM	WAM	7.39/3.33	6.72/1.83	17.47/9.08	6.32/4.67	0.57/0.25	7.69/3.83
OpenWAM	WAM	20.71/14.83	18.45/9.25	34.93/25.33	10.41/9.11	1.41/1.08	17.18/11.92
ACSE (init solver)	Agent+WAM	18.13/ 11.53	14.21/ 6.31	40.74/ 29.92	25.22/ 21.69	2.13/ 1.85	20.08/ 14.26
ACSE (evolved solver)	Agent+WAM	23.37/ 14.13	17.52/ 9.28	47.08/ 34.04	29.19/ 27.16	4.61/ 2.91	24.35/ 17.50

BiCoord assesses bimanual coordination across sequential subgoals (Peng et al., 2026). Together, they assess capability gains across complex tasks, coordination demands, and distribution shifts.

Baselines. We evaluate against two groups of baselines. Direct action-generation methods comprise ACT Zhao et al. (2023), RDT-1B (Liu et al., 2024), OpenVLA-OFT (Kim et al., 2025), X-VLA Zheng et al. (2026), π_0 (Black et al., 2024), $\pi_{0.5}$ Black et al. (2025), Hy-Embodied-0.5 Zhang et al. (2026a), and Spatial Forcing Li et al. (2026a). World action models that include GigaWorld-Policy (Ye et al., 2026a), Fast-WAM (Yuan et al., 2026), X-WAM (Guo et al., 2026), and OpenWAM (Wang et al., 2026). All baselines follow official configurations.

4.1 Main Experimental Results

Comparison with Existing Methods. Tables 1 and 2 present comparative results on RoboDojo and BiCoord, respectively. ACSE achieves the best average performance on both benchmarks. On RoboDojo, the evolved Solver improves both score and success rate over its initialization across all five categories, with the largest gains in generalization and precision. Improvements on long-horizon and memory tasks further support more reliable multi-stage execution. On BiCoord, higher average task and spatial success rates, together with shorter task lengths, indicate more effective and efficient bimanual manipulation.

Capability Gains across Evolution Rounds. Table 3 shows that ACSE's task and stage-wise success rate improve across all four evolution rounds, while task length consistently decreases. The gains become smaller in later rounds, suggesting diminishing returns from further evolution. The fourth round achieves the highest success rates and shortest task length, supporting the benefits of successive evolution rounds for both task completion and execution efficiency.

Failure-Driven Task Generation. Fig. 6 illustrates how failure-driven task generation expands training scenarios through lighting and background changes, added distractors, asset replacement,

Table 2: ACSE multi-task performance on BiCoord. SR, SSR, and TL denote Success Rate, Stage-wise Success Rate, and Task Length. The init solver method is agentic world-action model.

Task	OpenVLA-OFT			π_0			ACSE (initial)			**ACSE (evolved)**		
	SR ↑	SSR ↑	TL↓	SR ↑	SSR ↑	TL↓	SR ↑	SSR ↑	TL↓	SR ↑	SSR ↑	TL↓
Balance Roller	49.0	74.5	137	68.0	81.5	151	78.0	89.0	140	78.0	88.0	112
Build Bridge	2.0	2.0	312	44.0	47.0	289	55.0	65.0	336	57.0	68.3	308
Build Tower With Blocks	0.0	0.0	–	2.0	5.0	650	8.0	21.0	700	13.0	28.5	616
Clean Table	2.0	25.0	612	6.0	46.2	658	18.0	56.8	756	46.0	68.5	672
Collect Pens	5.0	46.2	795	47.0	81.8	457	60.0	86.0	560	68.0	86.0	476
Cook	10.0	25.0	442	14.0	29.5	504	28.0	42.0	420	36.0	54.8	392
Divide Block Tower	0.0	11.1	–	0.0	10.8	–	8.0	27.0	588	15.0	32.0	504
Exchange Mics	66.0	67.5	418	52.0	59.5	446	63.0	69.0	532	67.0	75.5	476
Exchange Pots	53.0	56.0	442	52.0	60.5	476	62.0	67.0	532	65.0	75.8	476
Extract Bottom Block To Top	53.0	55.5	177	40.0	43.5	208	50.0	53.5	364	52.0	60.3	308
Fetch Block With Roller	44.0	70.5	225	43.0	69.5	250	55.0	71.5	252	61.0	78.0	252
Handover Block With Bowls	0.0	3.0	–	4.0	9.0	188	12.0	16.0	336	14.0	27.0	280
Jigsaw	0.0	25.2	–	6.0	36.5	400	16.0	45.5	476	26.0	52.3	420
Match Blocks With Signs	5.0	8.7	760	8.0	18.7	638	17.0	41.0	616	31.0	47.0	532
Place Plate And Cup	55.0	79.2	315	25.0	66.2	302	40.0	70.8	308	58.0	80.5	308
Put Objects Cabinet	26.0	49.5	486	6.0	24.0	575	20.0	36.0	644	31.0	51.3	560
Stack Bowls	1.0	14.0	400	12.0	28.0	471	26.0	39.5	448	34.0	50.3	392
Sweep Block	44.0	44.0	234	61.0	61.0	253	72.0	72.0	336	76.0	76.0	308
Average	23.1	36.5	411	27.2	43.2	406	38.2	53.8	491	46.0	61.1	410

Table 3: Effect of evolution iterations on Bicoord.

Evo. Round	SR	SSR	TL
1	41.5	57.0	461
2	44.0	59.2	432
3	46.0	61.1	410
4	45.5	60.6	401

Table 4: Ablation of optimization strategies on Bicoord.

Method	**SR**	**SSR**	**TL**
Base	38.2	53.8	491
Random	40.1	55.4	470
Failure-Weighted	41.6	56.3	438
ACSE	46.0	61.1	410

Table 5: Solver evolution ablations on Bicoord.

Model Variant	**SR**	**SSR**	**TL**
w/o Harness Evolution	42.0	57.5	445
w/o World-Action Adaptation	40.5	57.0	459
w/o Both	38.2	53.8	491
ACSE	46.0	61.1	410

and novel tasks. Guided by the Solver's observed failures, these variations target weaknesses in perception, manipulation, and skill composition, adapting training tasks to its evolving capabilities. Gains across most evaluation tasks support the benefits of this self-evolution process.

4.2 Ablation Study

Strategy Optimization. Table 4 shows that random sampling improves performance over the base solver, while failure-weighted sampling further increases task and stage-wise success rate and reduces task length. The results suggest that focusing on challenges based on failure evidence provides more useful training signals. Adaptive meta-strategy weighting achieves the highest success rates and shortest task length, supporting adaptive allocation of training effort across generator strategies.

Solver Evolution Mechanisms. Table 5 shows that disabling either Harness evolution or World-Action adaptation lowers task and stage-wise success rate and increases task length. Removing World-Action adaptation causes a larger performance decline, suggesting that model parameter updates contribute more in this setting. Combining both mechanisms yields the best performance, supporting their complementary roles in experience reuse and action execution. Disabling both recovers the initial solver as a baseline for evaluating Solver evolution.

5 Conclusion

We introduced Agentic Collaborative Self-Evolution (ACSE), which couples capability-aware task generation with continual solver learning through a population-based meta-game. The generator targets the solver's capability boundary, while Harness evolution and World-Action adaptation improve the solver. Our analysis establishes non-degenerate task-completion feedback and convergence of averaged meta-strategies to the Nash equilibrium set of each fixed population game. Experiments on RoboDojo and BiCoord show improved task completion, long-horizon execution, and bimanual coordination, with ablations supporting adaptive weighting and complementary solver updates. Future work will reduce reliance on simulation interactions through efficient capability estimation and extend continual self-evolution to real robots.

AI USE STATEMENT

In this work, we have not used generative AI tools for any tasks with required disclosure, and all such tasks are not applicable to this work. We used generative AI tools only to polish the writing of parts of the manuscript. The authors reviewed all AI-assisted edits to ensure that they preserve the intended technical meaning and contain no unsupported claims. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

For reproducibility, the main paper and Appendices detail our methodology and implementation. We will release the complete source code and model weights.

REFERENCES

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 41–48, 2009.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Proceedings of the 9th Conference on Robot Learning (CoRL)*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40, 2025. URL https://proceedings.mlr.press/v305/black25a.html.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X. Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, Antoine Laurens, Claudio Fantacci, Valentin Dalibard, Martina Zambelli, Murilo Martins, Rugile Pevceviciute, Michiel Blokzijl, Misha Denil, Nathan Batchelor, Thomas Lampe, Emilio Parisotto, Konrad Zolna, Scott Reed, Sergio Gomez Colmenarejo, Jon Scholz, Abbas Abdolmaleki, Oliver Groth, Jean-Baptiste Regli, Oleg Sushkov, Tom Rothorl, Jose Enrique Chen, Yusuf Aytar, Dave Barker, Joy Ortiz, Martin Riedmiller, Jost Tobias Springenberg, Raia Hadsell, Francesco Nori, and Nicolas Heess. Robocat: A self-improving foundation agent for robotic manipulation. *Transactions on Machine Learning Research*, 2023.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. URL https://arxiv.org/abs/2212.06817.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025. URL https://arxiv.org/abs/2506.18088.

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Weijie Wan, Baijun Chen, Haoran Lu, Haowen Yan, Honghao Su, et al. RoboDojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026. URL https://arxiv.org/abs/2607.04434v1.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In *Advances in Neural Information Processing Systems*, volume 33, pp. 13049–13061, 2020.

Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1311–1320. PMLR, 2017.

Zhaopeng Gu, Bingke Zhu, Tianxi Lin, Guibo Zhu, Yingying Chen, Kai Wang, Tingyu Yuan, Chaoyang Zhao, Zhaowen Li, Peng Su, and Jinqiao Wang. Harnesswam: Bridging prediction and deliberation in world action models. *arXiv preprint arXiv:2608.09516*, 2026. URL https://arxiv.org/abs/2608.09516.

Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4d world action modeling from video priors with asynchronous denoising. *arXiv preprint arXiv:2604.26694*, 2026. URL https://arxiv.org/abs/2604.26694.

Jinbang Huang, Yuanzhao Hu, Zhiyuan Li, Ran Qi, Yixin Xiao, Zhanguang Zhang, Mark Coates, Tongtong Cao, and Yingxue Zhang. Roboharness: Memory-driven orchestration of heterogeneous robot policies for long-horizon planning, 2026. URL https://arxiv.org/abs/2607.18060.

Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. In *Advances in Neural Information Processing Systems*, volume 34, pp. 1884–1897, 2021a.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4940–4950. PMLR, 2021b.

Zhennan Jiang, Shangqing Zhou, Yutong Jiang, Zefang Huang, Mingjie Wei, Yuhui Chen, Tianxing Zhou, Zhen Guo, Hao Lin, Quanlu Zhang, Yu Wang, Haoran Li, Chao Yu, and Dongbin Zhao. Wovr: World models as reliable simulators for post-training vla policies with rl. *arXiv preprint arXiv:2602.13977*, 2026. URL https://arxiv.org/abs/2602.13977.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *Robotics: Science and Systems*, 2025. URL https://arxiv.org/abs/2502.19645.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *International Conference on Learning Representations*, 2026.

Kun Lei, Huanyu Li, Dongjie Yu, Zhenyu Wei, Lingxiao Guo, Zhennan Jiang, Ziyu Wang, Shiyu Liang, and Huazhe Xu. Performant robotic manipulation with real-world reinforcement learning. *Science Robotics*, 11(116):eaed6267, 2026. doi: 10.1126/scirobotics.aed6267.

Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. In *International Conference on Learning Representations*, 2026a. URL https://arxiv.org/abs/2510.12276.

Zhuoran Li, Zhiyang Li, Kaijun Zhou, and Jinyu Gu. Roboharness: A memory-augmented policy harness for vision-language-action model robustness via in-context adaptation. *arXiv preprint arXiv:2603.24060*, 2026b. URL https://arxiv.org/abs/2603.24060.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: a Diffusion Foundation Model for Bimanual Manipulation. *arXiv preprint arXiv:2410.07864*, 2024. URL https://arxiv.org/abs/2410.07864.

Zeting Liu, Zida Yang, Zeyu Zhang, and Hao Tang. Evovla: Self-evolving vision-language-action model. *arXiv preprint arXiv:2511.16166*, 2025. URL https://arxiv.org/abs/2511.16166.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*, 2024a.

Yecheng Jason Ma, William Liang, Hung-Ju Wang, Sam Wang, Yuke Zhu, Linxi Fan, Osbert Bastani, and Dinesh Jayaraman. Dreureka: Language model guided sim-to-real transfer. In *Proceedings of Robotics: Science and Systems*, 2024b.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, 2024.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17473–17498. PMLR, 2022.

Xingyu Peng, Chen Gao, Liankai Jin, Annan Li, and Si Liu. BiCoord: A bimanual manipulation benchmark towards long-horizon spatial-temporal coordination. *arXiv preprint arXiv:2604.05831*, 2026. URL https://arxiv.org/abs/2604.05831.

Nicholas Pfaff, Thomas Cohn, Sergey Zakharov, Rick Cory, and Russ Tedrake. Scenesmith: Agentic generation of simulation-ready indoor scenes, 2026. URL https://arxiv.org/abs/2602.09153.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*, 2020. URL https://arxiv.org/abs/2003.04664.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Team Wan, Ang Wang, Baole Ai, Bin Wen, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. URL https://arxiv.org/abs/2503.20314.

Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. Gensim: Generating robotic simulation tasks via large language models. In *International Conference on Learning Representations*, 2024a.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019. URL https://arxiv.org/abs/1901.01753.

Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeffrey Clune, and Kenneth Stanley. Enhanced poet: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9940–9951. PMLR, 2020.

Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 51936–51983. PMLR, 2024b.

Yuran Wang, Siqiao Huang, Mingleyang Li, Chenhao Zhang, Jiaqi Liang, Weiyang Jin, Yue Chen, Xuemin Chi, Donghao Zhou, Qize Yu, et al. Openwam: An open, modular exploration towards systematic world-action model pretraining. *arXiv preprint arXiv:2609.07398*, 2026.

Jiazhi Yang, Kunyang Lin, Jinwei Li, Wencong Zhang, Tianwei Lin, Longyan Wu, Zhizhong Su, Hao Zhao, Ya-Qin Zhang, Li Chen, Ping Luo, Xiangyu Yue, and Hongyang Li. Rise: Self-improving robot policy with compositional world model. *arXiv preprint arXiv:2602.11075*, 2026. URL https://arxiv.org/abs/2602.11075.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, et al. GigaWorld-Policy: An efficient action-centered world–action model. *arXiv preprint arXiv:2603.17240*, 2026a. URL https://arxiv.org/abs/2603.17240.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi Fan, and Joel Jang. World action models are zero-shot policies, 2026b. URL https://arxiv.org/abs/2602.15922.

Brent Yi, Hongsuk Choi, Himanshu Gaurav Singh, Xiaoyu Huang, Takara E. Truong, Carmelo Sferrazza, Yi Ma, Rocky Duan, Pieter Abbeel, Guanya Shi, Karen Liu, and Angjoo Kanazawa. Flow policy gradients for robot control. *arXiv preprint arXiv:2602.02481*, 2026. doi: 10.48550/arXiv.2602.02481. URL https://arxiv.org/abs/2602.02481.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026. URL https://arxiv.org/abs/2603.16666.

Qixin Zeng, Shuo Zhang, Hongyin Zhang, Renjie Wang, Han Zhao, Libang Zhao, Runze Li, Donglin Wang, and Chao Huang. Crl-vla: Continual vision-language-action learning. *arXiv preprint arXiv:2602.03445*, 2026. URL https://arxiv.org/abs/2602.03445.

He Zhang, Lingzhu Xiang, Haitao Lin, Zeyu Huang, Minghui Wang, Dingyan Zhong, Yubo Dong, Yihao Wu, Yongming Rao, Dongsheng Zhang, et al. Hy-embodied-0.5-vla: From vision-language-action models to a real-world robot learning stack. *arXiv preprint arXiv:2606.14409*, 2026a. URL https://arxiv.org/abs/2606.14409.

Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, Zhengru Fang, Yi Nie, Changxu Wei, Yu Wang, Wenbo Ding, and Chao Yu. Harness vla: Steering frozen vlas into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026b. URL https://arxiv.org/abs/2607.08448v1.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.

Weiheng Zhao, Haoyi Jiang, Xin Shi, Liu Liu, Fan Huang, Zhizhong Su, Wei Sui, and Xinggang Wang. Faster-wam: Efficient inference-time future conditioning for robust world action models. *arXiv preprint arXiv:2608.04404*, 2026. URL https://arxiv.org/abs/2608.04404.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=kt51kZH4aG.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023.

A Implementation Details of Scene Generator

We develop an LLM-based, capability-aware scene generator that synthesizes executable embodied tasks based on the solver's failure evidence and current capabilities. The generator first diagnoses failed rollouts and attributes them to specific reasoning, world-modeling, or action-execution failures, then generates structured tasks and scene configurations targeting the identified capability gaps. The generated tasks are validated through closed-loop interaction and used for online reinforcement learning, allowing the solver to continually adapt to newly exposed challenges.

A.1 Failure Attribution and Targeted Mutation

We analyze unsuccessful closed-loop rollouts to identify the solver's capability gaps and guide task generation. Failure attribution uses task context, execution traces, and observed outcomes without presuming a specific faulty module. The resulting evidence guides the selection of mutation operators across the four dimensions listed in Table 6. These dimensions describe what is modified rather than mutually exclusive failure causes or predefined difficulty levels.

Task Structure (T). Task mutations target weaknesses in skill selection, composition, and temporal coordination. Starting from the original plan (T_0), T_1 selects alternative goal-preserving skill realizations, T_2 recombines compatible skills, and T_3 varies valid execution-order and coordination constraints. All changes respect the task scope, supported skills, and verifiable conditions.

Physical Scene Configuration (S). Scene mutations probe sensitivity to physical configurations while preserving the task objective. S_0 retains the original scene, while S_1–S_4 vary asset instances or sizes, initial object or joint states, spatial relations, and clutter or obstacles, respectively. Candidates must maintain functional compatibility and physical feasibility.

Language Expression (W). Language mutations test instruction understanding without changing task semantics. W_0 preserves the original instruction, while W_1–W_3 introduce lexical, referential, and syntactic variations, respectively. References must remain unambiguous, with target entities, goals, and ordering requirements unchanged.

Visual Observation (V). Visual mutations probe robustness to observation changes. Starting from the original settings (V_0), V_1–V_4 vary illumination, surface appearance, camera viewpoint, and observation noise, respectively. These changes preserve physical properties and task semantics while retaining essential visual cues.

Failure evidence guides operator selection, while closed-loop evaluation estimates the resulting difficulty for the current solver. A mutation may affect multiple capabilities and does not by itself establish the cause of failure. For the AWAM-based solver, planning-related experience also informs subsequent Harness memory updates without changing the reasoning model's parameters. The conversion of selected mutations into executable tasks and scenes is detailed in Sec. A.2.

A.2 Capability-Aware Task and Scene Synthesis

We construct executable training tasks through an LLM-guided pipeline comprising task proposal, scene generation, task decomposition, and reward generation. ACSE conditions this process on rollout failure evidence, current solver capabilities, and the mutation operators in Table 6, directing generation toward identified capability gaps.

Task Proposal. The LLM receives the original task, failure evidence, capability estimates, and the selected mutation operator. It proposes a structured task specification containing the instruction, required objects, objectives, and constraints. Task mutations (T) modify skill realizations, compositions, or temporal dependencies. Scene (S), language (W), and visual (V) mutations modify physical configurations, instruction wording, and observation conditions, respectively. Proposals remain within the supported task scope and specify which requirements must be preserved.

Scene Generation. Scene construction considers relevant assets, asset sizes, initial asset states, and spatial configurations. Required objects are retrieved from the RoboTwin (Chen et al., 2025) asset library according to their semantic roles and functional requirements. Compatible assets are configured with appropriate sizes, poses, joint states, and support or containment relations. Visual mutations are applied through rendering, camera, or observation parameters while preserving

Table 6: **Task mutation operators.** Failure evidence guides task (T), scene (S), language (W), and visual (V) mutations. Subscripts index operators rather than difficulty, with 0 marking the baseline. Task mutations retain the base-task scope, while language and visual mutations preserve semantics.

Dimension	Operator	Category	Targeted Task Generation
T	T_0	Original Structure	Preserve the original skill plan.
	T_1	Skill Realization	Select alternative goal-preserving skill realizations.
	T_2	Skill Composition	Recombine compatible skills within the task scope.
	T_3	Temporal Dependencies	Vary valid ordering and coordination constraints.
S	S_0	Original Scene	Preserve the original physical configuration.
	S_1	Asset Variation	Vary asset instances or sizes, preserving functionality.
	S_2	Initial State	Perturb object poses or joint states.
	S_3	Spatial Layout	Vary spatial relations while preserving feasibility.
	S_4	Clutter and Obstacles	Add distractors or obstacles.
W	W_0	Original Instruction	Preserve the original instruction.
	W_1	Lexical Variation	Substitute semantically equivalent words or phrases.
	W_2	Referential Variation	Vary unambiguous references to the same entities.
	W_3	Syntactic Variation	Vary sentence structure without changing meaning.
V	V_0	Original Observation	Preserve the original observation settings.
	V_1	Illumination	Vary lighting while preserving visibility.
	V_2	Surface Appearance	Vary textures without altering physics or target identity.
	V_3	Camera Viewpoint	Perturb camera poses while preserving observability.

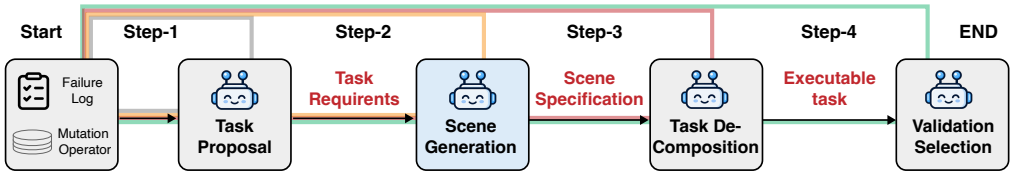


Figure 7: Highlights of performance for a wide range of benchmarks like, general benchmarks

physical properties and task semantics. Scene validation checks asset compatibility, stability, and reachability, as well as unintended collisions.

Task Decomposition. The LLM decomposes the proposed task into a graph of supported skills, specifying their goals and dependencies. Each skill is expanded into a sequence of grounded manipulation primitives with explicit object references and arm assignments. Bimanual programs additionally specify synchronization and temporal constraints. The resulting plan is checked for valid preconditions, consistent object ownership, and compatibility with the task objective.

Validation Selection. Finally, initialization and reference execution verify scene validity and task feasibility. Invalid candidates are revised or rejected. Solver rollouts estimate completion probabilities for capability-aware selection into the training pool, with subsequent outcomes guiding the next generation round.

B Implementation Details of the Robotic Solver

B.1 Model Architecture

We use AWAM as the base implementation of the Robotic Solver. It comprises an Agentic Reasoning Brain based on Qwen3.5 Qwen Team (2026), a World–Action Brain built on Wan2.2 Team Wan et al. (2025), and a Physical Head that extracts structured physical evidence from predicted world representations. The Harness manages execution history, experience memory, and information exchange between the modules.

As shown in Fig 8, the Agentic Reasoning Brain receives the task instruction, current visual observations, robot state, execution history, and relevant Harness memory. It produces a structured manipulation primitive specifying the next operation and its execution requirements. The World–

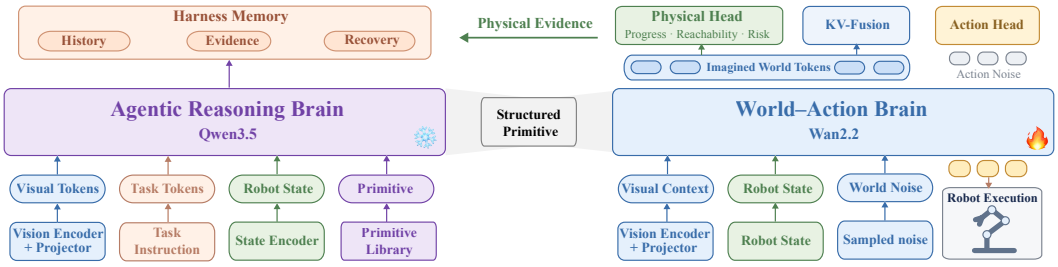


Figure 8: Architecture of the Robotic Solver. The Agentic Reasoning Brain generates structured primitives, while the World–Action Brain provides future-aware features for action generation and physical feedback. Harness memory retains interaction experience for subsequent reasoning.

Action Brain receives this primitive together with visual observations and the robot state, predicts future world representations, and generates an action chunk. The Physical Head estimates task outcomes, interaction states, and physical risks from the predicted representations.

The modules operate at different temporal scales. Qwen3.5 is invoked at primitive boundaries, whereas the World–Action Brain repeatedly generates action chunks during primitive execution. This organization allows the reasoning module to manage task progress while the world-action module handles observation-conditioned control.

B.2 Primitive Interface and Execution

The two brains communicate through structured primitives. Each primitive contains five fields: the operation type, target object, reference object, participating arm or arms, and execution constraints. Supported operations include grasping, lifting, carrying, placing, and coordinated bimanual manipulation. For example, a placement primitive identifies the object being moved, the reference object or destination, the responsible arm, and the required final spatial relation. Bimanual primitives additionally specify coordination requirements between the two arms.

At the beginning of each primitive stage, the Harness assembles the current observations, robot state, task progress, and relevant experience into the input context for Qwen3.5. The resulting primitive is passed to the World–Action Brain as its manipulation instruction. Hidden features from Qwen3.5 are not transferred to the action generation pathway.

The World–Action Brain executes the primitive through repeated action-chunk prediction. After each executed chunk, new observations and robot states are used for the next prediction. The primitive stage ends when its completion or failure condition is detected. The Harness then records the outcome and supplies the updated context to Qwen3.5, which selects the next primitive or revises the remaining plan.

B.3 World–Action Brain and Physical Head

The Wan2.2-based World–Action Brain contains a world representation stream and an action stream. The world stream processes visual context together with the primitive instruction and robot state to construct latent representations of future physical evolution. The action stream generates robot action chunks using these predicted world features as its conditioning input.

The two streams interact at selected layers. At each designated interaction layer, a future-context aggregation module collects world features and supplies them to the action stream. Other action layers process the action representation without an additional world-feature injection. The implementation therefore does not require dense interaction between the two streams at every layer.

The action generation pathway receives the primitive's semantic information through the conditioned world representations. It does not directly consume hidden features from Qwen3.5. The generated action chunk is passed to the robot controller, and subsequent predictions use the updated observations and robot state.

The Physical Head reads imagined world representations and produces structured predictions of task outcomes, interaction states, and physical risks. These predictions provide physical evidence for high-level reasoning and rollout evaluation. Observed execution outcomes supply supervision for updating the Physical Head during online adaptation.

B.4 Harness Memory and Online Updates

The Harness maintains interaction records across primitive stages. Each record contains the context before execution, the issued primitive, the executed actions, the resulting observations and robot state, and the execution outcome. Successful attempts, failures, and recovery behaviors are retained as experience for subsequent reasoning. When Qwen3.5 is invoked, relevant records are included in its input context to support primitive selection and plan revision.

Before ACSE evolution begins, demonstration-based training initializes the World–Action Brain and Physical Head. During online evolution, the Qwen3.5 parameters and the future prediction component remain frozen. Qwen3.5 adapts through changes in its Harness memory, while action generation is adapted through LoRA parameters in the Action Head. The Physical Head is updated using supervision derived from interaction outcomes.

Online trajectories are collected on tasks selected from the evolving training pool. The Action Head is optimized using FPO++, and the Physical Head receives its auxiliary prediction loss, as described in Sec. C. The frozen future prediction component supplies world features throughout these updates.

At the end of each evolution round, the updated action and physical prediction parameters, together with the accumulated Harness memory, are carried into the next round. Interaction records also provide the Generator with evidence about execution failures and task completion, connecting Solver updates to subsequent task generation.

C Interaction-Driven Solver Evolution

The Robotic Solver evolves by learning from capability-conditioned challenges supplied by the Generator. We instantiate it as AWAM and initialize its world-action and physical prediction components through imitation learning. This yields the initial Solver $\mathcal{S}_{\theta^0}$ with the execution capabilities needed to collect interaction evidence and initiate collaborative evolution.

Adaptation to Evolving Challenges. At round k, the Solver executes tasks sampled from the training pool constructed through feasibility validation and capability filtering. Unlike reinforcement learning on a fixed task distribution, these updates target challenges selected according to the Solver's current capabilities. For each intent under a shared task context, candidate rollouts $\{(A^j, W^j)\}_{j=1}^{K}$ are sampled from the old policy. Rewards are derived from execution outcomes and physical evidence and used to compute group-normalized advantages $\hat{A}^j$.

We optimize the action policy using the per-sample clipped objective adopted in FPO++ manipulation fine-tuning Yi et al. (2026) as follows::

$$\mathcal{L}_{\mathrm{FPO++}} = -\mathbb{E}\left[\frac{1}{KM}\sum_{j=1}^{K}\sum_{i=1}^{M}\min\Big(\rho_\theta^{j,i}\hat{A}^j, \mathrm{clip}\Big(\rho_\theta^{j,i}, 1-\eta, 1+\eta\Big)\hat{A}^j\Big)\right] \tag{11}$$

where η is the clipping parameter and $\rho_\theta^{j,i} = \exp(\ell_{\theta_{\mathrm{old}}}^{j,i} - \ell_\theta^{j,i})$. Here, $\ell_\theta^{j,i}$ denotes the conditional action flow-matching squared error, evaluated using the i-th of M flow-time and noise pairs shared by the old and current policies and summed over the action chunk. The expectation is taken over tasks sampled from the current training pool, old-policy rollouts, and flow-matching samples. The Physical Head receives additional supervision from interaction outcomes. The evolution objective is defined as follows:

$$\mathcal{L}_{\mathrm{evo}} = \mathcal{L}_{\mathrm{FPO++}} + \lambda'_{\mathrm{PHY}}\mathcal{L}_{\mathrm{PHY}} \tag{12}$$

where $\mathcal{L}_{\mathrm{PHY}}$ is the supervised physical prediction loss. During evolution, the Action Head and Physical Head are updated, while the parameters of the future prediction component and the Agentic Reasoning Brain remain frozen.

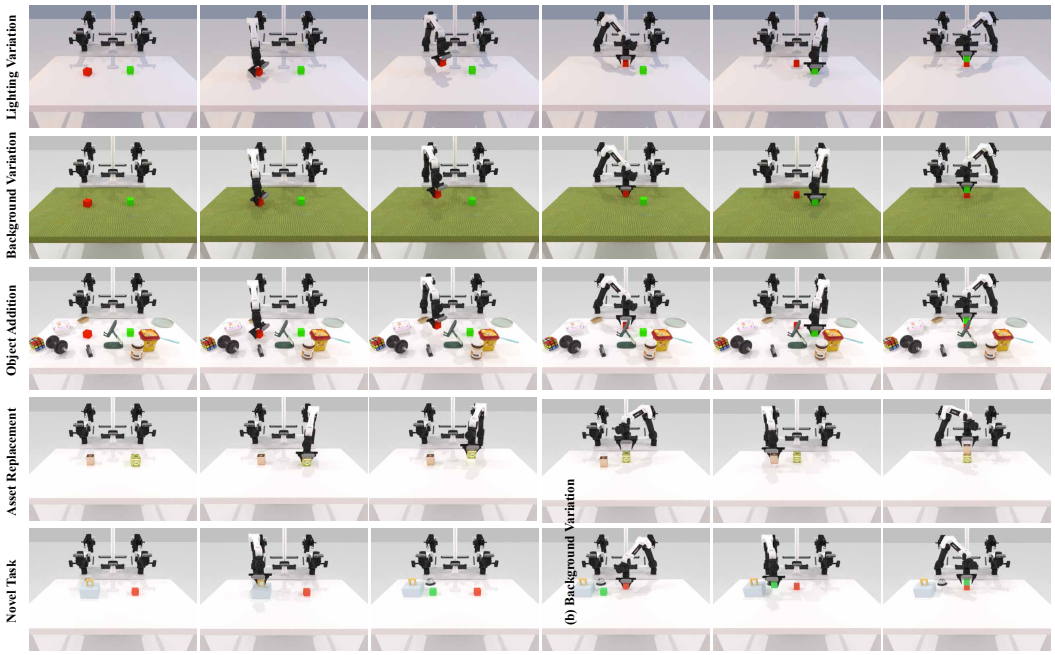


Figure 9: Generated task and scene variants for block manipulation, including lighting variation, background variation, distractor object addition, asset replacement, and novel task introduction. Each row depicts an execution sequence from left to right.

Experience Update and Generative Feedback. Alongside parameter adaptation, the solver incorporates accumulated interaction evidence into its reasoning memory to support subsequent planning and execution. The updated parameters and memory define the solver for the next round. The same interaction records provide capability evidence for the generator, informing subsequent task mutations and boundary filtering. As the solver masters previously challenging tasks, generation shifts toward its remaining capability gaps. This feedback loop couples the evolution of the training distribution with improvements in the solver's execution and reasoning capabilities.

D Qualitative Results

D.1 Visualization of Generated Tasks

Figure 9 illustrates generated variants of tableware manipulation tasks, including lighting variation, background variation, distractor object addition, asset replacement, and novel task introduction. These variants modify visual conditions, object properties, and manipulation goals, extending the original plate-and-cup task to diverse scenes and object configurations. Each row presents an execution sequence progressing from left to right.

Figure 10 presents the same five types of variation for block manipulation tasks. Lighting and background changes alter scene appearance, while added distractors introduce visual clutter. Asset replacement changes the manipulated objects, and novel tasks introduce new goals and skill compositions. Together, the two figures illustrate how task generation expands data diversity across visual environments, object types, and task structures, supporting model training and the evaluation of generalization.

D.2 General Long-Horizon Tasks Visualization

Fig. 11 illustrates the coordinated placement of two plate–cup pairs. The robot first grasps a plate and a cup with separate arms, moves the plate to the target area, and places the cup on it before repeating the sequence for the remaining pair. The timeline aligns object acquisition and placement skills with per-arm grasping and carrying primitives. This successful execution, without invoking

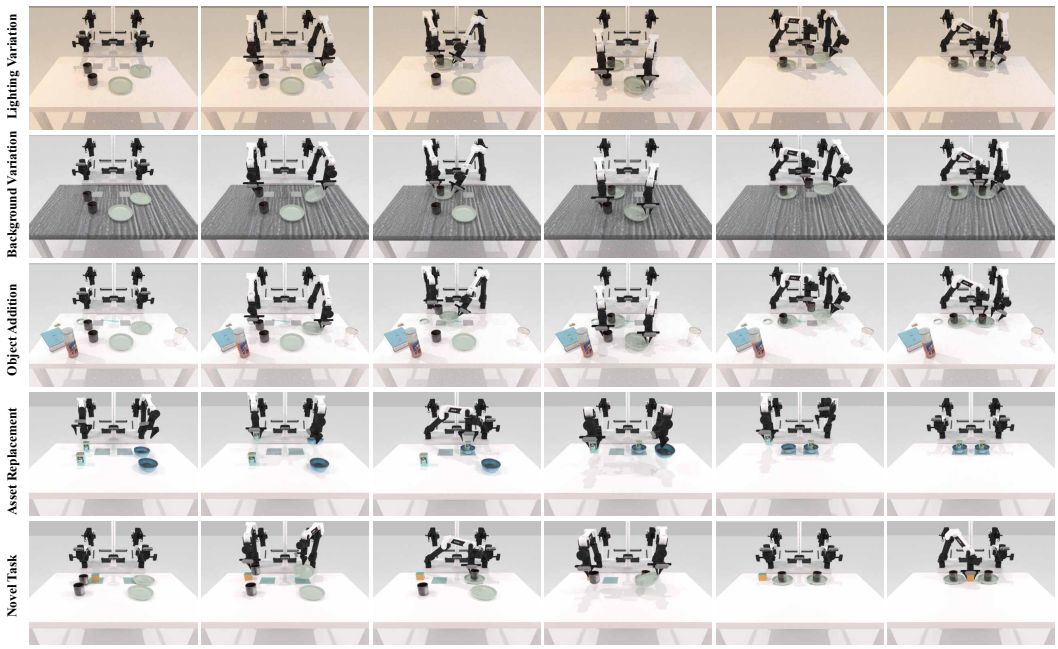


Figure 10: Examples of generated task and scene variants for bimanual manipulation, including lighting variation, background variation, distractor object addition, asset replacement, and novel task introduction. Each row shows an execution sequence progressing from left to right.



Figure 11: Successful bimanual execution of a sequential tableware manipulation task. The robot grasps and places two plate–cup pairs in sequence. The timeline aligns high-level skills with per-arm action primitives and execution steps. No recovery tools are invoked.

recovery tools, illustrates the model's ability to preserve task order and spatial relationships across successive subgoals.

D.3 Comparison of Solver Performance Before and After Evolution

Figures 12 and 14 compare the initial and evolved Solvers on cooking and plate-and-cup placement tasks. The execution frames and aligned timelines illustrate changes in task progress, skill transitions, and bimanual coordination following ACSE evolution.

Table 7: Real-world performance on two manipulation tasks. Task entries report successes out of 30 trials and success rates (%). Avg. denotes the mean success rate across tasks.

Method	**Real-World Tasks**		**Avg. (%)**
	Block Stacking (%)	Number Sorting (%)	
$\pi_{0.5}$	73.3	60.0	66.65
ACSE (Init)	83.4	76.6	80.0
ACSE (Evolved)	90.0	83.6	86.6

In the cooking task (Fig. 12), the initial Solver fails to transfer the food onto the plate and continues executing actions without completing the task. The evolved Solver successfully coordinates food placement, pan shaking, and pouring, completing the task in fewer execution steps. This example illustrates improved coordination across successive manipulation stages.

In the plate-and-cup task (Fig. 14), the initial Solver makes partial progress but subsequently repeats grasping attempts without completing the remaining placements. The evolved Solver completes two successive acquisition-and-placement sequences, placing both plates in the target areas with a cup on each. Its timeline shows coordinated per-arm primitives and transitions between subgoals, with no recovery tools invoked.

Together, these comparisons provide qualitative evidence that Solver evolution improves multi-stage execution, reduces unproductive repetition, and enables the completion of tasks that the initial Solver fails to finish.

D.4 Real-World Execution

We visualize real-world execution of the initial and evolved Solvers under matched task instructions and comparable scene configurations. Each sequence presents the initial observation, intermediate manipulation stages, and final outcome. Where perturbations are introduced, their timing and the subsequent responses are explicitly marked. These examples examine how capabilities acquired through ACSE evolution translate into physical task execution and recovery behavior.

D.5 Real-world Experiments

We assess ACSE on physical hardware using an AgileX Robotics Piper X platform. Two tasks test distinct manipulation requirements: *Block Stacking* involves positioning a red block on top of a yellow block, whereas *Number Sorting* requires placing numbered blocks in a specified sequence. The baselines are $\pi_{0.5}$ Black et al. (2025) and the initial Solver, allowing comparison with both an external policy and the starting point of self-evolution.

The evaluation comprises 180 trials, with 30 attempts for each of the three methods on each task. All methods share the robot setup, sensory inputs, instructions, and task-specific execution limits. We sample randomized object layouts and reuse them across methods to ensure comparable conditions. Parameters are held fixed throughout testing, and success requires completing the entire task within the allotted budget without human intervention. Success rates are summarized in Table 7. Qualitative execution comparisons further illustrate behavioral changes from the initial to the evolved Solver. It is as shown in Fig 13.

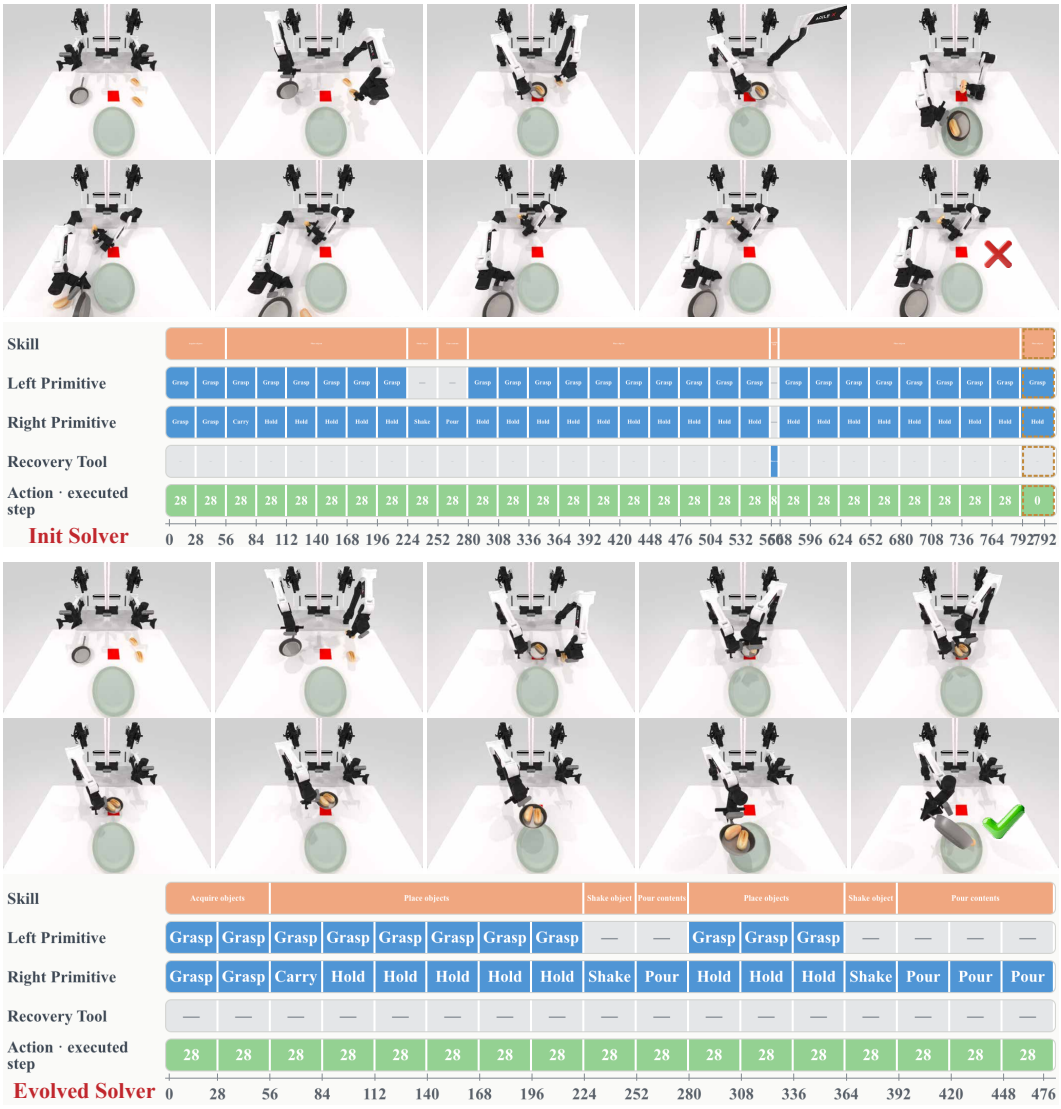


Figure 12: Cooking task execution before and after ACSE evolution. The initial Solver (top) fails to transfer the food onto the plate, whereas the evolved Solver (bottom) successfully completes the task through coordinated placement, shaking, and pouring. The timelines show the corresponding skills and per-arm primitives.

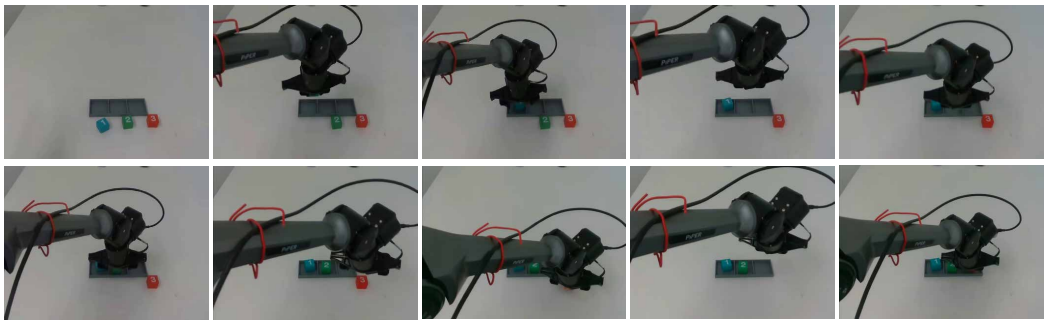


Figure 13: Number sorting performed by the ACSE-evolved Solver on a real robot. Numbered blocks are placed into tray slots following the specified order.

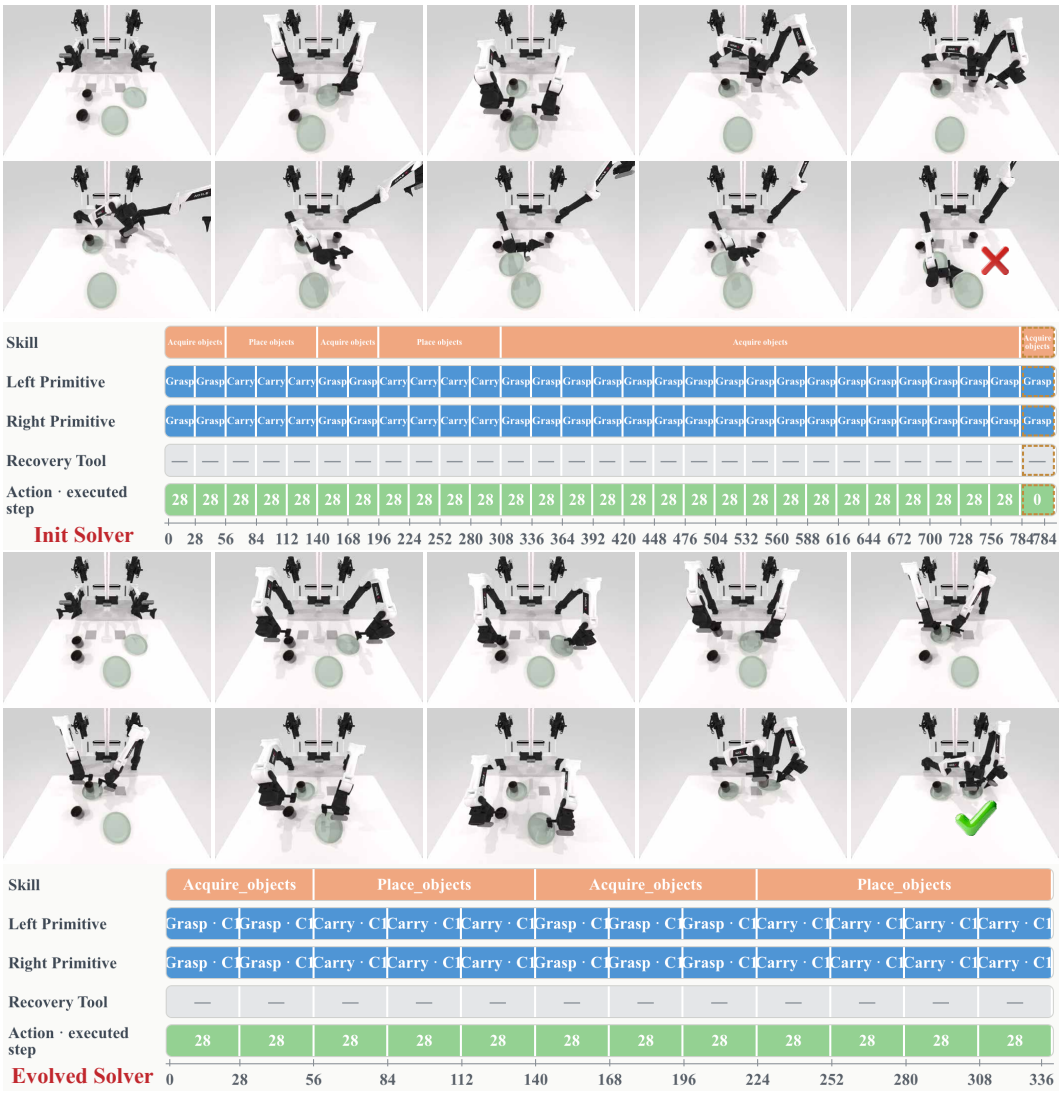


Figure 14: Plate-and-cup placement before and after ACSE evolution. The initial Solver (top) fails to complete the task and repeats grasping attempts, whereas the evolved Solver (bottom) successfully places both plate–cup pairs in the target areas. The timelines show the corresponding skills and coordinated per-arm primitives.